¡Robótica!

Guía del Estudiante

Versión en Castellano 1.5

Sobre la precisión de este texto:
Se realizó un gran esfuerzo para asegurar la precisión de este texto y los experimentos, pero puede haber errores aún. Si usted encuentra errores o algún tema que requiera información adicional, por favor infórmelo a stampsinclass@parallaxinc.com, así podemos continuar mejorando la calidad de nuestra documentación.

PARALLAX

Garantía

Parallax garantiza sus productos contra defectos en sus materiales o debidos a la fabricación por un período de 90 días. Si usted descubre un defecto, Parallax según corresponda, reparará, reemplazará o regresará el valor de la compra. Simplemente pida un número de autorización de regreso de mercadería (Return Merchandise Authorization, "RMA"), escriba el número en el exterior de la caja y envíela a Parallax. Por favor incluya su nombre, número telefónico, dirección y una descripción del problema. Nosotros le regresaremos su producto o el reemplazo, usando el mismo método de correo que usted usó para enviar el producto a Parallax.

Garantía de 14 días de regreso de dinero

Si dentro de los 14 días en que usted recibió su producto, encuentra que no es conveniente para sus necesidades, puede regresarlo, recibiendo un reembolso. Parallax regresará el precio de compra del producto, excluyendo los costos de manipuleo y correo. Esto no se aplica si el producto a sido alterado o dañado.

Derechos de Copia y Marcas Registradas

Esta documentación tiene derechos de copia Copyright 1999 por Parallax, Inc. BASIC Stamp (Estampilla BASIC) es una marca registrada de Parallax, Inc. Si usted decide usar el nombre "BASIC Stamp" en su página web o en material impreso, debe agregar la aclaración: "BASIC Stamp es una marca registrada de Parallax, Inc." Otros nombres de productos son marcas registradas de sus respectivos dueños.

Desvinculación de Responsabilidad

Parallax, Inc. no es responsable de daños por consecuencias, incidentes o daños especiales que resulten de cualquier violación de la garantía, bajo cualquier teoría legal, incluyendo pérdida de beneficio, tiempo, daño o reemplazo de equipo o propiedad y cualquier costo, recuperando, reprogramando o reproduciendo cualquier dato guardado o usado dentro de los productos Parallax. Parallax tampoco es responsable de cualquier daño personal, incluyendo vida o muerte, resultado del uso de cualquiera de nuestros productos. Usted tiene absoluta responsabilidad por la aplicación que desarrolle con el BASIC Stamp.

Acceso en Internet

Mantenemos sistemas de Internet para su uso. Estos pueden ser usados para obtener software, comunicarse con miembros de Parallax y comunicarse con otros clientes. Las rutas de acceso a la información se muestran a continuación:

E-mail: aalvarez@parallaxinc.com
Web: http://www.parallaxinc.com y http://www.stampsenclase.com

Lista de Discusión de BASIC Stamp en Internet

Mantenemos dos listas de discusión por e-mail para gente interesada en el BASIC Stamp. La lista trabaja así: mucha gente se suscribe a la lista y luego todas las preguntas y respuestas son distribuidas a todos los suscriptos. Es una forma rápida, divertida y gratis de discutir temas sobre el BASIC Stamp y obtener respuestas a preguntas técnicas. Para suscribirse a la lista de BASIC Stamp encuentre la información en www.parallaxinc.com. Esta lista genera aproximadamente 40 mensajes diarios.
También mantenemos una lista exclusiva para educadores que usan el BASIC Stamp en el aula. Usted puede unirse a esta lista en el sitio web http://www.stampsinclass.com. Esta lista genera aproximadamente 5 mensajes diarios.

Si busca una lista de discusión en castellano puede encontrarla en http://www.cursoderobotica.com.ar.

Contenido

Prefacio

Los robots son usados en la industria automotriz, médica, plantas fabriles y por supuesto, en las películas de ciencia-ficción. Construir y programar un robot es una resolución combinada de problemas, de electrónica y mecánica. Lo que usted experimentará con el Boe-Bot será útil para aplicaciones reales del uso del control robótico, la única diferencia es el tamaño y la sofisticación. Los principios de control electrónico, código de fuente y circuitos que usted usará, son muy similares, (y a veces idénticos), a las aplicaciones industriales desarrolladas por ingenieros electrónicos..

La palabra "robot" apareció por primera vez en una revista cómica de Checoslovaquia llamada *Rossum's Universal Robots* por Karel Capek en 1920. Los robots en esta obra, tendían a ser humanoides. De ahí en más se vieron en muchas historias de ciencia-ficción buenas, que los involucraban en revueltas contra la autoridad humana, lo cual requiere inteligencia. Esto cambió cuando General Motors instaló el primer robot en su planta de fabricación en 1961. En la ciencia-ficción o en la fabricación, la inteligencia es solamente instalada en un robot a través del programa.

Esta serie de experimentos de Robótica lo introducirá en conceptos de robótica básicos y programación, usando el robot de la Plaqueta de Educación (en adelante el "Boe-Bot"). Los experimentos comenzarán con la construcción del Boe-Bot. A continuación programaremos el Boe-Bot para que realice maniobras básicas y procederemos a agregarle sensores que le permitan al robot reaccionar al ambiente que lo rodea. El éxito del curriculum de robótica está en mostrarle a los estudiantes qué fácil es interesarse y emocionarse en los campos de la ciencia, información e ingeniería, a medida que diseñan, construyen y programan, un robot autónomo. El Board of Education Robot (Boe-Bot, Robot de Plaqueta de educación), provee a los estudiantes con un área de proyecto para construir y personalizar sus propios proyectos de programación, eléctricos o mecánicos. El uso de un robot para introducir el uso de microcontroladores es ideal, debido a que las salidas son casi siempre visibles y fáciles de personalizar.

La Plaqueta de Educación puede quitarse del Boe-Bot y ser usada en otros experimentos del currículum de Stamps en Clase. Esta portabilidad genera un ahorro en los costos, mejorando las oportunidades para explorar en robótica. La Plaqueta de Educación Rev A no fue diseñada originalmente para ser usada en un robot (el Boe-Bot fue creado en respuesta a la demanda de los clientes), así usted observará que hay que realizar una o dos tareas extras, debido a que cuando se hizo la Plaqueta de Educación, no se había considerado la construcción de robots. Específicamente los servos usan la fuente de alimentación sin regular, de 6 volts, de Vin en lugar de la regulada Vdd. Además se usa un capacitor de 3300 uF en Vss y Vdd para prevenir que el BASIC Stamp se reinicie debido a los picos de consumo de los servos. Modelos de plaquetas anteriores pueden también necesitar un cambio de regulador de voltaje, por el regulador de baja caída LM2940. Detalles de esto se muestran en el Apéndice E y los componentes de reemplazo se obtienen gratuitamente de Parallax.

La Plaqueta de Educación Rev B ha sido modificada para simplificar las aplicaciones en robótica, sin perjudicar las otras series de experimentos Stamps in Class (Stamps en Clase). Capacitores más grandes se han conectado al regulador de tensión LM2940 de la plaqueta Rev B, eliminando la necesidad de emplear un capacitor de 3300 µF que se usa en la Rev A. Cuatro puertos para servos se han agregado para permitir el conexionado de los mismos, sin ocupar espacio en el área de prototipo de la protoboard. Cada puerto está conectado a una línea de E/S (P12, P13, P14, o P15 dependiendo del puerto) y cada uno puede ser usado para controlar un servo independiente. Los servos están conectados a Vin, los 6 V sin regular del porta pilas, así que no es recomendable emplear una fuente de alimentación de mayor tensión, debido a que los servos pueden dañarse.

Algunos de los componentes de la Plaqueta de Educación Rev B han sido ligeramente desplazados, como el conector del puerto serial DB9 y el conector app-mod de 20 pines. Además, la conexión de Vdd del conector app-mod procede ahora del regulador de tensión del BASIC Stamp, mientras que la conexión de Vdd próxima a la protoboard se mantiene sin cambios. Para realizar los ejercicios de los libros de Stamps en Clase, use solamente la conexión de Vdd que está junto a la protoboard de la Plaqueta de Educación.

El currículum de Robótica será revisado y actualizado basándose en la realimentación, por parte de los estudiantes y educadores. Si usted quiere escribir un experimento para agregar en éste currículum, o tiene ideas para realizar mejoras, por favor envíelas a stampsinclass@parallaxinc.com. Nosotros haremos nuestro mayor esfuerzo para integrar sus ideas y asistirlo con el soporte técnico, soporte de ventas, o el entrenamiento que usted necesite. Si aceptamos su proyecto Boe-Bot , le enviaremos un Boe-Bot gratis.

Destinatarios y Guías para Profesores

El currículum Robótica fue creado para edades a partir de los 17 años, como texto siguiente a la guía "¿Qué es un Microcontrolador?". Como todos los currículums de Stamps en Clase, éste enseña técnicas nuevas y circuitos, con mínima superposición con los otros textos. Los temas introducidos en esta serie son: navegación básica del Boe-Bot bajo el control del programa, navegación en función de las señales generadas por los sensores, navegación usando realimentación y varias técnicas de control y navegación usando inteligencia artificial programada. Cada tema comienza con una introducción diseñada para lograr una comprensión conceptual, seguida por algunas experiencias prácticas. Aquellos que piensen profundizar en tecnología industrial, electrónica o robótica, recibirán grandes beneficios de las experiencias iniciales obtenidas con estos temas.

Expertos en distintos campos escriben independientemente cada juego de experimentos de Stamps en Clase, en formatos muy distintos. Como resultado, la profundidad y disponibilidad de guías para profesores varía ampliamente. Por favor contacte a Parallax, Inc. si tiene preguntas. Si está interesado en contribuir con material a la serie de libros de Stamps en Clase, por favor envíe su propuesta a stampsinclass@parallaxinc.com.

Derechos de Copia y Reproducción

Convenciones Tipográficas

- Instrucción de lista de control. El cuadradito señala una instrucción de procedimiento. Estas instrucciones podrían aparecer secuencialmente, como una lista de control, en cada actividad del libro.

✓ **TIP**	Preste atención y siga estas instrucciones. Las actividades le resultarán más fáciles y ahorrará tiempo. Nota del traductor: TIP=Pista o Consejo.

FYI	Este recuadro contiene información útil. Nota del traductor: FYI=For Your Information=Para Su Información.

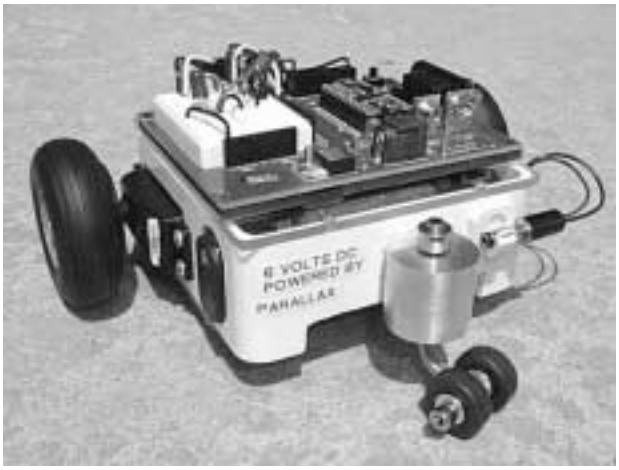
	**Precaución: siga estrictamente estas instrucciones, o puede terminar dañando su hardware.**

```
' Listados de Programas PBASIC.
```

```
      ' Extracto PBASIC del listado de un programa. Este tipo de extractos
      ' siempre están a continuación de un texto que explica qué es
      ' y cómo trabaja.
```

El código PBASIC dentro de un texto tiene el formato: **comando *argumento1, argumento2, etc***. Observe que el comando no está en cursiva, pero los argumentos sí.

Colaboradores de ¡Robótica!



Chuck Schoeffler Ph.D., escribió parte de la versión 1.2 de este currículum, en conjunto con Parallax, Inc. En esa época, el Dr Schoeffler era profesor del departamento de Educación Tecnológica Industrial de la Universidad de Idaho. Él diseñó el Robot de la Plaqueta de Educación (Boe-Bot) original, que se muestra en la foto y muchos robots similares con funciones únicas. Después de varias versiones, el diseño de Chuck fue adoptado como la base del Boe-Bot de Parallax, que es usado en este texto. Russ Miller de Parallax diseñó el Boe-Bot, basado en este prototipo.

Andrew Lindsay, Jefe de Robótica de Parallax, realizó la mayor parte de la versión 1.3 con tres objetivos en mente. Primero, incluir en todas las actividades del texto instrucciones procedimentales meticulosas. Segundo, exponer al lector y al estudiante a nuevos conceptos de circuitos, programación, ingeniería y robótica, en cada capítulo. Tercero, asegurarse que los experimentos pueden realizarse con un alto grado de éxito, usando la Plaqueta de Educación Rev A o Rev B. El pasante de verano del 2000 de Parallax, Branden Gunn, asistió en las ilustraciones de esta revisión.

Se agradece a Dale Kretzer por la revisión editorial, que fue incorporada en la v1.4. También se agradece a los siguientes participantes de la lista de discusión de Stamps en Clase por sus comentarios: Richard Breen, Robert Ang, Dwayne Tunnell, Marc Pierloz y Nagi Babu. Sus contribuciones fueron: correcciones de errores, sugerencias de edición útiles, o material nuevo. La corrección de errores y la mayoría de las sugerencias de edición fueron incorporadas en esta revisión. Material nuevo, incluyendo el indicador de batería baja y ejercicios mecánicos extra, aparecen en esta versión 1.5.

Si tiene sugerencias, piensa que encontró un error, o quiere contribuir con una actividad o capítulo para el próximo ¡Robótica! v1.6 o más textos de ¡Robótica!, contáctenos a stampsinclass@parallaxinc.com. Suscríbase a la lista de discusión de Stamps en Clase para enterarse de las últimas ofertas de hardware gratuito para los colaboradores de ¡Robótica!. Vea la sección de lista de discusión de BASIC Stamp en Internet para obtener la información para suscribirse.

Cada persona que trabaja en Parallax, Inc. Ha contribuido de alguna forma con Stamps en Clase. Un agradecimiento especial a todo el grupo de Parallax por todo lo que hicieron para que el programa de Stamps en Clase sea un éxito.

Traducción

La Versión en Castellano de ¡Robótica! V 1.5, es la traducción correspondiente al libro Robotics! V1.5.

Traducido y adaptado al castellano por Arístides A. Alvarez. Corrección y edición de la traducción: Ana M. Lusi y Arístides L. Alvarez. Si encuentra errores en el texto, contáctese con nosotros, para poder mejorar la calidad de la documentación en castellano.

e-mail: aalvarez@parallaxinc.com
Mar del Plata
Argentina

Sitios web en Castellano:
www.stampsenclase.com
www.cursoderobotica.com.ar

Capítulo 1: Construcción y Prueba de su Boe-Bot

Capítulo 1: Construcción y Prueba de su Boe-Bot

Imagine que arma su Boe-Bot y lo programa para que se mueva hacia adelante. Luego, todo lo que sucedería es que se movería medio centímetro y se detendría. Si no se siguen cuidadosamente las instrucciones, ésta sería una de las muchas dificultades que afrontaría con su Boe-Bot. Este capítulo le mostrará como:

- Conectar su BASIC Stamp 2 a la Plaqueta de Educación y hacerla funcionar.
- Usar y probar servos modificados y sin modificar.
- Modificar, calibrar y probar servos para que funcionen como motores del Boe-Bot, controlados por el BASIC Stamp 2.
- Armar su Boe-Bot.
- Programar el BASIC Stamp 2 para hacer que el Boe-Bot se mueva a lugares determinados.

El Capítulo 1 no trata solamente sobre el armado del Boe-Bot. También sirve para asegurarse de que su Boe-Bot funcione correctamente al ir probando los subsistemas claves en todo el proceso de montaje. Siguiendo estas instrucciones, obtendrá experiencias prácticas sobre ingeniería de sistemas y desarrollo de subsistemas, control y solución de problemas.

Sistemas, Subsistemas y Competencias de Robótica

Los estudiantes de escuelas técnicas y preparatorias que se presentan en varias competencias de robótica, aprenden lo que es desempeñarse como ingenieros. Trabajan en equipos desarrollando subsistemas, integrándolos en sistemas, controlando y solucionando problemas durante todo el proceso.

Algunas veces, la solución de problemas se convierte en la etapa más difícil del desarrollo de un robot. Muchos pueden desvelarse una noche, tratando de hacer que el robot trabaje como era de esperarse. Una vez un grupo estuvo cinco horas intentando hacer funcionar correctamente un robot luchador de Sumo, sin éxito. Más tarde, utilizando la pantalla de Debug (depuración de errores) del BASIC Stamp, la solución del error tomó menos de 5 minutos.

FYI	El término BASIC Stamp será usado a través del libro, para referirse al BASIC Stamp 2.

Cuando está diseñando y construyendo un robot, es mejor imaginarlo como un conjunto de sistemas, subsistemas y elementos básicos. Un buen ejemplo de un sistema que puede ser descompuesto en

subsistemas y elementos básicos podrían ser los servos del Boe-Bot. Como sistema, un par de servos modificados que funcionan como motores, trabajando juntos, hacen que el Boe-Bot se desplace. Cada servo puede ser visto como un subsistema. Cada servo tiene una pequeña plaqueta de circuito impreso en su interior, con componentes electrónicos. Este es un ejemplo de un subsistema dentro de otro subsistema. Cada componente electrónico, si no puede seguir siendo descompuesto en componentes más pequeños, podría considerarse un elemento básico. Cada servo también tiene un subsistema de engranajes. Un engranaje en particular no puede ser separado en más partes, así que será un elemento básico. Cada servo recibe señales eléctricas, que le dicen lo que debe hacer, desde el BASIC Stamp, el cerebro del Boe-Bot. El Basic Stamp es otro sistema.

Una de las actividades más importantes, cuando se hace un robot, es el desarrollo y prueba de cada subsistema individual, de un sistema dado. Luego, también deben realizarse pruebas a nivel del sistema, para asegurarse que todos los subsistemas trabajan conjuntamente en la forma que se esperaba. Por último, pero no menos importante, se realiza la integración de sistemas, asegurándose que los mismos funcionen coordinadamente. La prueba y la solución de problemas en cada fase del desarrollo, en niveles de sistema y subsistema es, hasta cierto punto, una habilidad que se perfecciona con la práctica. Al seguir las técnicas introducidas en este capítulo y en el resto del texto, se hallará en camino de obtener esta habilidad. Con la práctica, preferirá resolver problemas pequeños de cinco minutos y no un problema gigante de cinco horas.

Armar, Probar, Modificar, Probar, Armar, Probar...

El desarrollo de robots es un proceso iterativo en varios sentidos. Desarrollo "iterativo" significa probar repetidamente y ajustar algo hasta que trabaje como se planeaba. La clave del desarrollo iterativo está en que los resultados de las pruebas son usados para realizar ajustes finos. Luego se realizan más pruebas y ajustes en la siguiente "iteración" de pruebas. En este capítulo, el proceso iterativo será desarrollado, probado, ajustado si es necesario, luego desarrollado un poco más, probado nuevamente, etc. El objetivo principal es armar el Boe-Bot y hacerlo funcionar, sin tener que desarmarlo para realizar más pruebas y reparaciones.

Para el final del capítulo, habrá aprendido a programar su Boe-Bot para que se desplace hacia atrás y adelante y gire en su lugar. En el camino, probará y calibrará los servos para ajustar el movimiento hacia delante y atrás. Aunque la mayor parte de la calibración de los servos se realiza desarmándolos y siguiendo las instrucciones de la Actividad 3, también hay un ajuste fino que se realiza simplemente cambiando algunos de los números de los programas de ejemplo. Esta técnica es llamada "calibración por software" y la realizará en la Actividad 4 y la Actividad 6.

Hardware del Boe-Bot

Para todas las Actividades de este texto, necesitará una computadora personal (PC) con el sistema operativo Windows 95, 98, 2000, o NT4.

Herramientas Recomendadas

Las herramientas de la fila superior de la Figura 1.1 se recomiendan para las Actividades del Capítulo 1.

(1) Destornillador con punta Phillips 0 con punta endurecida, en buenas condiciones
(1) Destornillador con punta Phillips 1
(1) Llave combinada de ¼"
(1) Alicate de corte de 3" (conveniente)
- o -
Tenaza corta clavos
(1) Antiparras protectoras (no se muestran)

Las herramientas de la fila inferior serán útiles a partir de las Actividades del Capítulo 2.

(1) Pinza de puntas finas
(1) Corta/pela cable

Figura 1.1: Herramientas Recomendadas.

Antes de comenzar, haga un inventario de los componentes de su kit. El Apéndice A: Lista de Componentes y Suministros del Boe-Bot, le dirá las cantidades de cada componente que debería tener su kit. Si necesita ayuda para identificar cada pieza, use la contratapa de este texto, donde aparecen imágenes rotuladas de todos ellos. Todas las piezas de hardware usadas en el Capítulo 1 se muestran en la Figura 1.2.

- ❑ Separe las partes mostradas en la Figura 1.2 para usarlas en las seis actividades de este capítulo.

Nota para los alumnos que heredaron un Boe-Bot armado en un curso anterior:	• Cuando haga el inventario de componentes, concéntrese en el Kit de Componentes de ¡Robótica! (#28124) listado en el Apéndice A. Comience por el principio de la lista y deténgase cuando comienza el listado del hardware del Boe-Bot. • Es muy importante que continúe y realice los ejercicios en cada actividad de este capítulo, para asegurarse de que su Boe-Bot está funcionando correctamente. • Desarmar el Boe-Bot no será necesario, a menos que descubra un problema con los servos. • Aunque este capítulo lo guía en el montaje y prueba del Boe-Bot, todos los componentes que conectará y probará, son accesibles en un Boe-Bot previamente armado.

Lista de componentes del Capítulo 1:

A	(1)	Chasis del Boe-Bot
B	(1)	Porta pilas
C	(2)	Servos
D	(2)	Ruedas plásticas
E	(1)	Bolilla de polietileno
F	(2)	Bujes de goma 9/32"
G	(1)	Buje de goma 13/32"
H	(1)	Plaqueta de Educación y BASIC Stamp 2
I	(2)	Cubiertas de goma
J	(1)	Chaveta
K	(10)	Tuercas 4-40
L	(2)	Tornillos cabeza plana 4-40
M	8)	Tornillos 3/8" 4-40
N	(8)	Tornillos 1/4" 4-40
O	(4)	Separadores 1/2"
P	(1)	Cable Serial
Q	(4)	Pilas alcalinas AA
R	(1)	CD de Parallax

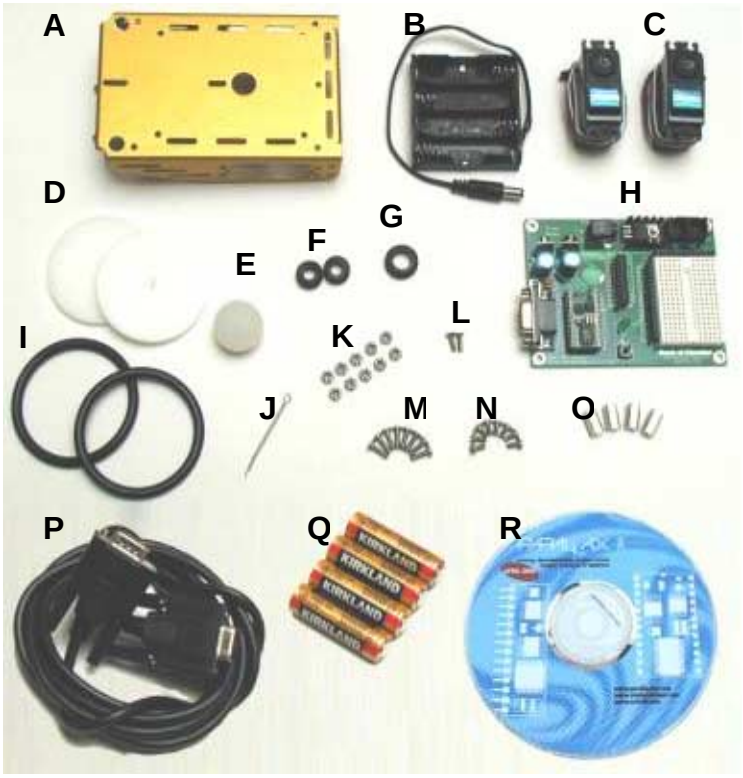


Figura 1.2: Componentes del Capítulo 1.

Actividad 1: Control de Comunicación PC - BASIC Stamp

Esta actividad tiene instrucciones para que usted siga al conectar el BASIC Stamp, la PC y el porta pilas a la Plaqueta de Educación. También tiene instrucciones resumidas sobre la instalación del Stamp Editor y la ejecución de un programa PBASIC simple. Esa actividad también provee un ejemplo simple de comprobación a nivel de sistema e integración. El BASIC Stamp y la PC son sistemas que han sido completamente desarrollados y probados. Su tarea es seguir las instrucciones para conectar los dos sistemas y lograr que se comuniquen.

Los comandos en el lenguaje de programación PBASIC son ingresados en el Stamp Editor. Cuando su programa PBASIC esté listo, también usará el Stamp Editor para tokenizar el programa (traducirlo a lenguaje simbólico) y descargarlo en el BASIC Stamp. Esto podría sonar complicado pero solamente toma dos clics del mouse. Para que estos clics funcionen, el Stamp Editor debe ser capaz de usar la PC para comunicarse con el BASIC Stamp, de forma de poder descargar el programa tokenizado. Dependiendo del programa PBASIC, su Boe-Bot puede ser instruido para realizar muchas tareas. Para tener una idea de las tareas para las que puede ser programado un Boe-Bot, simplemente eche un vistazo a las actividades listadas en el Contenido.

El Stamp Editor también tiene una característica llamada Debug Terminal (Terminal de Depuración). Puede usar la Debug Terminal para mostrar mensajes recibidos desde el BASIC Stamp y también mandar mensajes hacia el BASIC Stamp. La Debug Terminal será una de sus asistentes más útiles para la prueba y solución de problemas. Lograr que el BASIC Stamp se comunique con la Debug Terminal es muy fácil de hacer usando el lenguaje de programación PBASIC. Para comenzar, todo lo que se necesita es una línea de código PBASIC.

Presentación de la Plaqueta de Educación o Board of Education (BOE)

La abreviatura de Board of Education (Plaqueta de Educación) es BOE. Como hacemos un robot con la "Boe" lo llamamos Boe-Bot. La Figura 1.3 muestra (a) la BOE Rev A y (b) la BOE Rev B. Este capítulo cubre los pasos necesarios para armar y hacer funcionar al Boe-Bot usando la BOE Rev B. Si tiene una Rev A, el proceso es muy parecido. La Rev A no tiene puertos específicos para conectar los servos. El Apéndice D le mostrará como realizar los puertos para los servos, como un paso previo, para poder usarlos.

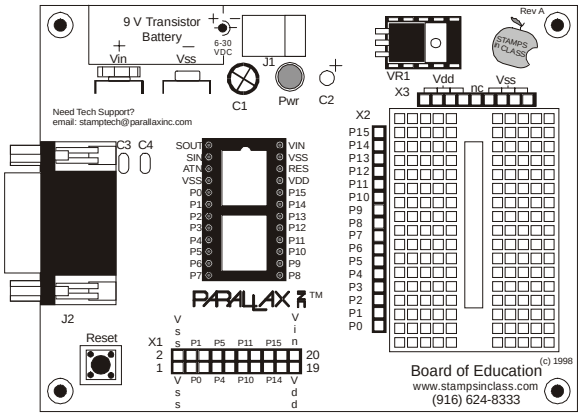
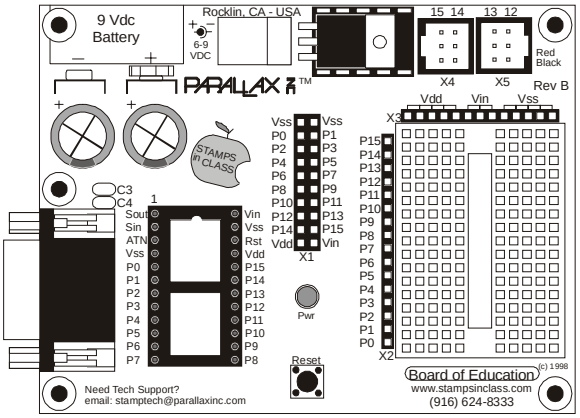


Figura 1.3: (a) BOE Rev A



(b) BOE Rev B.

- Controle en este momento que tipo de BOE tiene, Rev A o Rev B. Use las imágenes de la Figura 1.3 o busque una etiqueta que diga "Rev A" o "Rev B" cerca de la esquina superior derecha de su BOE.

Componentes

La Figura 1.4 muestra los componentes usados en la Actividad 1.

(4) Tornillos 4/40 ¼"
(4) Separadores
(1) Basic Stamp 2
(1) Plaqueta de Educación
(1) porta pilas
(4) Pilas alcalinas AA
(1) Cable Serial
(1) CD de Parallax

Figura 1.4: Componentes de la Actividad 1.

¡Constrúyalo!

La Figura 1.5 muestra el porta pilas antes y después de colocarle las pilas.

Siempre use pilas AA alcalinas de 1.5 V. No use pilas de Níquel-Cadmio (Ni-Cad) de 1.2 V.

- Coloque las pilas en el porta pilas con la polaridad que aparece impresa en el interior.

Figura 1.5: porta pilas con y sin pilas.

La Figura 1.6 muestra al BASIC Stamp 2 montado en su zócalo en la BOE (Plaqueta de Educación). El BASIC Stamp tiene medio círculo impreso en el centro de su extremo. Esto sirve como referencia en muchos circuitos integrados. Cuando coloque el BASIC Stamp en su zócalo, asegúrese que el semicírculo esté cercano a los rótulos Sout y Vin. A modo de verificación, asegúrese que el chip negro más grande con el rótulo PIC16C57 quede en la parte inferior, entre los rótulos P7 y P8.

- Si su BASIC Stamp y su BOE fueron enviados por separado, coloque el BASIC Stamp en el zócalo de la BOE como se muestra en la Figura 1.6. Asegúrese de alinear los pines (patitas) del BASIC Stamp con los agujeros del zócalo, luego presione el BASIC Stamp firmemente con su pulgar. Los pines del BASIC Stamp deberían introducirse unos 5 milímetros en el zócalo.

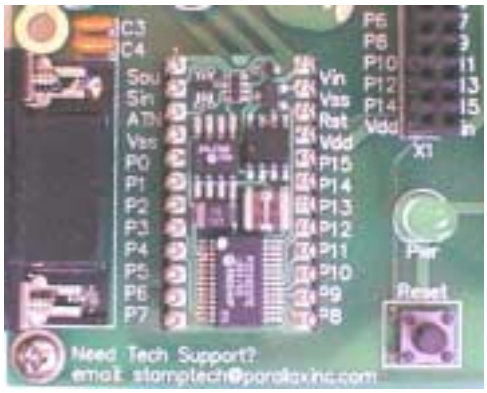
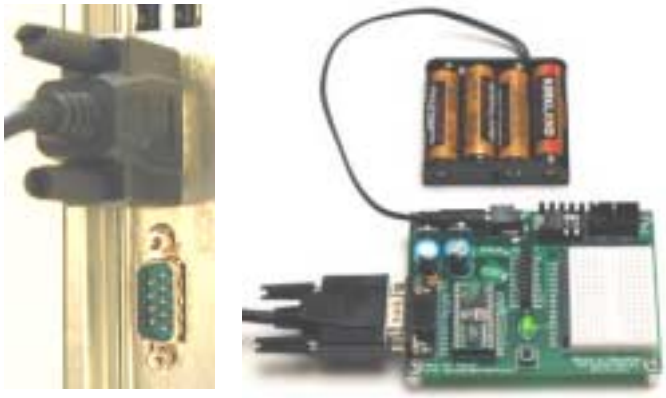


Figura 1.6: BASIC Stamp 2 insertado en el zócalo de la BOE.

La Figura 1.7 muestra (a), el cable serial conectado a un puerto com. en la parte trasera de una computadora y (b), el cable serial y el porta pilas conectados a la BOE, que está apoyada en una mesa sobre sus separadores. Estos separadores evitan que las soldaduras de la cara inferior de la BOE entren en contacto con la superficie de trabajo, que en el caso de ser conductiva, podría originar cortocircuitos.

- Use los tornillos #4-40 para sujetar los separadores en cada esquina de la cara inferior de la BOE.



(a) **(b)**

Figura 1.7: (a), Cable Serial conectado a un puerto com. y (b), BOE conectada al cable serial y porta pilas.

- Conecte la ficha hembra del cable serial en uno de los puertos seriales disponibles de su computadora.

- ❑ Conecte la ficha macho del cable serial en el zócalo DB9 de la BOE.
- ❑ Conecte la ficha del porta pilas a la BOE como en la Figura 1.7 (b).

Software y Primer Programa

Esta sección cubre los pasos para:

- Instalar el Editor del Stamp.
- Usar el Editor del Stamp para establecer una comunicación PC – BASIC Stamp.
- Ejecutar un ejemplo del programa en PBASIC que usa el comando **debug**.

> Nota: Estas instrucciones son para instalar el Stamp Editor desde el CD de Parallax. Una copia gratuita de este CD puede ser pedida a stampsinclass@parallaxinc.com. También puede obtener la última versión del Stamp Editor en la página de Downloads en el sitio web www.parallaxinc.com.

- ❑ Si aun no lo ha hecho, cargue el CD de Parallax en su unidad CDROM.

El CD de Parallax tiene un programa de exploración que se ejecuta automáticamente después que el CD colocado en la unidad CDROM de su computadora. La Figura 1.8 (a) muestra la pantalla del explorador cuando se inserta por primera vez el CD. La Figura 1.8 (b) muestra como se ve la pantalla cuando se ejecuta la Welcome application (aplicación de Bienvenida).

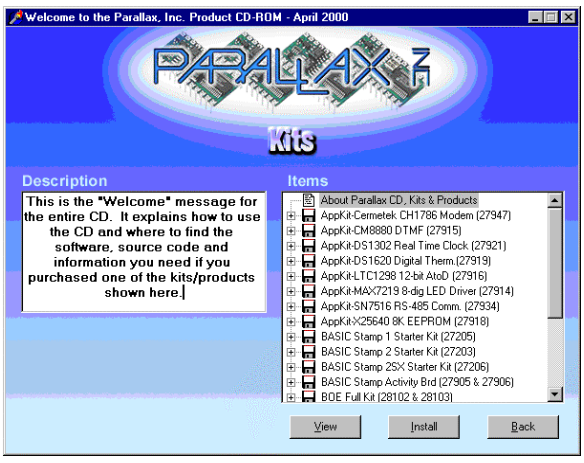
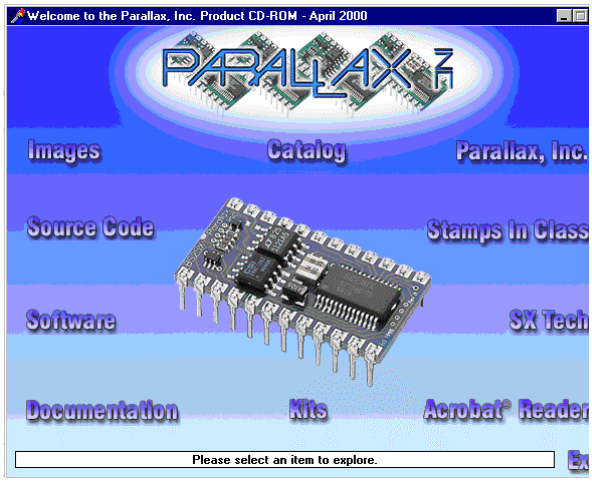


Figura 1.8: Aplicación de Bienvenida (a) Página de Kits y



(b) Página de Parallax.

- Si la Welcome application no se ejecutó automáticamente, le explicamos como hacerlo manualmente: Clic en el botón Inicio de la barra de tareas de Windows y seleccione Ejecutar. Cuando la ventana de Ejecutar aparezca, ingrese la letra correspondiente a su unidad CDROM, seguida por dos puntos, una barra invertida y el texto "Welcome.exe." (sin las comillas). Por ejemplo, si la letra de su unidad CDROM es D, escriba "D:\Welcome.exe." Clic en el botón Aceptar y la Welcome application se ejecutará.

- Si es la primera vez que ejecuta la Welcome application, se mostrará automáticamente un documento de texto sobre el CD de Parallax. Cuando termine de leer el documento de texto, minimícelo o ciérrelo para poder ver la página de Kits.

- Si no es la primera vez que ejecuta la Welcome application, se mostrará la página de Parallax en lugar de la de los Kits. Haga clic en el link Kits para abrir la página Kits.

- Estando en la página de Kits, haga clic en el ícono de disco rotulado "Boe-Bot Full Kit (28132)."

- Clic en el botón Install y seleccione Yes cuando la ventana Confirm le pregunte si quiere instalar los archivos con el mensaje: "Install selected files to C:\Parallax\BOE\?"

Después de instalar el software, ejecútelo siguiendo estos pasos:

- Haga clic en el botón Inicio de la barra de tareas de Windows y seleccione Ejecutar.

- Escriba "C:\Parallax\Stamp\Stampw_v1_091.exe," y haga clic en Aceptar.

- Si es la primera vez que ejecuta el software, aparecerá la ventana Edit Port List que se muestra en la Figura 1.9. Si sabe el número del puerto com. que está usando y no aparece en la lista, ingrese el número en el campo Com. #, luego presione Aceptar. Si sabe que cierto puerto com. que aparece en la lista de Known Ports está conectado a un módem, selecciónelo y luego oprima Delete. Caso contrario, haga clic en OK.

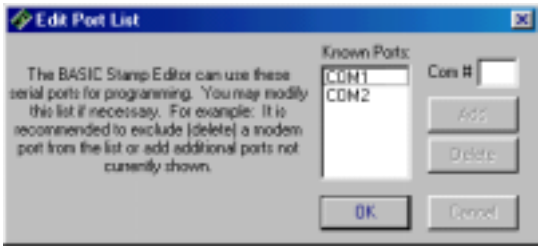


Figura 1.9: Ventana Edit Port List.

> ✓ **TIP** Puede modificar esta información más adelante en la ventana Preferences. Haga Clic en Edit y luego en Preferences. La configuración de puertos está en la pestaña Editor Operation.

La Figura 1.10 (a) muestra la ventana Stamp Editor (Editor del Stamp) que aparecerá a continuación.

- Haga Clic en Run y seleccione Identify (identificar) como se muestra en la Figura 1.10 (b).

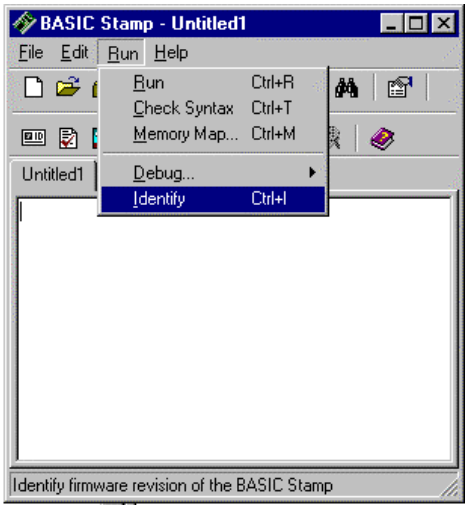


Figura 1.10: (a) Stamp Editor, (b) Stamp Editor con Run | Identify seleccionado.

Respuestas a Run | Identify (Ejecutar | Identificar)

Cuando todo está conectado y trabaja apropiadamente, aparecerá una ventana con el mensaje:

- "Information: Found BS2-IC (firmware v1.0.)." (Información: Se encontró BS2-IC, (firmware v1.0)

Este mensaje significa que el BASIC Stamp y la PC se están comunicando. Continúe a la siguiente sección titulada "Primer Programa."

Algunos de los mensajes que pueden aparecer son:

- "Error: Basic Stamp II detected but not Responding…Check power supply."
  - "Error. Basic Stamp II detectado pero no responde. Controle la alimentación."
- "Error: BASIC Stamp II not responding…Check serial cable connection. Check power supply."
  - "Error. BASIC Stamp II no responde. Controle la conexión del cable serial y la alimentación."

Primero siga las sugerencias de los mensajes de error. Si no se soluciona el problema, o si el mensaje de error no da sugerencias para solucionar el problema, vea el Apéndice B: Solución de Problemas de Comunicación entre la PC y el BASIC Stamp.

Primer Programa

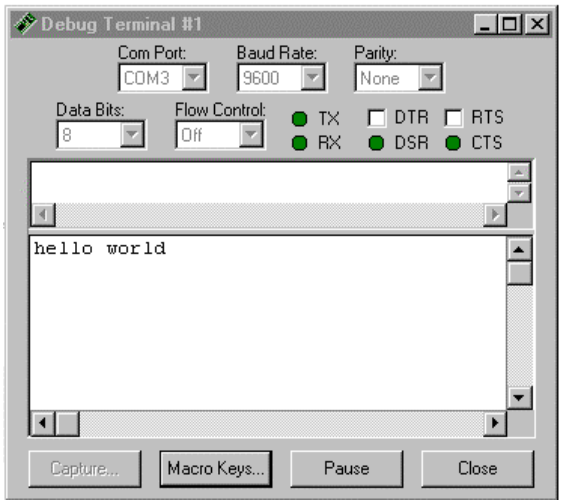
Su primer programa demostrará la habilidad del BASIC Stamp para comunicarse con el mundo exterior, usando la Debug Terminal (Terminal de Depuración). Esta herramienta puede ser usada para lograr la comunicación bidireccional entre su PC y el BASIC Stamp. Por ahora, nos centraremos en la programación del BASIC Stamp para enviar mensajes a la PC. En capítulos posteriores, el BASIC Stamp será programado para recibir mensajes provenientes de la PC.

```
' ¡Robótica! v1.5, Programa 1.1: ¡Hola mundo!

debug "hola mundo"
```

- Escriba el Programa 1.1 en el Stamp Editor como se muestra en la Figura 1.11 (a).
- Clic en Run y seleccione Run. La Debug Terminal #1 debería aparecer en una segunda ventana, como se muestra en la Figura 1.11 (b). Si escribe la versión del programa que se muestra más arriba, el texto aparecerá en castellano.

Figura 1.11: (a) Stamp Editor



(b) Debug Terminal.

Anteriormente, se había mencionado que se necesitaban solamente dos clics de mouse para tokenizar y descargar un programa PBASIC. Esto se refería a hacer clic en Run y luego seleccionar Run. El Stamp Editor muestra dos mensajes antes de abrir la Debug Window. El primero aparece en la barra de estado, en la esquina inferior derecha del Stamp Editor y muestra el mensaje "Tokenize Successful" (Tokenizado Exitosamente). Cuando un programa PBASIC es tokenizado, sus instrucciones son convertidas a instrucciones numéricas hexadecimales (tokens) que el chip intérprete del BASIC Stamp ejecuta. Luego, una ventana muestra el Download Progress (Progreso de la Descarga). Durante la descarga, el Stamp Editor envía el código hexadecimal al BASIC Stamp en forma de señales binarias, a través del cable serial. Luego aparece la Debug Terminal (Pantalla de Depuración) y se muestra el mensaje "hola mundo", que el BASIC Stamp estaba programado para mostrar.

Como Trabaja "Hola mundo"

- Antes de leer esta sección, vaya al Apéndice C: Referencia Rápida de PBASIC, o consulte el BASIC Stamp Manual (en Inglés) y lea sobre el comando **`debug`** introducido en este programa.

La primera línea del programa comienza con un apóstrofe. Esto significa que no es un comando, sino simplemente un comentario. El programa funciona exactamente igual si no se introducen los comentarios en los programas PBASIC del Stamp Editor.

La segunda línea comienza con un comando llamado **`debug`**. Cuando se ejecuta un programa que contiene un comando **`debug`**, el Stamp Editor abre una ventana de Debug Terminal. Cuando el BASIC Stamp ejecuta el comando **`debug`**, envía el mensaje "hola mundo" a la computadora a través del cable serial. El mensaje "hola mundo" es una cadena de texto, que es uno de los varios tipos de datos de salida que el BASIC Stamp puede ser programado para enviar, mediante el comando **`debug`**. Los datos de salida pueden tomar muchas formas distintas. Algunos ejemplos incluyen variables, constantes, expresiones, modificadores de formato y caracteres de control. La próxima sección le mostrará cómo usar el comando **`debug`** para enviar mensajes que incluyan muchos de estos tipos de datos.

Su Turno

El mismo comando **`debug`** puede ser usado para enviar más de un mensaje. Cada mensaje debe ser separado del anterior por una coma. Los caracteres de control, tales como **`cr`** pueden ser enviados para indicar un salto de línea hasta el margen izquierdo. Una o más constantes pueden ser mostradas con su propio formato. Un ejemplo de una constante podría ser el número 16. Indicadores de formato tales como **`dec`**, pueden ser usados para mostrar valores decimales. Los indicadores de formato **`dec1`**, **`dec2`** y hasta **`dec5`** pueden ser usados para mostrar valores decimales con un número fijo de dígitos. Otros ejemplos de indicadores de formato son **`bin`** y **`hex`**, útiles para mostrar datos en formatos binario o hexadecimal.

Hay muchos más indicadores de formato en el BASIC Stamp Manual, en la sección sobre el comando **debug**. Además de constantes, se pueden usar algunas expresiones matemáticas para resolver ciertos problemas, antes que el comando **debug** muestre los valores. Por ejemplo, la expresión **dec 7+9** dará el mismo resultado en la Debug Terminal que **dec 16**. El Programa 1.2 y los ejercicios que le siguen demuestran algunas de estas características.

```
' ¡Robotica! v1.5, Programa 1.2: Más comando debug.

debug cr, cr, "hola mundo", cr, cr      ' 2 líneas, hola mundo y 2 líneas más
debug dec 16                            ' muestra el valor decimal 16.
```

Para comprender mejor las variadas formas de usar **debug**, intente realizar las modificaciones al Programa 1.2 que se listan a continuación. Para guardar su trabajo, haga clic en File y seleccione Save, o Save As, dependiendo si quiere guardar un programa con el mismo nombre, o modificar su nombre o ubicación. Luego de realizar cada modificación en su programa, recuerde volver a ejecutar su programa haciendo clic en Run y seleccionando Run.

- Sustituya el indicador de formato **dec** por **dec3**. También pruebe con **dec2**, **bin** y **hex**. No olvide ejecutar su programa después de cada modificación para ver los resultados.
- Reemplace el número **16** con la expresión: **11 + 5**
- Pruebe con la expresión: **2*8**
- Al final del programa, agregue una tercera línea de código: **debug home, "hola mundo, otra vez"**

Actividad 2: Prueba de Servos

Como se mencionó anteriormente, los servos modificados trabajarán juntos para conformar el sistema de motorización del Boe-Bot. En esta actividad, cada servo será aislado y probado como un subsistema. Si los servos son nuevos o heredados de una clase anterior, dos o más de las preguntas de abajo deberán ser contestadas:

- ¿Los servos están modificados o no?
- Cuando el BASIC Stamp envía señales de control, ¿se comporta como se esperaba?
- Si están modificados, ¿están correctamente calibrados?
- ¿Están en buenas condiciones de trabajo?

Las pruebas simples que realizará en esta actividad, responderán estas preguntas y además indicarán que hacer a continuación. Por ejemplo, si los servos son nuevos, ellos funcionarán como servos, en lugar de funcionar como motores. Cada servo deberá ser desarmado, modificado y calibrado en la Actividad 3.

Si Está Probando un Boe-Bot Previamente Armado

Para probar los servos modificados de un Boe-Bot previamente armado por los estudiantes de una clase anterior, colóquelo de forma que las ruedas no toquen el piso. Puede controlar la salida de cada servo observando el movimiento de las ruedas del Boe-Bot. Si los servos están bien calibrados puede saltear la Actividad 3: Modificación y Calibración de Servos. Si los servos están un poco descalibrados, ciertos números de los programas de ejemplo de las Actividades 4 y 6 pueden ser modificados para calibrarlos. Esto se llama "calibración por software" y fue mencionado anteriormente. Por otro lado, las pruebas de esta actividad podrían indicar que uno o ambos servos necesitan ser desarmados y recalibrados. Si este es el caso, deberá quitar cada servo del Boe-Bot y seguir la Actividad 3.

Como Funcionan los Servos

Los servos para hobby son motores especiales con realimentación de posición interna. Su rango de giro es típicamente de 90° ó 180° y son especiales para aplicaciones donde se requiera un movimiento de mucha fuerza con precisión y a bajo costo. Son muy populares en los sistemas de control de autos, botes y aviones radio controlados. Los servos están diseñados para controlar la posición de un alerón en un avión o el timón en un bote radio controlado. En la actividad 3, modificaremos los servos del Boe-Bot para que controlen la velocidad y la dirección de las ruedas del Boe-Bot.

La Figura 1.12 muestra el circuito que se establece cuando un servo es conectado en el puerto para servos rotulado con un 12, en la esquina superior derecha de la BOE Rev B. Los cables red (rojo) y black (negro) se conectan a la fuente de alimentación y el white (blanco o algunas veces amarillo=yellow) es conectado a la fuente de señal. Cuando un servo es conectado en el puerto para servos 12, la fuente de señal para el servo es el pin P12 del BASIC Stamp.

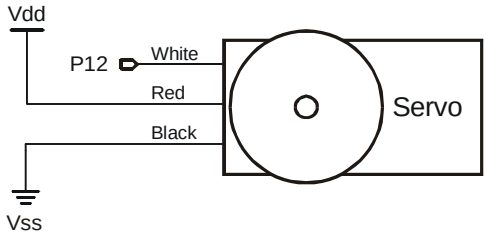


Figura 1.12: Esquema de conexionado del servo.

El BASIC Stamp puede ser programado para enviar señales a través de P12 que hagan que el servo se mueva. Con servos sin modificar, una señal de control dada hace que el engranaje de salida se mueva a un lugar en particular, dentro de su rango de movimiento de 180°.

El engranaje de salida de un servo modificado, por otro lado, girará continuamente con la misma señal.
Un servo sin modificar tiene una posición central, que se encuentra en la posición media entre ambos extremos del recorrido. La misma señal que hace que un servo sin modificar se mueva hacia su posición central es la que hace que un servo modificado no produzca ningún movimiento. Es posible tener control del sentido de giro de los servos modificados porque una señal que mueve un servo sin modificar a una posición en sentido antihorario, es la misma que hará girar continuamente a un servo modificado en el mismo sentido. Lo mismo se aplica para el sentido horario. También se puede controlar la velocidad de un servo modificado. Enviando una señal próxima a la posición central se moverá lentamente, cuando la señal se aleje de esta posición, aumenta la velocidad.

Componentes Extra

Se vuelven a usar los componentes de la Actividad 1. Además, se necesitan las partes listadas abajo y mostradas en la Figura 1.13.

(2) Servos

Figura 1.13: Servos Parallax.

¡Modifíquelos!

Muchos de ustedes han completado el libro ¿Qué es un Microcontrolador?, de la serie de Stamps en Clase, así que saben conectar el servo a la BOE basándose en el esquema mostrado en la Figura 1.12. Aún así, controlen el conexionado con el resto de las imágenes.

✓ **TIP**	La BOE Rev A no tiene puertos de conexión para servos. Si tiene una BOE Rev A, vea el Apéndice D: Construcción de Puertos para Servos en la Plaqueta de Educación Rev A.

- Empiece con el mismo circuito de la Actividad 1. La BOE debería estar conectada al porta pilas y a la computadora a través del cable serial.

Siga estas instrucciones cuando conecte un servo a su BOE:

- Desconecte el porta pilas de la BOE retirando el cable de la ficha de alimentación.

La Figura 1.14 (a) muestra un acercamiento de los puertos de los servos de la Rev B BOE. Los números en la parte superior indican el número de puerto. Si conecta un servo en el Puerto 12, significa que la línea de control del servo está conectada al pin de E/S P12. La línea que conecta a P12 es un trazo metálico en la BOE que une el pin superior del puerto del servo con el pin de entrada salida P12 del BASIC Stamp. Los rótulos del costado derecho de los puertos para servos están para asegurarse que los conecte en el sentido correcto. La Figura 1.14 (b) muestra un servo conectado al puerto 12 de forma que el cable negro (black) coincida con el rótulo black y que el cable rojo (red) coincida con el rótulo red. Aunque en la Figura 1.14 (b) el tercer cable aparece rotulado como "white" (blanco), este también puede ser amarillo (yellow).

Asegúrese de que los rótulos "Black" (Negro) y "Red" (Rojo) a la derecha de los puertos de servos, coincidan con los colores de los cables provenientes del servo, antes de conectarlo.

- Conecte el servo usando la Figura 1.14 como guía.

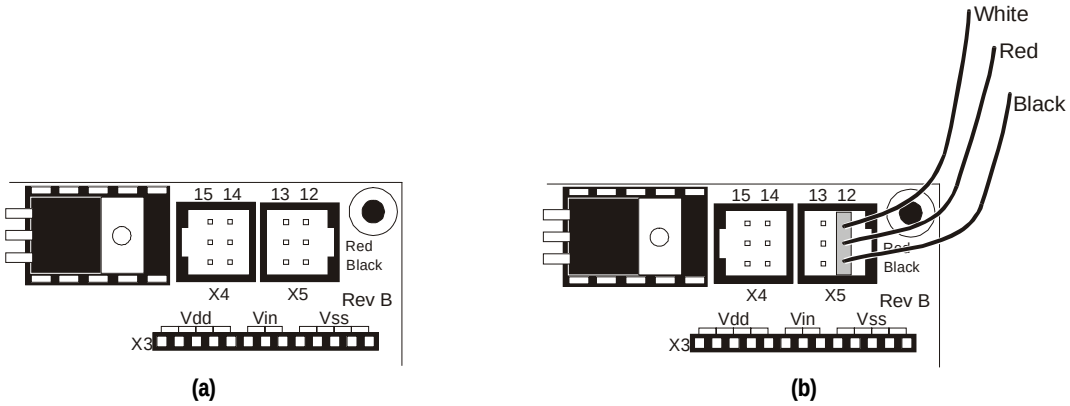


Figura 1.14: Puertos de Servos de la BOE Rev B (a) antes y (b) después de conectar un servo.

Cuando se conecta el porta pilas, el LED verde de la BOE puede indicarle si hay algún problema en su circuito.

Signos de advertencia:

Si la luz verde no se enciende, tiene menos intensidad que lo usual, o parpadea, desconecte inmediatamente el porta pilas y controle su cableado. Cualquiera de estos signos de advertencia podría indicar un problema en el conexionado, que podría ser peligroso para el servo o para el BASIC Stamp.

- Conecte el porta pilas a la BOE mientras mira la luz verde buscando si se produce algún inconveniente. Desconecte inmediatamente el porta pilas si se produce alguna de las advertencias que se mencionaron arriba.

Figura 1.15: BOE Rev B con un servo conectado.

Su conexionado debería verse como el de la Figura 1.15.
En este libro, nos referiremos al tiempo en unidades de segundos (s), milisegundos (ms) y microsegundos (µs). Los segundos son representados por una letra s minúscula. Así, un segundo se escribirá como: 1 s. Los milisegundos se abrevian como ms y representan una milésima de segundo. Un microsegundo es una millonésima de segundo. El recuadro de Milisegundos y Microsegundos de la derecha muestra estas equivalencias con fracciones y notación científica.

Milisegundos y Microsegundos

$$1\,\text{ms} = \frac{1}{1000}\,\text{s} = 1\times10^{-3}\,s$$

$$1\,\mu\text{s} = \frac{1}{1,000,000}\,\text{s} = 1\times10^{-6}\,s$$

Un nivel de tensión se mide en volts (voltios), que es abreviado con una letra V mayúscula. La BOE tiene conectores rotulados Vss, Vdd y Vin. Vss es llamada masa del sistema o tensión de referencia. Cuando se enchufa el porta pilas, Vss es conectado al terminal negativo. En lo que respecta a la BOE, BASIC Stamp y conexión serial a la computadora, Vss es siempre 0 V. Vin son los 6 V sin regular y están conectados al terminal positivo del porta pilas. Vdd representa a los 5 V regulados en la BOE por el regulador de tensión, que serán usados junto con Vss para alimentar los circuitos que se construyan en la protoboard.

Tensiones y Rótulos de la BOE

Vss = 0 V (masa)
Vdd = 5 V (regulados)
Vin = 6 V (sin regular)

Use solamente los conectores de Vdd que se encuentran junto a la protoboard de la BOE, para las actividades de este libro. No use la conexión de Vdd del conector de 20 pines app-mod.

Programar los Servos para que Permanezcan Estáticos y "Centrados"

La señal de control que el BASIC Stamp envía a la línea de control del servo es llamada "tren de pulsos," y se muestra en la Figura 1.16. El BASIC Stamp puede ser programado para producir esta forma de onda en cualquiera de sus pines de E/S. En esta actividad, usaremos el pin de E/S P12, que ya se encuentra conectado al puerto de servos 12 por un trazo metálico de la Plaqueta de Educación. Primero, el BASIC Stamp fija la tensión de P12 a 0 V (estado bajo) por 20 ms. Luego, fija la tensión de P12 a 5 V (estado alto) durante 1.5 ms. luego, repite el ciclo con un estado bajo por 20 ms y una salida en estado alto por 1.5 ms y así sucesivamente.

Este tren de pulsos tiene un tiempo de encendido de 1.5 ms y un tiempo de apagado de 20 ms. El tiempo de encendido o de estado alto se suele llamar ancho del pulso. Cuando hablamos de pulsos se sobreentiende que son pulsos positivos. Los pulsos negativos se toman como tiempo de descanso o separación entre pulsos positivos. Los trenes de pulsos tienen otros parámetros técnicos tales como ciclo de trabajo. Estos temas se desarrollan en Analógico y Digital Básicos, Experimento 6.

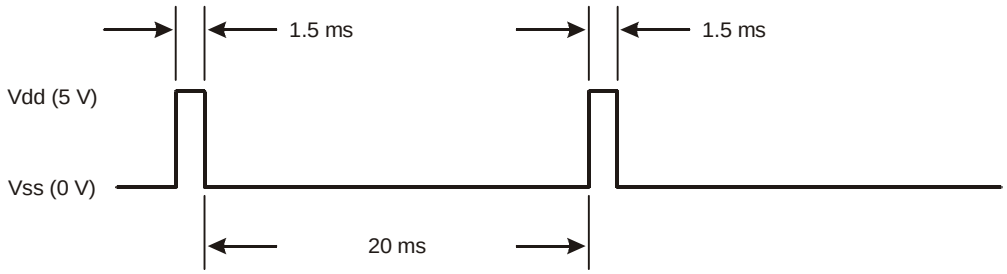


Figura 1.16: Tren de pulsos.

Un servo tiene una salida analógica, lo que significa que puede girar dentro de un rango de valores continuo. Así trabajan los servos del Boe-Bot. Si un servo sin modificar recibe pulsos de 1 ms, su engranaje de salida rotará en sentido horario lo más lejos que pueda, hasta llegar a los 180° de su rango de movimiento. Si el servo recibe pulsos de 2 ms, rotará en sentido antihorario hasta su límite de movimiento. Los pulsos de 1.5 ms harán girar a un servo sin modificar hasta quedar estable en el centro de su rango de movimiento de 180°. Esta se llama la posición central del servo. Pulsos de 1.3 ms harán que el engranaje de salida de un servo sin modificar rote ligeramente en sentido horario, partiendo desde el centro y pulsos de 1.7 ms harán lo mismo pero en sentido antihorario.

FYI El ancho del pulso es lo que controla el movimiento del servo. La separación entre pulsos puede variar entre 10 y 40 ms sin afectar el rendimiento del servo.

Luego de que un servo es modificado, se le pueden enviar pulsos para hacerlo girar constantemente. Los anchos de pulsos para los servos modificados típicamente oscilan entre 1.3 y 1.7 ms para velocidad máxima en sentido horario y antihorario respectivamente. El ancho del pulso central sigue siendo 1.5 ms y un servo modificado y calibrado correctamente debería permanecer estacionario cuando recibe pulsos de 1.5 ms. Si gira muy lentamente en respuesta a estos pulsos puede realizarse un ajuste por software. Si gira rápidamente con los mismos pulsos, el servo deberá ser desarmado y recalibrado.

Comencemos programando al BASIC Stamp para enviar el tren de pulsos "centrales" mostrados en la Figura 1.16. Esto debería hacer que un servo girase hasta su posición central y permaneciese allí. Un servo modificado quedaría estático, o rotaría muy lentamente.

- Tome notas del comportamiento del servo, para realizar posteriormente un diagnóstico.
- Ingrese el Programa 1.3 en el Stamp Editor.

```
' ¡Robótica! v1.5, Programa 1.3: Programa para centrar los servos.

low 12                          ' Configura a P12 como salida en estado bajo

bucle:                          ' Rótulo que indica el inicio de un bucle

  pulsout 12, 750               ' envía pulsos de 1.5 ms por P12
  pause 20                      ' cada 20 ms.

goto bucle                      ' envía el programa hacia el rótulo "bucle:".
```

- Guarde el programa usando un nombre descriptivo, preferiblemente incluyendo "1_3", para indicar que es el Programa 1.3.
- Ejecute el programa.
- Observe y registre el comportamiento de su servo.

Cómo Trabaja el Programa

- Busque cada una de las siguientes funciones en el Apéndice C, Referencia Rápida de PBASIC o en el BASIC Stamp Manual antes de continuar: **`low, pulsout, pause, goto.`**

Como antes, la primer línea del programa comienza con un apóstrofe, convirtiéndola en un comentario en lugar de un comando PBASIC. También hay un comentario al costado derecho de cada comando PBASIC. Estos comentarios también comienzan con apóstrofes y dan una breve explicación de lo que hace ese. Cuando ingrese los comandos del programa en el Stamp Editor, no necesita incluir comentarios. Los comentarios y otras formas de documentar sus programas se vuelven importantes si escribe códigos más complejos, o si alguien más debe trabajar con el mismo programa. Si quiere ahorrar tiempo, ingrese solamente los comandos en el Stamp Editor.

El comando **`low 12`** hace dos cosas. Configura al pin de E/S P12 del BASIC Stamp como salida, luego fija su valor de salida en bajo (0 V). Como salida, P12 puede enviar señales de tensión, lo opuesto a cuando se usa **`input`**, que significa que P12 podrá únicamente leer señales. Fijar el valor de la salida en bajo significa que la tensión que envía P12 es igual a Vss, 0 volts. Si fuera usado el comando **`high 12`**, P12 enviaría un estado alto, que sería igual a Vdd, 5 volts.

Cuando se escribe una palabra que no es ningún comando de PBASIC seguida por dos puntos, se denomina etiqueta. A medida que se familiarice con el PBASIC, empezará a reconocer automáticamente que palabras son comandos y cuáles son etiquetas. La etiqueta **`bucle:`** trabaja junto con el comando **`goto bucle`** del final del programa. La etiqueta **`bucle:`** está marcando un lugar del programa y cada vez que el programa ejecuta el comando **`goto bucle`**, automáticamente comienza a ejecutar los comandos posteriores a la etiqueta **`bucle:`**. El resultado es que los comandos **`pulsout`** y **`pause`** se ejecutan una y otra vez, enviando la señal que centra al servo.

La estructura del programa **`bucle:...goto bucle`** es llamada bucle infinito. Es infinito porque el código se repite una y otra vez sin posibilidad de que alguna instrucción detenga la ejecución del mismo conjunto de instrucciones. Con programas de computadoras normales, esto es un problema. Sin embargo, los bucles son muy usados en la programación de microcontroladores. De hecho, la mayoría de los programas de microcontroladores, incluyendo los de este libro, se escriben dentro de un bucle infinito. Con el BASIC Stamp, siempre puede detener un bucle infinito desconectando la alimentación o ejecutando (cargando) un programa diferente.

El comando **`pulsout 12, 750`** envía un pulso de 1.5 ms. El comando tiene dos argumentos, el número de pin de E/S y la duración. El uso del número de pin de E/S es obvio; el número 12 se refiere al pin de E/S P12. ¿Qué hay sobre el argumento que indica una duración de **`750`**? ¿Cómo es que corresponde a un pulso de 1.5 ms? La respuesta es que el argumento de duración del comando **`pulsout`** está especificado en incrementos

de 2 µs. Así, si quiere que el pulso dure 1.5 ms, debe usar un número que le dé 1.5 ms cuando sea multiplicado por los incrementos de 2 µs. Esta es la demostración de que un **`pulsout`** de **`750`** es el correcto.

$$\begin{aligned} 750 \times 2\,\mu s &= 750 \times (2 \times 10^{-6})s \\ &= 1500 \times 10^{-6}\,s \\ &= 1.5 \times 10^{-3}\,s \\ &= 1.5\ ms \end{aligned}$$

El comando **`pause 20`** es mucho más obvio. Esto es debido a que el argumento de duración del comando **`pause`** está especificado en incrementos de ms. Así, si quiere una pausa de 20 ms, **`pause 20`** cumplirá esa tarea.

Su turno

Hacen falta dos pruebas más para decidir qué hacer con los servos. Registre sus observaciones sobre cada prueba.

- Ejecute el Programa 1.4 que se muestra abajo. Tome notas del comportamiento del servo.
- Reemplace el argumento de duración del comando **`pulsout`** con el número 850 y reejecute el programa. Tome notas del comportamiento del servo otra vez.
- Después de usar los Programas 1.3 y 1.4 para probar un servo, repita las mismas pruebas para el otro.

```
' ¡Robotica! v1.5, Programa 1.4: Programa para centrar el servo.

low 12                          ' Configura P12 como salida baja.

bucle:                          ' Etiqueta hacia donde saltar

  pulsout 12, 650               ' Envía pulsos de 1,3 ms por P12
  pause 20                      ' cada 20 ms.

goto bucle                      ' Salta hacia "bucle: ".
```

Diagnóstico del servo

Dependiendo de cada servo y de su historia, puede haber exhibido uno de varios comportamientos distintos. A continuación damos una lista de los comportamientos más comunes. Cada comportamiento es seguido por

una explicación de las acciones que se deberán realizar. Si ambos servos no se comportan igual, asegúrese de encontrar el comportamiento que se ajuste para cada uno en particular.

Comportamiento del Servo:

En respuesta a pulsos de 1.5 ms (pulsout 12, 750), el engranaje de salida del servo gira a cierta posición (posición central) y permanece quieto. Hace fuerza si usted quiere alejarlo de esa posición. En respuesta a pulsos de 1.7 ms (pulsout 12, 850), el servo rota ligeramente en sentido antihorario desde la posición central y se detiene. En respuesta a pulsos de 1.3 ms (pulsout 12, 650), el servo gira ligeramente en sentido horario y se detiene.

Explicación:

Este es el funcionamiento normal de un servo sin modificar. Modifique y calibre este servo siguiendo las instrucciones de la Actividad 3.

Comportamiento del Servo:

En respuesta a pulsos de 1.5 ms, el servo permanece estacionario o gira lenta y constantemente. Muy lentamente sería si da menos de dos vueltas completas por minuto. En respuesta a pulsos de 1.7 ms, el servo gira en sentido antihorario y se mantiene girando rápidamente, aproximadamente a 50 revoluciones por minuto (RPM). En respuesta a pulsos de 1.3 ms, el servo se mantiene girando a aproximadamente la misma velocidad, pero en sentido horario.

Explicación:

El servo ya fue modificado y calibrado. La calibración existente servirá para todas las actividades de este texto. Pase a la Actividad 4.

Comportamiento del Servo:

El servo gira constantemente a más de 2 RPM cuando es usado el comando pulsout 12, 750.

Explicación:

El servo fue modificado pero no calibrado apropiadamente, o puede haberse descalibrado. Un servo puede descalibrarse si es expuesto a fuertes vibraciones, golpes fuertes y algunas otras condiciones extraordinarias. Siguiendo los pasos de la Actividad 3 solucionará el problema.

Comportamiento del Servo:

Cuando el servo gira, hace un ruido intermitente.

Explicación:

Si el ruido se repite una o dos veces por giro completo, indica que el servo fue modificado pero una parte del tope del engranaje no fue quitado correctamente. Para arreglarlo, repase la Actividad 3 y asegúrese de remover completamente el tope del engranaje. Si el servo gira continuamente pero hace un zumbido, puede tener algún problema mecánico. Cuando realice la Actividad 3, asegúrese que los engranajes del servo estén limpios y que nada obstruye su movimiento, antes de rearmarlos. Si esto no soluciona el problema, se deberá reemplazar el servo.

Comportamiento del Servo:

El servo nunca se movió.

Explicación:

Hay muchas explicaciones posibles para esto. Si un servo no hace nada, intente con el otro. Si uno funciona pero el otro no, el problema se aísla hacia un servo. Revise las conexiones eléctricas desde la ficha de conexión hasta el servo.

Si no funciona ningún servo, el problema podría estar en el programa o en las conexiones eléctricas entre el pin de E/S del BASIC Stamp y el servo. Los usuarios de la BOE Rev B deberían asegurarse de que su servo esté correctamente conectado en su puerto correspondiente. Los usuarios de la BOE Rev A deberían revisar su cableado.

Para revisar el código de los programas, agregue el comando **`debug "comenzando", cr`** al principio del programa. Agregue el comando **`debug "repitiendo", cr`** dentro de la rutina **`bucle:...goto bucle`**. Estos programas le ayudarán a ver qué está haciendo el programa. Cuando ejecute el programa modificado, preste mucha atención. ¿Muestra la ventana Debug Terminal una línea que dice "comenzando" seguida por una rápida serie de líneas que dicen "repitiendo"? Si es así, su programa se está ejecutando correctamente. Aún puede haber un error en uno de los comandos. Asegúrese de usar el argumento 12 cuando se refiera al pin que controla al servo. Si nada funciona, intente usar el puerto de servos 13. Tendrá que cambiar todos los argumentos que indican el pin para el servo del programa, de 12 a 13.

Si aparece el mensaje "comenzando" una y otra vez, significa que el BASIC Stamp se está reseteando (reiniciando) debido a una condición llamada brownout (reinicio por baja tensión). Una inspección a fondo de la BOE, dará como resultado pilas bajas o pilas recargables de 1.2 V. Asegúrese de usar pilas alcalinas de 1.5 V e inténtelo nuevamente. En una Rev A BOE, esto también puede ser causado por una conexión incorrecta o defectuosa del capacitor de 3300 µF. Asegúrese de que el puerto del servo y el capacitor estén cableados como se describe en el Apéndice D: Construcción de Puertos para Servos en una Plaqueta de Educación Rev A. Los usuarios de la BOE Rev A también deberían controlar su regulador de tensión. Consulte el Apéndice E: Kit de Actualización del Regulador de Tensión de la Plaqueta de Educación Rev A.

El mensaje "comenzando" podría aparecer seguido de un mensaje "repitiendo" y luego la ventana Debug Terminal podría dejar de imprimir mensajes. También podrían aparecer los mensajes "comenzando" y "repitiendo" en forma alternada. Ambos problemas indican fallas en la escritura de la etiqueta **`bucle:`** o en el comando **`goto bucle`**. Revise su programa.

Actividad 3: Modificación de Servos

Si usted esta comenzando con un servo sin modificar, esta actividad le enseñará a desarmar, modificar, calibrar y probar cada servo. Si su objetivo es recalibrar un servo, seguirá los mismos pasos excepto la parte de la modificación. Es muy importante hacer las pruebas de verificación después que los servos han sido calibrados. Este es un ejemplo de desarrollo y prueba iterativo.

Más Sobre el Funcionamiento de los Servos

¿Qué es Un...

Potenciómetro

Es un resistor ajustable con una pequeña perilla, que puede ser girada para modificar el valor de su resistencia. Puede aprender más sobre potenciómetros en ¿Qué es un Microcontrolador?, Experimento 4 y Analógico y Digital Básicos, Experimentos 1 y 3.

¿Alguna vez se preguntó cómo sabe un servo en qué posición se encuentra? ¿Cómo hace un servo sin modificar para saber hacia que posición girar cada vez que recibe pulsos de cierto ancho? Su funcionamiento se basa en la realimentación. Un servo usa un potenciómetro conectado a su engranaje de salida como sensor de posición.

El circuito de un servo sin modificar compara el valor de su potenciómetro con el ancho de los pulsos que recibe por la línea de control. Luego activa su motor para corregir cualquier diferencia entre ambos valores. Hace esto con cada pulso, así que si intenta desplazar de su posición a la palanca de control de un servo, su circuito detectará una diferencia entre el valor del potenciómetro y los pulsos, causando el encendido del motor, para anular esta diferencia. Esto sucede tan rápidamente que usted solamente siente que la palanca del servo resiste la fuerza que intenta desplazarla de su posición.

La información que el servo recibe del potenciómetro se llama realimentación. La comparación entre el valor del potenciómetro y el ancho del pulso y las correcciones que origina, son los componentes del proceso llamado control de lazo cerrado. Para hacer que los servos giren como motores, hay que eliminar la realimentación. Cuando la unión entre el engranaje de salida y el potenciómetro es eliminada, el funcionamiento se realizará a lazo abierto en lugar de a lazo cerrado.

Una vez que los servos funcionan a lazo abierto, aún no serán capaces de realizar giros completos hasta que se elimine el tope mecánico de uno de sus engranajes. Los engranajes de nylon son bastante blandos y el tope puede ser eliminado con facilidad. Es fácil hacerlo con alicates, pero cualquier herramienta de corte servirá.

Antes de rearmar el servo, el potenciómetro debe ser ajustado para que el engranaje de salida quede inmóvil cuando el servo reciba la señal para ir a la posición central. Estos son los pulsos de 1.5 ms que enviábamos en el ejercicio anterior. Esta vez, cuando el tren de pulsos de 1.5 ms sea enviado, el motor del servo comenzará a girar. Luego puede ajustar manualmente el potenciómetro para lograr que el servo se detenga. Este proceso es llamado centrado.

Luego de centrar el servo y quitarle la realimentación y el tope de engranaje, se procederá a armarlo y probarlo.

¡Modifíquelo!

El servo debe ser removido para su modificación y/o calibración. La Figura 1.17 muestra (a) las partes delantera (b) y trasera del servo. La vista frontal muestra la palanca de control sujetada al engranaje de salida por un tornillo. Hay que desmontar esta palanca. La vista posterior muestra los cuatro tornillos que sujetan todo el servo.

Figura 1.17: Vistas del servo: (a) delantera **(b) posterior.**

Desarmado del Servo

- Desconecte el porta pilas de la BOE.
- Desconecte el servo de la BOE.
- Usando un destornillador Phillips, quite el tornillo que sujeta la palanca de control del servo.
- Si está trabajando con un Boe-Bot construido en una clase anterior, quite el tornillo del centro de la rueda y quite la rueda. Luego, retire los cuatro tornillos que sujetan el servo al chasis.
- La palanca de control del servo no se volverá a colocar cuando se lo arme. El tornillo se usará para sujetar la rueda del Boe-Bot.
- Quite los cuatro tornillos de la parte trasera del servo usando un destornillador Phillips pequeño.
- Ponga el servo de forma que el engranaje de salida quede hacia arriba.

- ❑ Quite la cubierta de engranajes del servo de forma que el sistema de engranajes quede a la vista.

Los engranajes montados se muestran en la Figura 1.18 (a). La Figura 1.18 (b) muestra los nombres de cada parte con la que trabajará.

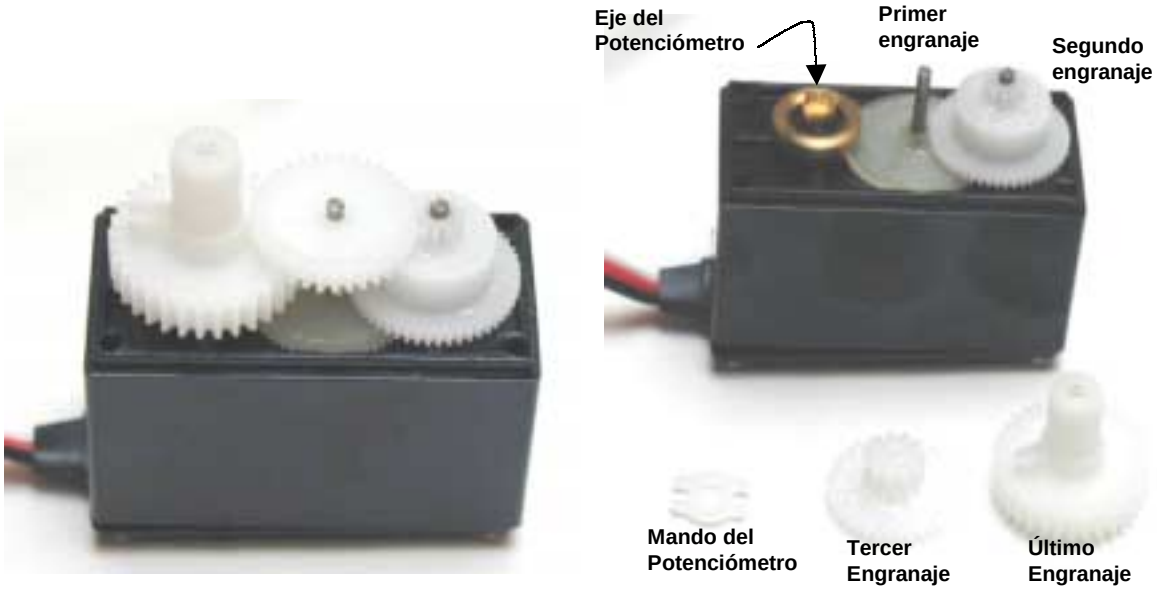


Figura 1.18: (a), Servo sin la cubierta de engranajes y (b), componentes removidos con su identificación.

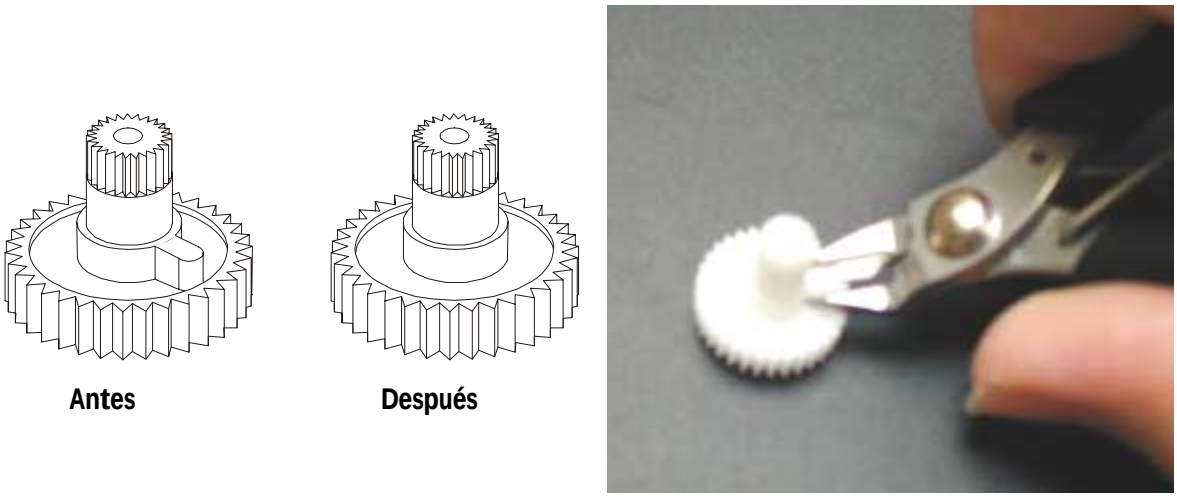
- ❑ Quite el último y el tercer engranaje como se muestra en la Figura 1.18 (b).

Modificación del Servo

- ❑ Encuentre y quite el mando del potenciómetro que se muestra en la Figura 1.18 (b). Se encontrará en el eje del potenciómetro o en la base del último engranaje. Esta pieza es el vínculo de la realimentación que debe ser removido para que el servo funcione a lazo abierto.

La Figura 1.19 (a) muestra el último engranaje e indica el tope que debe ser eliminado. La Figura 1.19 (b) muestra a un alicate cortando el tope.

Seguridad: Use anteojos de seguridad cuando remueva el tope.



(a) **(b)**

Figura 1.19: (a), Diagrama del último engranaje (con y sin tope) y (b), corte del tope.

- Use sus herramientas de corte para remover el tope del último engranaje.
- Asegúrese de no dejar remanentes del tope sobre el engranaje. Si la punta de su alicate de corte no llega a retirar los últimos pedazos del tope, inténtelo con otra herramienta. Asegúrese de limpiar todas las limaduras y desperdicios que se hayan adherido al engranaje antes de rearmar el servo.

Calibración del Servo

Antes de rearmar el servo, se lo debe centrar. En otras palabras, el potenciómetro debe ser ajustado para que el servo permanezca estático cuando recibe un ancho de pulso de 1,5 ms.

- Conecte el servo desarmado a la BOE como se muestra en la Figura 1.20.
- Conecte el porta pilas a la BOE.
- Abra el Programa 1.3 haciendo clic en el menú File del Stamp Editor y seleccionando Open, o rescriba el programa en el Stamp Editor.

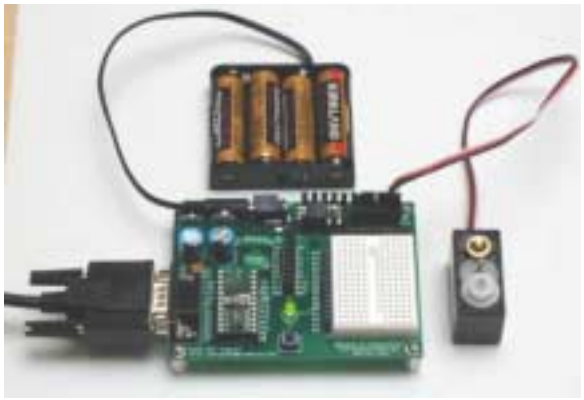


Figura 1.20: Servo preparado para ser centrado.

Importante:	**Asegúrese de que el Programa 1.3 sea ingresado exactamente como se muestra abajo antes de ejecutarlo. Especialmente, controle que el argumento de duración del pulso sea de 750.**

- Después de asegurarse de haber ingresado correctamente el Programa 1.3, ejecútelo.

```
' ¡Robotica! v1.5, Programa 1.3: Programa para centrado de servo.

low 12                          ' Ajusta a P12 como salida baja.

bucle:                          ' Etiqueta hacia donde saltar

  pulsout 12, 750               ' Envía pulsos de 1.5 ms por P12
  pause 20                      ' cada 20 ms.

goto bucle                      ' Salta hacia la etiqueta "bucle:".
```

El motor del servo comenzará a funcionar y el primer y segundo engranaje girarán.

- Gire el eje del potenciómetro.

Si gira el eje en una dirección, la velocidad del servo aumentará y en la dirección opuesta, disminuirá.

- Gire cuidadosamente el eje del potenciómetro hasta que el servo se detenga.
- Desconecte el servo de la BOE.
- Ármelo, sin colocar el mando del potenciómetro. Tampoco coloque la palanca de salida del servo.
- Ambos servos deben ser modificados y calibrados. Repita todos los pasos para cada servo.

Prueba del Servo

- Conecte el servo rearmado en la BOE como se muestra en la Figura 1.21.
- Ahora que los servos están rearmados, asegúrese de que el BASIC Stamp esté ejecutando el Programa 1.3 y conecte cada servo en el puerto 12.

Cuando cada servo es conectado en el puerto 12, su engranaje de salida debería permanecer quieto o rotar lentamente (menos de 2 RPM).

- Pruebe cada servo con el Programa 1.4. Asegúrese de usar un argumento de duración de **`pulsout`** igual a **`650`**.

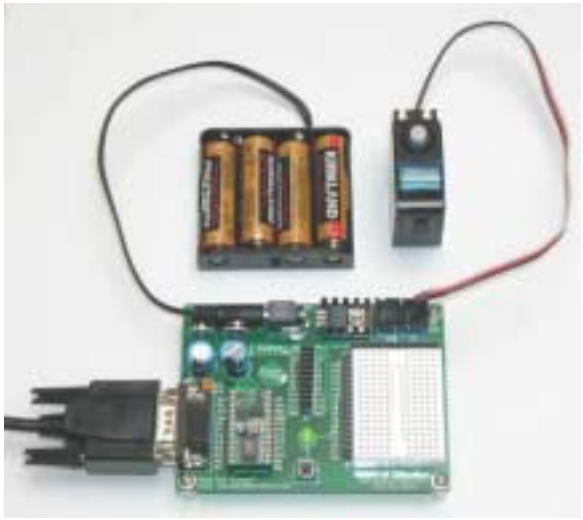


Figura 1.21: Servo rearmado listo para ser probado.

El engranaje de salida debería rotar bastante rápido, a aproximadamente 50 RPM.

- Pruebe cada servo con el Programa 1.4, pero con un argumento de duración de **`pulsout`** de **`850`**.

Esta vez, el engranaje de salida debería rotar en sentido antihorario a aproximadamente 50 RPM.

Si el servo pasó estas tres pruebas, está calibrado. Si falló en alguna, repita la sección de Diagnóstico de Servos de la Actividad 2.

Actividad 4: Centrado de Servos – Calibración por Software

La calibración por software involucra probar un servo con duraciones de pulsos cercanas al valor de **`pulsout`** central ideal de 750 (1.5 ms). Lo que se busca es el valor de duración más exacto que logre detener totalmente el engranaje de salida del servo. Esto es especialmente importante si alguno de los servos giró lentamente cuando se ejecutó el Programa 1.3 después de ser rearmado. Este es un ejemplo de una segunda iteración en el proceso de calibración.

- Modifique el Programa 1.4 para que la duración del comando **`pulsout`** sea 735 y reejecute el programa. El servo debería girar lentamente en sentido horario.
- Modifique el argumento de duración a 736 y reejecute el programa. Repita para 737, 738 y así hasta 765.
- Tome nota de los argumentos de duración que hacen detener al servo y de aquellos que lo ponen nuevamente en movimiento.
- Tome el promedio de los anchos de pulso que detienen al servo y llame a ese valor ancho de pulso central.

El verdadero valor del argumento de duración para centrar el servo está en algún punto entre el valor que detiene al servo y el valor que lo hace girar nuevamente. Por ejemplo, si el servo deja de girar con 749 y comienza a girar nuevamente con 751, es fácil. Use 750 como ancho central para ese servo.

Pongamos un ejemplo un poco más complicado. ¿Qué sucede si el servo no deja de girar hasta un argumento de duración de 752? Luego, ¿qué sucede si no empieza a girar nuevamente hasta un argumento de duración de 757? La verdadera duración para centrar este servo modificado está en algún lugar entre 752 y 757, ¿pero dónde? Tomando el promedio de los dos números, la duración sería:

$$\text{duración promedio} = \frac{752 + 757}{2} = 754{,}5$$

El argumento de duración del comando **`pulsout`** solamente puede tomar valores enteros entre 0 y 65535. Algunos ejemplos de argumentos de duración válidos son: 0, 1, 2,...,754, 755,.... así que, 754,5 no puede ser usado como argumento válido. La duración promedio de **`pulsout`** debe ser redondeada, ¿pero hacia dónde? Aunque es una práctica común redondear hacia arriba si el valor decimal es 0,5 o superior, el servo puede no responder a esta práctica. Podría determinar experimentalmente hacia qué sentido redondear probando duraciones de **`758`** y **`751`** en su programa. Continuando con el ejemplo, si 751 hace que el servo gire más rápido, redondee hacia arriba, tomando a 755 como duración central. Por otro lado, si 758 lo hace girar más rápido, redondee hacia abajo, tomando a 754 como duración central.

- Escriba el argumento de duración del pulso central para cada servo sobre una etiqueta autoadhesiva y péguela sobre el encapsulado plástico. De esta forma, cuando escriba futuros programas, sabrá la duración central de cada servo.

- Si viene siguiendo el manual para armar el Boe-Bot, quite los tornillos y separadores de su BOE y sepárelos para usarlos en la Actividad 5.

- Si está usando un Boe-Bot que fue armado por estudiantes de una clase anterior, vuelva a colocar los servos en el chasis. Luego, pase directamente a la Actividad 6:

Actividad 5: Construcción del Boe-Bot

Montando el Hardware de la Parte Superior

La Figura 1.22 muestra el chasis del Boe-Bot, el hardware de la parte superior y tornillos de montaje.

Lista de materiales:

(1) Chasis Boe-Bot
(4) Separadores
(4) Tornillos 1/4" 4-40
(2) Pasa cables de goma 9/32"
(1) Pasa cables de goma 13/32"

Figura 1.22: Chasis y hardware de la parte superior.

Montaje:

La Figura 1.23 muestra al hardware superior montado en el chasis del Boe-Bot. Cada pasa cables de goma tiene una ranura en su cara exterior que la mantiene sujeta al agujero del chasis del Boe-Bot.

- Inserte la goma pasa cables de 13/32" en el agujero central del chasis del Boe-Bot.
- Inserte las otras dos gomas de 9/32" en los agujeros de las esquinas como se muestra en la fotografía.
- Use los cuatro tornillos 1/4" 4-40 para colocar los cuatro separadores.

Figura 1.23: Hardware superior montado.

Colocando los Servos

Lista de Materiales:

La Figura 1.24 muestra los servos modificados y el hardware para fijarlos al chasis.

(1) Chasis de Boe-Bot parcialmente armado
(2) Servos
(8) Tornillos 3/8" 4-40
(8) Tuercas 4-40

Figura 1.24: Servos y hardware para su montaje.

Armado:

La Figura 1.25 muestra los servos colocados en el chasis.

- Use los ocho tornillos 3/8" 4-40 y tuercas para fijar los servos al chasis del Boe-Bot.

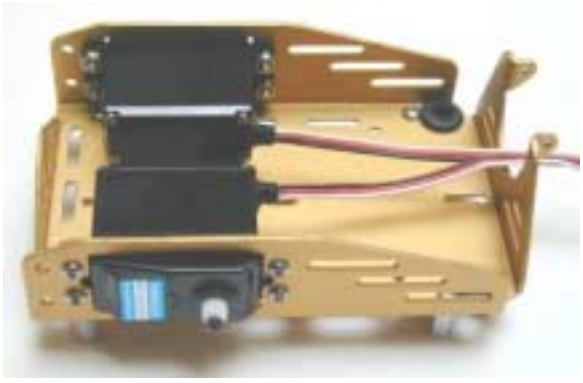


Figura 1.25: Servos montados en el chasis.

Montaje del Porta Pilas

La Figura 1.26 muestra el porta pilas y los tornillos necesarios para su montaje.

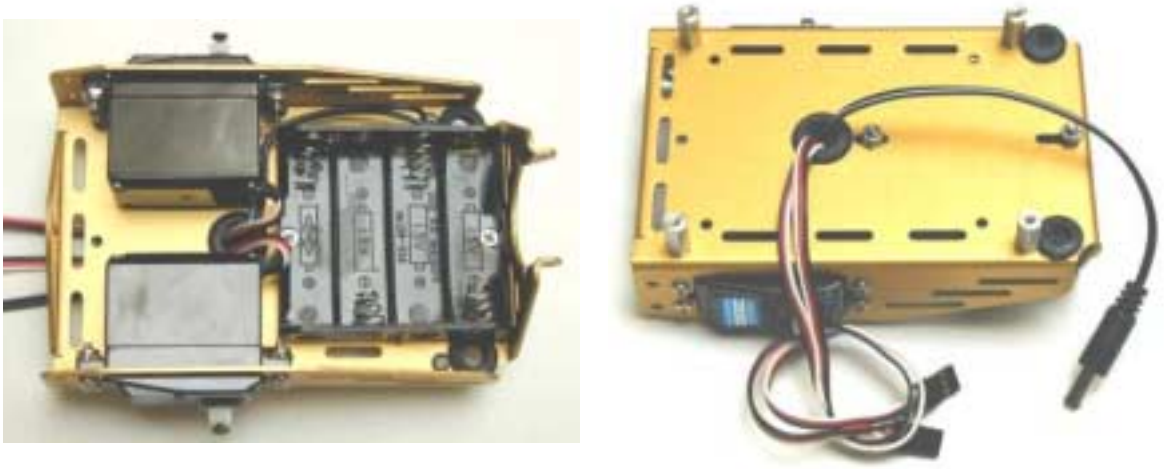
Lista de Materiales:

(1) Chasis de Boe-Bot parcialmente armado.
(1) Porta pilas vacío
(2) Tornillos cabeza plana 4-40
(2) Tuercas 4-40

Figura 1.26: Porta pilas y tornillos para su montaje.

Montaje:

La Figura 1.27 muestra el chasis del Boe-Bot con el porta pilas montado visto (a) desde abajo y (b) desde arriba.



(a) (b)

Figura 1.27: (a) Porta pilas instalado, (b) forma de pasar los cables.

- Use los tornillos cabeza plana y las tuercas para colocar el porta pilas en la parte inferior del chasis del Boe-Bot como se muestra en la Figura 1.27 (a). Asegúrese de insertar los tornillos en el porta pilas y luego colocar y apretar las tuercas en la cara superior del chasis.
- Pase el cable del porta pilas a través de la goma pasa cables más grande, ubicada en el centro del chasis.
- Pase los cables de los servos por el mismo agujero.
- Acomode los cables de los servos y de alimentación como se muestra en la Figura 1.27 (b). El cable de alimentación debe salir hacia el frente del Boe-Bot entre los separadores y los cables de los servos entre los separadores del costado izquierdo del chasis.

Montaje de la Plaqueta de Educación al chasis del Boe-Bot

La Figura 1.28 muestra la Plaqueta de Educación, BASIC Stamp y hardware de montaje.

Lista de Materiales:

(1) Plaqueta de Educación con BASIC Stamp 2
(4) tornillos 1/4" 4-40

Figura 1.28: BOE con BASIC Stamp y tornillos de montaje.

Montaje:

La Figura 1.29 muestra la Plaqueta de Educación montada sobre el chasis del Boe-Bot con los servos conectados en los puertos correspondientes.

- Asegúrese de que la protoboard blanca de la Plaqueta de Educación quede hacia el frente del Boe-Bot como se muestra en la figura.
- Use los cuatro tornillos 1/4" para sujetar la Plaqueta de Educación a los separadores.

Para saber cuál es el servo derecho y cual el izquierdo, mire la Figura 1.29. El servo derecho del Boe-Bot es el que se muestra y el izquierdo está del otro lado del chasis (no se muestra).

Figura 1.29: BOE colocada en el chasis.

- Conecte el servo derecho del Boe-Bot en el puerto 12 y el izquierdo en el puerto 13.

Las Ruedas

La Figura 1.30 muestra el hardware para colocar las ruedas.

Lista de Materiales:

(1) Boe-Bot parcialmente armado (no se muestra)
(1) Pasador (chaveta) 1/16"
(2) Cubiertas de goma
(1) Bolilla de polietileno de 1"
(2) Llantas de plástico
(2) Tornillos que originalmente sujetaban la palanca de mando del servo y fueron separados en la Actividad 3

Figura 1.30: Componentes para montar las ruedas.

Montaje:

La Figura 1.31 (a) muestra la rueda trasera sujetada al chasis del Boe-Bot con el pasador y la Figura 1.31 (b) muestra una rueda delantera colocada en el engranaje de salida del servo.

- La bolilla plástica del Boe-Bot es usada como rueda trasera o de arrastre y la chaveta es su eje. Para evitar que la chaveta que sujeta la bolilla al chasis del Boe-Bot se salga, separe sus puntas una vez colocada, como se muestra en la Figura 1.31 (a).
- Coloque las cubiertas de goma en las llantas de plástico.
- Cada rueda tiene un rebaje que coincide con el engranaje de salida de los servos. Una vez que la rueda y el engranaje estén alineados, presione firmemente hasta que el engranaje se introduzca en la rueda.
- Use los tornillos que obtuvo en la Actividad 3, para sujetar las ruedas en su lugar.

(a) (b)

Figura 1.31: (a), Rueda trasera montada en el chasis del Boe-Bot y (b), rueda delantera montada en el engranaje del servo.

Actividad 6: Navegación y Más Ajustes del Servo por Software

Ahora, es tiempo de sacar a pasear al Boe-Bot. Esta actividad le enseñará a programar al Boe-Bot para que se mueva hacia adelante, atrás y gire. Se realizará un ajuste fino del programa para asegurarse que el Boe-Bot pueda moverse en línea recta. Esta será la última iteración de la prueba del software y ajuste, de este capítulo.

Conectando Todo Nuevamente

La Figura 1.32 muestra un Boe-Bot armado. El porta pilas y el cable serial se reconectan a la BOE.

Lista de Materiales:

(1) Boe-Bot completo
(4) Pilas AA
(1) PC con el Stamp Editor y cable serial

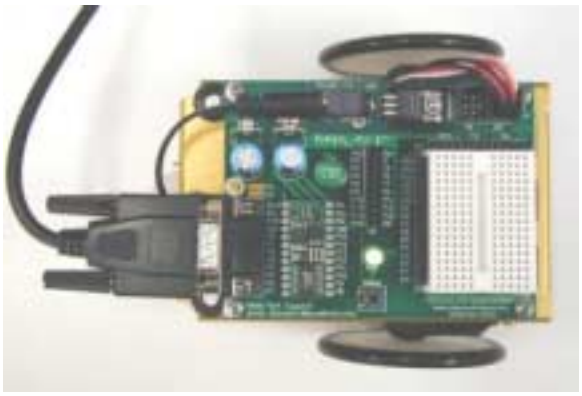


Figura 1.32: Boe-Bot listo para ser programado.

Armado:

- Coloque las pilas en el porta pilas, controlando su polaridad.
- Conecte la ficha del porta pilas en la BOE.
- Conecte el cable serial a la Plaqueta de Educación.

Los servos no necesariamente funcionarán parejos. Como resultado, aún cuando el Boe-Bot es programado para avanzar en línea recta, podría girar lentamente hacia la izquierda o la derecha. La calibración por Software es usada para solucionar este problema.

Programando el Boe-Bot para que se Mueva

La Figura 1.33 muestra las referencias para izquierda, derecha, adelante y atrás del BOE-Bot, que serán empleadas en todo el libro.

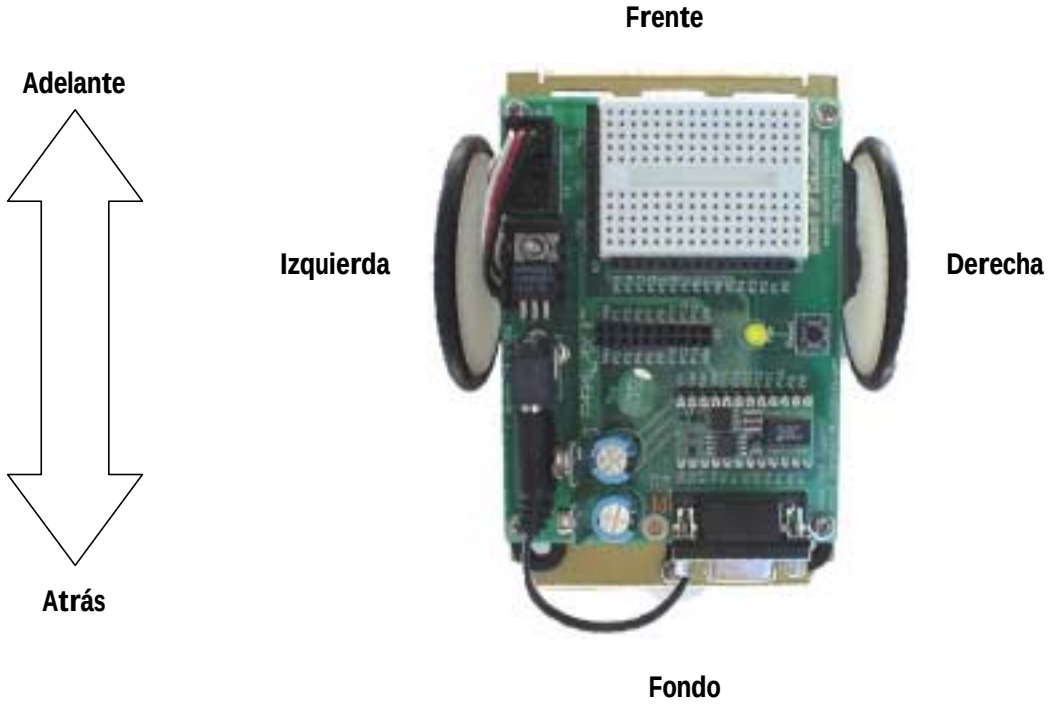


Figura 1.33: Boe-Bot desde el asiento del conductor.

Figura 1.34 muestra un esquema del Boe-Bot con los servos conectados en los puertos para servos 12 y 13. El servo del costado derecho del Boe-Bot está conectado a P12 y el de la izquierda a P13.

Hacer funcionar al Boe-Bot es tan fácil como agregar un segundo comando **`pulsout`**. La parte difícil es determinar los parámetros de duración de cada pulso.

Mire el costado derecho del Boe-Bot. Para hacer que esta rueda gire hacia adelante, el servo debe girar en sentido horario. Esto implica un argumento de duración menor que el central. Una duración de 650 servirá para lograr la velocidad máxima hacia adelante. Ahora mire el costado izquierdo del Boe-Bot. Para hacer girar esta rueda del Boe-Bot hacia adelante, el servo debe girar en sentido antihorario. En lugar de 650, es necesaria una duración de 850.

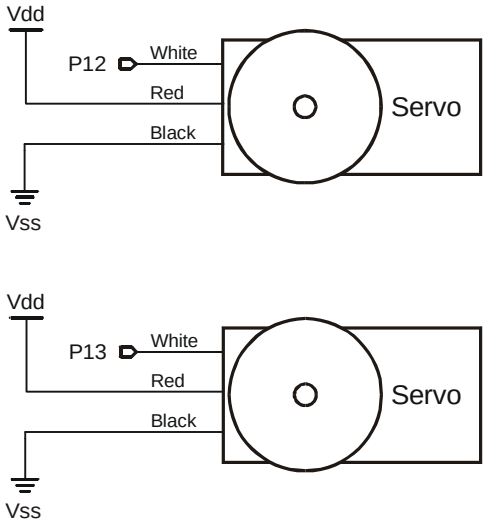


Figura 1.34: Esquema de conexión de los servos.

- Ingrese y ejecute el Programa 1.5.

```
' ¡Robótica! v1.5, Programa 1.5: Movimiento hacia adelante.

low 12                          ' Ajusta a P12 como salida baja.
low 13                          ' Ajusta a P13 como salida baja.

bucle:                          ' Etiqueta hacia donde saltar

  pulsout 12, 650               ' Envía pulsos de 1.3 ms por P12
  pulsout 13, 850               ' y de 1.7 ms por P13

pause 20                        ' cada 20 ms.

goto bucle                      ' Salta hacia la etiqueta “bucle:”.
```

Si el Boe-Bot se movió hacia atrás en lugar de hacia adelante, las líneas de los servos están invertidas. Significa que el servo que debería estar conectado al puerto para servos 12, fue conectado al puerto 13 y viceversa.

Cómo genera el Programa 1.5 los pulsos para los dos servos

Se han agregado dos líneas de programa, una para configurar a P13 como salida baja y una segunda para enviar pulsos por P13. Esto es todo lo que se necesita para controlar un segundo servo con un BASIC Stamp. El servo derecho, que está conectado a P12, recibe un pulso con un argumento de duración de 650 (1.3 ms) para hacerlo girar en sentido horario. Mientras tanto el servo izquierdo, que está conectado a P13, recibe un pulso con un argumento de duración de 850 (1.7 ms) para girar en sentido antihorario.

Su Turno

Guarde el Programa 1.5 con otro nombre y modifique los argumentos de duración de los comandos **`pulsout`**. Intercambiando los argumentos de duración el Boe-Bot se moverá hacia atrás. Colocando los argumentos de duración para el valor central que determinó en la Actividad 4, el Boe-Bot permanecerá estático. Colocando ambos argumentos en 650 el Boe-Bot rotará en su lugar en sentido antihorario, mientras que con ambos en 850, lo hará en sentido horario.

En algunos casos el Boe-Bot no se moverá hacia adelante en una línea recta. En ese caso, probablemente tampoco se moverá en línea recta hacia atrás. Esta vez, en lugar de ajustar la duración del pulso central, lo haremos con los valores de duración del pulso que hagan que el Boe-Bot se desplace en línea recta.

Por ejemplo, si el Boe-Bot se desvía hacia la derecha cuando se programa para ir hacia adelante en línea recta, hay que disminuir la velocidad de la rueda izquierda, o aumentar la de la derecha. Dado que los servos giran casi a su velocidad máxima, la mejor solución es disminuir la del izquierdo. Reduzca la duración del pulso del servo izquierdo (P13). En lugar de usar el comando **`pulsout 13, 850`**, podría intentar con **`pulsout 13, 840`**. esto reducirá la velocidad de la rueda izquierda. Probando con diferentes valores (iterativamente), encontrará finalmente los que hagan girar las ruedas del Boe-Bot a la misma velocidad. Después de algunos intentos, su Boe-Bot se moverá en línea recta.

- Tome notas de las duraciones de **`pulsout`** que obtuvo para el movimiento en línea recta. Puede necesitar estos valores en las próximas actividades.

- Agregue los argumentos de duración de pulso para los movimientos en línea recta en la etiqueta de cada servo, que ya tiene el valor de duración central.

Sumario y Aplicaciones

¡Felicitaciones por la construcción de su Boe-Bot! A lo largo de los procedimientos de este capítulo, ha tenido su primer encuentro con la prueba y ajuste de sistemas y subsistemas. Muchos temas esenciales que se trataron serán usados nuevamente en este texto. Por ejemplo, la Debug Terminal será su mejor y más usada herramienta en la depuración de errores de circuitos, así como también de programas.

Se enseñó a usar el lenguaje de programación PBASIC con algunos programas de ejemplo, para poder activar la Debug Terminal y el Boe-Bot. Se presentaron los comentarios y etiquetas de PBASIC, así como también los comandos **`low`**, **`pulsout`**, **`pause`** y **`goto`**. También se demostró la calibración por software.

Ejemplo del Mundo Real

Desde el trasbordador espacial hasta el Boe-Bot, aislar y probar los subsistemas durante cada fase del desarrollo es esencial, para asegurarse de que todo el sistema funcione al juntar las partes que lo componen. Más importante aún, aislar y probar cada subsistema minimiza el tiempo (y el nivel de dificultad) de la resolución de problemas. Al principio del capítulo, se discutieron los problemas asociados con el desarrollo y pruebas no iterativas. Imagine que nadie controle los subsistemas del Trasbordador Espacial antes de ensamblarlo. ¡Podría tomarle cientos de años a la NASA encontrar o resolver un problema!

Sin importar si se trata de competencias de robots, desarrollo de productos, o programas espaciales, las pruebas y desarrollos a niveles de sistemas y subsistemas son el mejor camino para evitar demoras innecesarias, cuando se trabaja en un proyecto desde el principio hasta el final. Especialmente en el desarrollo de productos, cada ingeniero del grupo desarrolla por su cuenta sistemas y subsistemas. A menudo, no es hasta la etapa final que se realiza la prueba a nivel de sistema y la integración del sistema. Algunas veces, todo lo que sabe el grupo de diseño son los requerimientos de entrada y salida (I/O) para su módulo particular dentro del proyecto total. A pesar de eso, los grupos de diseño deben desarrollar, simular (cosa que no hemos hecho) y probar iterativamente los subsistemas del módulo de proyecto sobre el que están trabajando.

También hemos tratado la calibración por software. Este es un tema muy actual debido a que muchos fabricantes están intentando incorporar microcontroladores para que sus productos puedan comunicarse por Internet. Los programas de diagnóstico remoto pueden incrementar la capacidad de los dispositivos microcontrolados para auto calibrarse. Además, los técnicos podrían realizar calibraciones por software, diagnóstico y en algunos casos reparaciones, todo a distancia. Imagine que en el futuro su TV falla. Para arreglarlo presionaría un par de botones del control remoto, para que el microcontrolador se comunique por Internet al centro de reparación. Luego de realizar un diagnóstico, tal vez hasta una computadora podría arreglar la falla. Si hace falta una reparación mayor, el técnico iría a su casa con los repuestos necesarios.

Aplicaciones del Boe-Bot

Un ítem que investigará en la sección Preguntas y Proyectos es qué ocurre cuando se cambia el conexionado de los servos. ¿Cómo se maneja esto? Involucra más cambios de los que pueda imaginar. Por ejemplo, si desconecta un servo del puerto 12 y lo coloca en el puerto 15, para que el sistema funcione no basta con cambiar la foto que muestra en que puerto se conecta el servo. El esquema eléctrico, que es el método principal para comunicar la información de cableado, debe ser modificado, así como también los programas. En algún momento podría querer conectar más servos para agregar un brazo o pinza para el Boe-Bot. Aunque no se incluye el tema en este texto , Preguntas y Proyectos tiene ejercicios que lo entrenarán en la conexión de servos a distintos puertos.

En este capítulo aprendió a programar el Boe-Bot para que se mueva hacia adelante, atrás o rote en el lugar. Durante algunas de las pruebas, se descubrieron y corrigieron pequeñas variaciones en el rendimiento del servo, mediante el software. La calibración por software se realizó a tres velocidades: adelante a máxima velocidad, atrás a máxima velocidad y estático en su lugar. En la sección de proyectos tendrá oportunidad de llevar a fondo y generalizar la calibración por software para distintas velocidades. En los capítulos siguientes, escribirá funciones y tablas para ajustar los servos a cualquier velocidad.

Preguntas y Proyectos

Preguntas

1. Explique cómo funciona el sistema de desarrollo y prueba iterativo. ¿Cómo se usó con el Boe-Bot?

2. Explique las dos cosas que hace el comando **debug**.

3. ¿Cuáles son los diferentes tipos de datos de salida que pueden ser usados con el comando **debug**? Explique cada tipo.

4. Explique como controlar y diagnosticar un servo asumiendo que no responda a ninguna de las pruebas de la Actividad 2. Además, explique como puede ser usado el comando **debug** para realizar pruebas adicionales que permitan aislar el problema.

5. Asumiendo que el motor de CC verifica lo que está haciendo cada 20 ms, ¿cuántas veces por segundo lo hace? Pista: Este es un problema de división.

6. Comente cómo modificaría la Figura 1.12 y el Programa 1.3 si quisiera realizar la calibración del servo en la línea de entrada/salida (I/O) P15. ¿Cómo se modificaría la Figura 1.14 (b)?

7. Suponga que en la Actividad 6 quiere usar las líneas de I/O 14 y 15 para conectar los servos. Realice un informe con las modificaciones necesarias a los diagramas, imágenes y programas de esta actividad, para que se ajuste a este cambio.

8. Cómo haría para conectar los servos a las líneas de I/O P10 y P11. Pista: Esto será muy fácil para los que usan la BOE Rev A. Repita las actividades de la Pregunta 7.

Ejercicios

1. ¿Qué sucedería si se usa **`pause 30`** en lugar de **`pause 20`** en los programas del Capítulo 1? ¿Cómo se vería afectado el funcionamiento del servo? Dibuje un diagrama similar a la Figura 1.16 basándose en un tren de pulsos que use **`pause 30`** y **`pulsout 12, 750`**.

2. La sección Cómo Trabaja el Programa de la Actividad 2, muestra cómo un **`pulsout`** con un argumento de duración de 750 genera pulsos de 1.5 ms. Repita este ejercicio para **`pulsout 12, 650`** y **`pulsout 12, 850`**.

3. ¿Cuál es el valor del argumento de duración de **`pulsout`** para obtener un pulso de 1.626 ms?

4. ¿Cuáles son los anchos de pulso máximo y mínimo que pueden ser generados con el comando **`pulsout`**?

5. Desafío: Dado que el tiempo en estado bajo del tren de pulsos no es relevante para la señal del servo, no se ajustó el tiempo de la pausa cuando se agregó un segundo comando **`pulsout`**. Sin embargo, cada comando **`pulsout`** causa un retardo adicional a la pausa de 20 ms. ¿Cómo ajustaría el tiempo de la pausa en el Programa 1.5 para asegurarse de que se genere un pulso cada 20 ms?

Proyectos

1. En la Pregunta 7, modificó las imágenes y programas de la Actividad 6 para poder usar las líneas de I/O P14 y P15 para conectar los servos, en lugar de las líneas de I/O P12 y P13. Use su Boe-Bot para verificar estas modificaciones. Asegúrese de guardar sus programas con diferentes nombres, porque en el siguiente capítulo, volveremos a usar las líneas I/O 12 y 13.

   Registre el comportamiento de su Boe-Bot cuando ejecute por primera vez su programa. ¿Se comportó como se esperaba? Si no es así, realice modificaciones (registrando todos los datos) hasta que su Boe-Bot funcione correctamente. Puede haber muchas iteraciones en este proceso; asegúrese de registrar todos los pasos hasta lograr que el Boe-Bot funcione como se esperaba.

2. Programe el Boe-Bot para que se mueva con varios patrones diferentes. Intente lo siguiente:

   (a) Identifique un par de valores de **`pulsout`** que hagan que el Boe-Bot se mueva hacia adelante lentamente. Busque velocidades de giro de las ruedas de 4 RPM.

(b) Identifique un par de valores de **`pulsout`** que hagan que el Boe-Bot se mueva lentamente en línea recta hacia atrás, a la misma velocidad anterior. ¿Cómo se comparan (o no) estos valores con los obtenidos en Proyecto 2 (a)?

3. Haga un gráfico de la velocidad de las ruedas en función del ancho del pulso para cada servo. Use varios anchos de pulso entre 1 y 2 ms (valores de **`pulsout`** entre 500 y 1000). Cuente cuantas revoluciones completa la rueda en un tiempo específico (20 segundos o un minuto), o vea cuánto tiempo demora en completar 10 revoluciones. Su gráfico debería verse así:

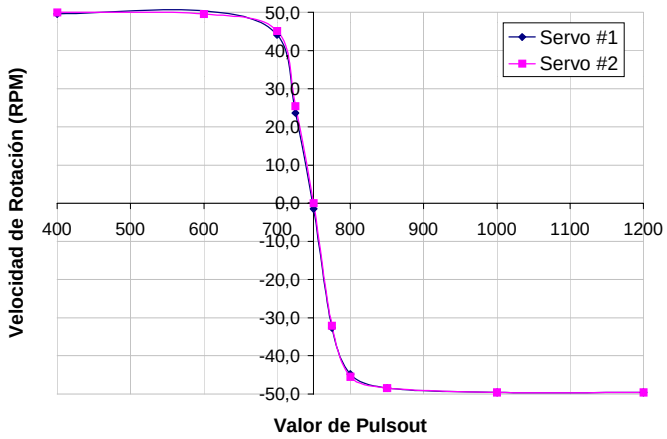


Figura 1.35: Valor de pulsout vs. Velocidad angular de los servos del Boe-Bot.

4. La Figura 1.35 muestra un gráfico generado por una hoja de cálculo de Microsoft Excel usando ocho puntos de datos. En su propio gráfico, sería más conveniente realizar muchas más mediciones, especialmente en la parte curva del gráfico. Use su programa de hoja de cálculo para magnificar el área entre 600 y 900.

   Use este gráfico para predecir los anchos de pulso requeridos para hacer que su Boe-Bot se mueva en línea recta, sin necesidad de prueba y error. Pista: A la velocidad angular de un servo le corresponde un cierto valor de **`pulsout`**. Luego busque la velocidad angular negativa (con el mismo valor absoluto) para el otro servo, seleccionando la duración apropiada de **`pulsout`**.

   Pruebe la precisión de las predicciones determinadas gráficamente, usando esos valores en un programa en su Boe-Bot, verificando si se mueve linealmente.

Capítulo 2: Programación de Movimientos del Boe-Bot

Capítulo 2: Navegación del Boe-Bot bajo el Control del Programa

Todo el Capítulo 2 se dedica a la programación de movimientos del Boe-Bot. Escribirá programas para que el Boe-Bot realice distintas maniobras. Algunos programas pueden ser usados para moverse por lugares pequeños, otros para realizar figuras. Cualquiera sea la maniobra, este capítulo presenta las herramientas necesarias para programar al Boe-Bot. Esto es lo que aprenderá a hacer en el Capítulo 2:

- Construir un indicador de batería baja.
- Programar su Boe-Bot para que se mueva en distintas direcciones, todas en el mismo programa.
- Escribir programas que ajusten la precisión de las maniobras del Boe-Bot.
- Escribir programas que almacenen listas largas de instrucciones de movimiento.
- Escribir programas que hagan que el Boe-Bot acelere y desacelere durante las maniobras.

Además, el Capítulo 2 introduce varias herramientas de programación en PBASIC, tales como bucles **`for...next`** e instrucciones **`if...then`**. Los ejercicios de este capítulo también ofrecen mucha práctica en el uso de variables y control de flujo para distintas tareas. También se introducirán conceptos matemáticos para convertir comandos de programa en distancia y velocidad. Para algunos, este será su primer encuentro con la física (dinámica) elemental.

Convirtiendo Instrucciones en Movimiento

En el capítulo anterior, programó al Boe-Bot para que se mueva hacia adelante, atrás y gire en el lugar. Además, se determinaron los ajustes de software para que el Boe-Bot se mueva en línea recta hacia adelante y atrás y para que quede estático.

Cada programa de la Actividad 6 del Capítulo 1 se enfocó en una dirección. Si el Boe-Bot se programaba para ir hacia adelante, debía ser reprogramado para que fuera hacia atrás y reprogramado nuevamente para que gire en el lugar. En este capítulo, se incorporarán todas las direcciones en un único programa. Midiendo cuantos pulsos son necesarios para que el Boe-Bot rote un cierto ángulo durante un giro, puede realizar programas para efectuar maniobras más precisas. Por ejemplo, el Boe-Bot puede ser programado para realizar cuadrados, sinusoides o triángulos.

Las maniobras preprogramadas son útiles, pero cuando se trata de largas listas de maniobras, estos programas se volverán complicados. El PBASIC tiene un método simple y eficiente para grabar y leer listas de direcciones en la memoria del programa. Observará que mientras el Boe-Bot realiza sus maniobras, ejecuta

cambios bruscos de dirección. Para solucionar este problema se pueden agregar comandos al programa para que el Boe-Bot acelere y desacelere en los cambios de dirección. Esto prolongará la vida útil de los servos del Boe-Bot.

El comportamiento del Boe-Bot cuando las baterías están bajas puede ser bastante extraño. Las baterías bajas no son buenas para los servos ni para el BASIC Stamp. En la primera actividad realizará la construcción y prueba de un indicador de batería baja.

Actividad 1: Indicador de Batería Baja

La condición de un circuito cuya fuente de alimentación cae por debajo del nivel necesario para su funcionamiento apropiado, es conocida como brownout. El BASIC Stamp se auto protege del brownout apagando los chips de memoria y procesador, hasta que la tensión de la fuente de energía regrese a los valores normales. Un valor inferior a 5.2 V en Vin genera un valor menor de 4.3 V a la salida del regulador de tensión interno del BASIC Stamp. Un circuito llamado detector de brownout en el BASIC Stamp controla esta condición. Cuando la alimentación se normaliza, el BASIC Stamp es reiniciado (reset). En respuesta a un reset, el BASIC Stamp se comporta igual que cuando se conecta la alimentación por primera vez, comenzando a ejecutar su programa desde el principio.

Una forma de indicar un reset es incluir una señal inconfundible al principio de todos los programas del Boe-Bot. La señal se produce cada vez que la alimentación es conectada, o cada vez que se produzca un reset debido a un brownout. La señal más efectiva es un parlante que emita un tono cada vez que el programa del BASIC Stamp se ejecute desde el principio o se reinicie. El símbolo esquemático y la vista superior del parlante piezoeléctrico que serán usados en esta actividad se muestran en la Figura 2.1.

Lista de Componentes

(1) Boe-Bot armado y probado
(1) Parlante piezoeléctrico
(varios) Cables

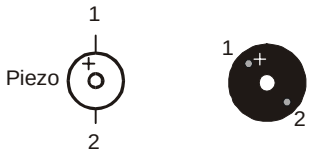


Figura 2.1: Piezoeléctrico

¡Constrúyalo!

La Figura 2.2 el esquema y el diagrama de conexionado de los servos y el parlante piezoeléctrico.

- Construya el circuito que se muestra en la Figura 2.2 (b).

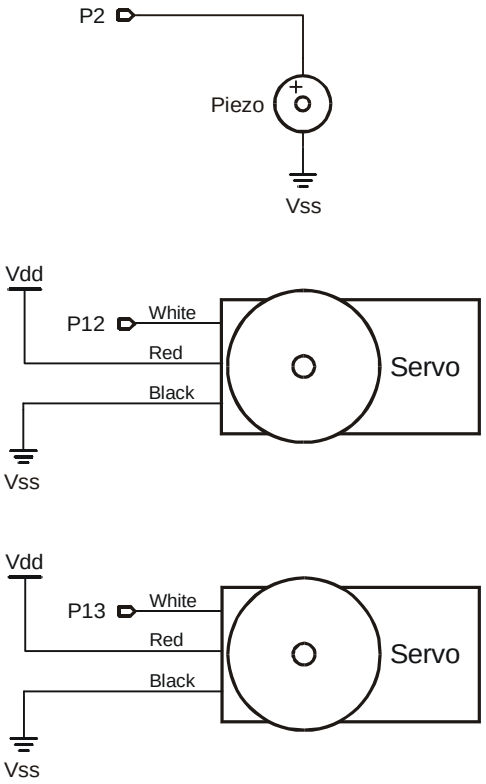
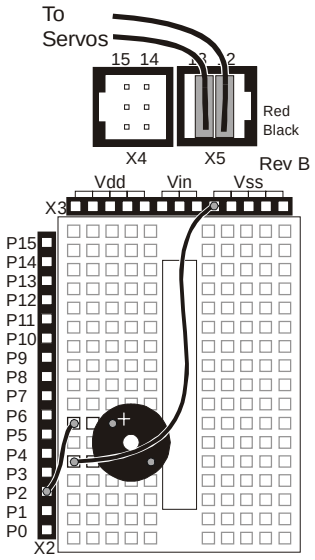


Figura 2.2 (a) Esquema



(b) Diagrama de conexionado

✓
TIPS

¿Primera vez que arma circuitos guiándose por esquemas? Lea el Apéndice F: Reglas para armado de circuitos en Protoboard.

Las instrucciones para todas las actividades a partir de este punto asumirán que los servos se encuentran conectados como se muestra en las Figuras 2.2 (a) y (b).

En el Capítulo 1, se introdujeron las unidades de milisegundos y microsegundos. En este capítulo, nos ocuparemos de los hertz y kilohertz. Un hertz indica una frecuencia de una vez por segundo y se abrevia 1 Hz. Un kilohertz representa mil veces por segundo y se abrevia 1 kHz.

Hz y kHz

$$1\,Hz = \frac{1}{s} = 1\ s^{-1}$$

$$1\,kHz = \frac{1000}{s} = 1000\ s^{-1}$$

Programando el Indicador de Batería Baja

Al principio del programa, un comando que emita un tono por el parlante constituirá el indicador de batería baja. Luego de ejecutar este comando, el programa entrará en un bucle infinito, hasta que ocurra un reset. Si el circuito está correctamente armado y el programa funciona correctamente, el tono debería escucharse cada vez que la alimentación es conectada. También debería sonar cada vez que se presione el botón reset. Esto garantizará que se emita el sonido cuando ocurra un reset causado por baterías bajas.

- Conecte el porta pilas a la BOE.
- Conecte el cable serial de la computadora al conector DB9 de la BOE.
- Ingrese el Programa 2.1 en el Stamp Editor
- Ejecútelo haciendo clic en Run y seleccionando Run.
- Este programa usa la Debug Terminal, así que debe dejar conectado el cable serial a la BOE mientras el Programa 2.1 se esté ejecutando.

```
' ¡Robótica! v1.5, Programa 2.1, Indicador de batería baja.

debug cls, "Beep!!!"                    ' Muestra cuando suena el parlante.
output 2
freqout 2, 2000, 3000                   ' Envía una señal de 3 kHz durante 2 s.

bucle:                                  ' Etiqueta del bucle.
  debug "Esperando un reset…", cr       ' Muestra que el BS2 está esperando.
goto bucle:                             ' Salta a la etiqueta bucle (infinito).
```

- Verifique que el indicador de batería baja funciona presionando varias veces el botón de reset de la BOE.

Al igual que cuando se ejecutó por primera vez, el parlante debería tocar un tono agudo durante 2 s. Al mismo tiempo, la Debug Terminal debería mostrar el mensaje "Beep!!!". Luego, todo debería quedar en silencio mientras la Debug Terminal repite la línea de mensaje "Esperando un reset…".

- Si el parlante piezoeléctrico no emite ningún sonido, controle el código y el conexionado y luego ejecute nuevamente el programa.

Los problemas con el uso de Debug normalmente son causados por errores en la escritura del programa.

- Si la pantalla Debug no se comporta como era de esperarse, controle su programa y asegúrese de que el código sea exactamente igual al mostrado en el Programa 2.1 anterior.

Cómo Funciona el Programa Indicador de Batería Baja

- Use el Apéndice B o el BASIC Stamp Manual para buscar la descripción de los comandos **`freqout`** y **`output`** que se usaron en el Programa 2.1.

El Programa 2.1 inicia mostrando el mensaje "Beep!!!". Luego, inmediatamente después de imprimir el mensaje, el comando **`freqout`** toca un tono de 3 kHz mediante el parlante piezoeléctrico durante 2 s. Son necesarios dos pasos para emitir el sonido. Primero, P2 debe ser configurado como salida con el comando **`output 2`**. Después se emite el sonido con la ejecución del comando **`freqout 2, 2000, 3000`**. Envía pulsos por P2 que hacen que el parlante piezoeléctrico vibre a 3 kHz por 2 s. Cuando el sonido termina, el programa ingresa en un bucle infinito, mostrando una y otra vez el mensaje "Esperando un reset…". Cada vez que el botón reset de la BOE es presionado o se conecta la alimentación, el programa comienza desde el principio.

Las líneas de código del programa indicador de batería que generan el sonido serán empleadas al inicio de cada programa, de aquí en adelante. Puede considerarse como parte de la "rutina de inicio" o "secuencia de encendido" para cada programa del Boe-Bot.

Su turno

- Copie el comando **`freqout`** del principio del Programa 2.1 al principio del Programa 1.5 del Capítulo 1, Actividad 6.

- Ejecute la versión modificada del Programa 1.5. El Boe-Bot debería permanecer estático y emitir un sonido durante dos segundos, antes de comenzar a moverse hacia adelante.

Actividad 2: Controlando la Distancia

Hasta ahora, los programas del Boe-Bot terminaban en bucles infinitos. Por ejemplo, el Programa 1.5 hizo que el Boe-Bot se moviera hacia adelante, pero eso es todo. El Boe-Bot simplemente seguía avanzando. Esta actividad introduce una técnica para controlar la distancia que recorre el Boe-Bot.

Programación con Control de Distancia

Un bucle infinito puede cumplir una tarea, pero no sabe cuándo detenerse. La mejor forma de resolver el problema es reemplazar el bucle infinito por otro llamado bucle **`for…next`**. Puede usar un bucle **`for...next`** para especificar la cantidad de veces que se ejecutarán los comandos que están en su interior.

Un bucle **`for…next`** usa una variable para contabilizar el número de repeticiones. Un bucle for…next usa la variable para almacenar un número que se incrementa cada vez que se repite el bucle. En PBASIC, una variable debe ser declarada antes de usarla. El Programa 2.2 muestra ejemplos de la declaración de la variable y un bucle **`for…next`**. El bucle **`for…next`** es usado para reemplazar el bucle infinito que se usó en el Programa 1.5. El bucle **`for...next`** hace que el Boe-Bot se mueva hacia delante y luego se detenga, controlando la cantidad de pulsos que se envían a los servos.

- Conecte el porta pilas a la BOE.
- Conecte el cable serial de la computadora al conector DB9 de la BOE.
- Ingrese el Programa 2.2 en el Stamp Editor.
- Ejecute el programa haciendo clic en Run y seleccionando Run en el Stamp Editor.
- Cuando el parlante del Boe-Bot comienza a emitir un sonido, indicando que el programa ha empezado, presione y mantenga presionado el botón Reset de la BOE.
- Desconecte el cable serial de la Boe-Bot.
- Coloque el Boe-Bot en la superficie sobre la cual desea que se mueva.
- Suelte el botón Reset.
- Observe el comportamiento del Boe-Bot.

FYI	Desde este punto en adelante, el procedimiento que se acaba de describir se repetirá con todos los programas.

```
' ¡Robótica! v1.5, Programa 2.2: Control de distancia

'-----Declaración-------

  cuenta_pulsos   var   word            ' Declara una variable para contar pulsos.

'-----Inicialización-----

  output 2                              ' Configura a P2 como salida.
  freqout 2, 2000, 3000                 ' Sonido que indica inicio/reinicio.
  low 12                                ' Configura a P12 y
  low 13                                ‘ P13 como salidas bajas.

'-----Rutina Principal-------

principal:

  adelante:                             ' Inicio de rutina.
    for cuenta_pulsos = 1 to 100        ' Bucle que envía 100 pulsos para adelante.
      pulsout 12, 650                   ' Pulso de 1.3 ms en el servo derecho.
      pulsout 13, 850                   ' Pulso de 1.7 ms en el servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next

  stop                                  ' Espera hasta el reset.
```

Cómo Funciona el Programa de Control de Distancia

- Busque los comandos **`for…next`** y **`stop`** en el Apéndice C: Referencia Rápida de PBASIC o en el BASIC Stamp Manual (en Inglés) antes de continuar.

Los programas de ejemplo en PBASIC de este libro estarán organizados por secciones. Tres de las cinco secciones comúnmente usadas en un programa son: declaraciones, inicializaciones y rutina principal. A medida que los programas en PBASIC se vuelvan más largos y complejos, usar comentarios para separar las secciones como **`'-----Declaración-------`** e '-----**Inicialización**----- hacen que el programa sea más fácil de leer. Las secciones y sus contenidos serán discutidos con más detalle después del Programa 2.6.

Una variable llamada **cuenta_pulsos** se declara con un tamaño de dos bytes (un word) de capacidad de almacenamiento, mediante el comando **cuenta_pulsos var word**. Una variable con tamaño word puede almacenar números entre 0 y 65535. Hay gran variedad de comandos PBASIC que pueden ser usados para cambiar el valor de una variable. Otros comandos PBASIC pueden usar el valor de una variable para tomar decisiones. Las variables también pueden ser usadas en lugar de números, como argumentos de ciertos comandos PBASIC. Verá ejemplos de todos estos usos de variables en este capítulo.

La Tabla 2.1 muestra las opciones de declaración de variables y el rango de números que pueden almacenar. El tamaño del número que una variable puede almacenar depende de cuántos bits contiene. Un simple bit, es una ubicación de memoria que puede almacenar un "1" o un "0." A mayor cantidad de bits mayor es el número binario que puede almacenar.

Tabla 2.1: Tamaños de Declaración de Variables

Declaración de Tamaño	Cantidad de Bits	Puede Almacenar Números de-hasta
bit	1	0 a 1
nib	4	0 a 15
byte	8	0 a 255
word	16	0 a 65535 (o –32768 a +32767)

Los cuatro comandos del sector de inicialización le deberían parecer bastante familiares a esta altura. Los primeros dos son los comandos **output** y **freqout** usados para emitir un sonido cuando el programa se comienza a ejecutar. También incluye los comandos **low** que son usados para fijar los valores iniciales de las líneas de E/S usadas para controlar los servos.

Recuerde:	Si el sonido se emite, sin razón aparente, cuando el Boe-Bot está a la mitad de una maniobra, indica una condición de brownout, causada por baterías bajas.

La rutina **principal** usa un bucle **for...next** seguido por un comando **stop**. El bucle **for...next** reemplaza el bucle infinito **goto** usado en los ejemplos anteriores. Los comandos interiores al bucle **for...next** le deberán parecer familiares. Son los comandos que envían los pulsos a los servos del Boe-Bot. Los valores del argumento ***duración*** usados en los comandos **pulsout,** son los valores para hacer que el Boe-Bot se mueva hacia adelante.

La primera vez que se ejecuta el bucle for next, el valor de **cuenta_pulsos** es fijado en "1." Luego se ejecutan los dos comandos **pulsout** y el comando **pause**. Cuando el programa llega a la instrucción **next**, salta al principio de la instrucción **for**. La segunda vez que se ejecuta el bucle, la instrucción **for** le suma uno al valor de la variable **cuenta_pulsos**. Ahora el valor de **cuenta_pulsos** es "2," y la instrucción **for** controla si **cuenta_pulsos** es igual a 100. Dado que **cuenta_pulsos** aún no llega a 100, el programa

continua ejecutando las líneas interiores del bucle hasta la instrucción next. Los tres comandos, los pulsos para controlar los servos y la pausa de 20 ms, se ejecutan nuevamente y la instrucción **next** vuelve a enviar el control del programa a la instrucción for. El valor de **cuenta_pulsos** es incrementado y comparado con el valor final nuevamente y así sucesivamente.

La 100ma vez que el programa llega a la instrucción **next** del bucle **for...next**, envía el programa hasta la instrucción **for** nuevamente. Esta vez, el valor de **cuenta_pulsos** es incrementado a 101. Ahora, cuando **cuenta_pulsos** es comparado con el argumento ***duración***, se encuentra que es mayor que el valor límite 100. Así que en lugar de ejecutar los comandos que envían otro pulso, la instrucción **for** envía el control del programa a la instrucción inmediatamente posterior a **next**.

El comando **stop** es ejecutado inmediatamente después del bucle **for...next** y hace que el programa se detenga. Lo único que puede reactivar un BASIC Stamp después de un comando **stop** es un reset (reinicio) por hardware. En otras palabras, si quiere ejecutar nuevamente el programa, presione el botón Reset (Rst) de la BOE.

Su turno

- Modifique la rutina principal del Programa 2.2 como se muestra abajo.

```
'-----Rutina Principal-------

principal:                              ' Rutina Principal.

  adelante:                             ' Rutina hacia adelante.
    for cuenta_pulsos = 1 to 60         ' Envía 60 pulsos hacia adelante.
      pulsout 12, 650                   ' Pulso de 1.3 ms del servo derecho.
      pulsout 13, 850                   ' Pulso de 1.7 ms del servo izquierdo.
      pause 20                          ' Pausa por 20 ms.
    next

    pause 500                           ' Pausa por 0.5 s.

  atras:
    for cuenta_pulsos = 1 to 60         ' Envía 60 pulsos hacia atrás.
      pulsout 12, 850                   ' Pulso de 1.7 ms del servo derecho.
      pulsout 13, 650                   ' Pulso de 1.3 ms del servo izquierdo.
      pause 20                          ' Pausa por 20 ms.
    next

    pause 500                           ' Pausa por 0.5 s.

  stop                                  ' Se detiene hasta un reset.
```

- Ejecute la versión modificada del Programa 2.2. Su Boe-Bot debería moverse hacia adelante, luego hacia atrás y luego detenerse.

Cuando programa el Boe-Bot, normalmente necesita que se mueva una distancia determinada o que gire un cierto ángulo. Es muy útil poder calcular la distancia que recorrerá o que ángulo girará el Boe-Bot con un comando específico. Es fácil llegar a una respuesta. Sabemos que la circunferencia es igual a pi (π) multiplicado por el diámetro de la rueda:

circunferencia = π x diámetro de la rueda

circunferencia = 3.14159 x 6.67 cm $\cong$ 21 cm

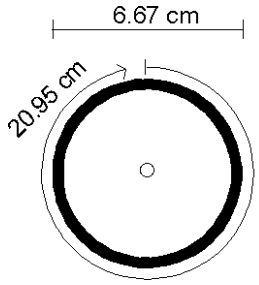


Figura 2.3: Diámetro y Circunferencia de la Rueda

Con un giro completo de las ruedas, el Boe-Bot se moverá aproximadamente 21 cm. Si le enviamos pulsos al servo durante la cantidad correcta de tiempo, el Boe-Bot se moverá una distancia predeterminada. Por ejemplo, con un comando **`pulsout`** de 850 el servo girará a aproximadamente50 revoluciones por minuto (RPM), o 0,83 revoluciones/seg. Así que la velocidad será aproximadamente:

21 cm/revolución x 0.83 revoluciones/sec = 17.5 cm/s

El tiempo que le lleva al Boe-Bot desplazarse 100 cm es:

t_{viaje} = 100 cm / 17.5 cm/s $\cong$ aprox. 5.7 segundos

Debido a que las duraciones de la pausa y los pulsos son conocidas, se puede calcular el tiempo de ejecución de cada bucle:

t_{bucle} = 1.3 ms + 1.7 ms + 20 ms = 23 ms

Calcular el número de bucles (el argumento ***`duración`*** del bucle **`for...next`**) se reduce a resolver el cociente entre t_{viaje} y t_{bucle}.

Número de bucles = 5.7 sec ÷ 23 ms/bucle

= 5.7 s ÷ 0.023 s/bucle

$\cong$ 247 bucles

- Modifique el Programa 2.2 para que realice 247 bucles hacia adelante (100 cm), luego ejecútelo.

¿Qué distancia recorrió el Boe-Bot? Varios factores pueden influir sobre la distancia que recorrió el robot, incluyendo diferencias entre los servos y tensión de las pilas. Sin embargo, 247 es un valor inicial razonable. El argumento de ***duración*** del bucle **`for...next`** puede ser ajustado con más precisión.

Actividad 3: Maniobras – Haciendo Giros

Si el mismo valor es sumado al ancho de pulso central de un servo y restado del ancho de pulso central del otro, el Boe-Bot se moverá en línea recta, hacia adelante o atrás. Cuando el servo derecho recibe una duración de **`pulsout`** de 650 (1.3 ms) y el servo izquierdo recibe una duración de **`pulsout`** de 850 (1.7 ms), el Boe-Bot se mueve hacia adelante. Cuando se intercambian las duraciones de pulsos de cada servo, el Boe-Bot va hacia atrás.

Si ambos servos reciben pulsos de 1.3 ms, girarán en la misma dirección, haciendo que el Boe-Bot gire en sentido antihorario. Si se aplican muchos pulsos, el Boe-Bot seguirá rotando. Si se envían unos 30 pulsos, el efecto neto será un giro a la izquierda de 90°. El mismo principio se aplica si ambos servos reciben pulsos de 1.7 ms, girando el Boe-Bot en sentido horario, en lugar de antihorario.

Programando Giros a la Derecha e Izquierda

El Programa 2.3 como modificar las rutinas **`adelante`** y **`atras`** del Programa 2.2 para lograr que el Boe-Bot realice giros.

- Ingrese y ejecute el Programa 2.3.

```
' ¡Robótica! v1.5, Programa 2.3: Girando en el lugar.

'-----Declaración-------

  cuenta_pulsos   var   word            ' Declara un contador tipo word.

'-----Inicialización-----

  output 2                              ' Configura a P2 como salida.
  freqout 2, 2000, 3000                 ' Emite un tono de 3 kHz por 2 s.
  low 12                                ' Fija a P12 y 13 como salidas en
  low 13                                ´ estado bajo.

'-----Rutina Principal-------
```

```
principal:                              ' Rutina principal

  giro_izq:                             ' Rutina de giro a la izquierda.
    for cuenta_pulsos = 1 to 30         ' Envía 30 pulsos.
      pulsout 12, 650                   ' Pulso de 1.3 ms al servo derecho.
      pulsout 13, 650                   ' Pulso de 1.3 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next

    pause 500                           ' Pausa de 0.5 s.

  giro_der:                             ' Rutina de giro a la derecha.
    for cuenta_pulsos = 1 to 30         ' Envía 30 pulsos.
      pulsout 12, 850                   ' Pulso de 1.7 ms al servo derecho.
      pulsout 13, 850                   ' Pulso de 1.7 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next

    pause 500                           ' Pausa de 0.5 ms.

  stop                                  ' Se detiene hasta un reset.
```

Cómo Funciona el Programa

Solamente se hicieron tres modificaciones al Programa 2.2 para hacer el Programa 2.3. Primero, las etiquetas **`adelante:`** y **`atras:`** se cambiaron por **`giro_izq:`** y **`giro_der:`** respectivamente. Luego, se fijaron los dos argumentos de duración de **`pulsout`** de la rutina **`giro_izq`** en 650. Los argumentos de **`pulsout`** de la rutina **`giro_der`** se fijaron en 850.

Su Turno

- Agregue las rutinas **`adelante`** y **`atras`** del Programa 2.2 al Programa 2.3. Para hacer que el Boe-Bot se mueva hacia adelante y atrás antes de girar a la izquierda y derecha, inserte las rutinas **`adelante`** y **`atras`** entre las etiquetas **`principal`** y **`giro_izq`**. Ejecute el programa y observe los resultados.

- En lugar de usar un argumento de ***`duración`*** de 30 para cada bucle de giro **`for...next`**, pruebe los valores 27, 28, 29, etc. hasta encontrar el valor más exacto que cause que el Boe-Bot realice un giro de 90º. Los valores pueden o no ser los mismos para las rutinas **`giro_izq`** y **`giro_der`**.

Actividad 4: Maniobras – Acelerando

La aceleración es una forma de aumentar gradualmente la velocidad de los servos, en lugar de ponerlos a funcionar bruscamente a velocidad máxima. Esta técnica aumentará la vida útil de las pilas y los servos de su Boe-Bot.

Programando Movimientos con Aceleración

La clave es usar variables y constantes para determinar la duración del pulso de los servos. Un bucle **`for…next`** incrementa una variable cada vez que se ejecutan las instrucciones entre **`for`** y **`next`**. Dado que su valor aumenta gradualmente, puede ser usado para incrementar gradualmente el ancho del pulso. El Programa 2.4 muestra como puede ser usada esta técnica para hacer que el Boe-Bot acelere y desacelere su velocidad en la rutina adelante.

```
' ¡Robótica! v1.5, Programa 2.4: Aceleración y Desaceleración.
'-----Declaración-------

  cuenta_pulsos  var   word             ' Contador de bucle for...next.
  ancho_der  var   word                 ' Almacena el ancho del pulso derecho.
  ancho_izq   var   word                ' Almacena el ancho del pulso izquierdo.

'-----Inicialización-----

  output 2                              ' Configura a P2 como salida.
  freqout 2, 2000, 3000                 ' Sonido indicador de reset.
  low 12                                ' Fija P12 y 13 salidas en nivel bajo.
  low 13

'-----Rutina Principal-------

principal:                              ' Rutina principal.
  acelera_adelante:                     ' Rutina que acelera hacia adelante.
    for cuenta_pulsos = 0 to 100 step 2 ' El bucle cuenta en pasos de 2.
      pulsout 12, 750 - cuenta_pulsos   ' Pulso de 1.5 ms – cuenta_pulsos.
      pulsout 13, 750 + cuenta_pulsos   ' Pulso de 1.5 ms + cuenta_pulsos.
      pause 20                          ' Pausa de 20 ms.
    next

  adelante:                             ' Rutina de avance.
    for cuenta_pulsos = 1 to 100        ' Bucle que envía 100 pulsos adelante.
      pulsout 12, 650                   ' Pulso de 1.3 ms al servo derecho.
      pulsout 13, 850                   ' Pulso de 1.7 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next
```

```
desacelera_adelante:                  ' Rutina que desacelera el avance.
  for cuenta_pulsos = 100 to 0 step 2 ' Bucle que cuenta en pasos de 2.
    pulsout 12, 750 - cuenta_pulsos   ' Pulso de 1.5 ms – cuenta_pulsos.
    pulsout 13,750 + cuenta_pulsos    ' Pulso de 1.5 ms + cuenta_pulsos.
    pause 20                          ' Pausa de 20 ms.
  next

stop                                  ' Se detiene hasta un reset.
```

Cómo Funciona el Programa

Los pulsos en la rutina **acelera_adelante** están dentro de un bucle **for…next**. Observe el argumento ***step 2*** agregado a la instrucción **for**. Ninguno de los ejemplos anteriores del bucle **for...next** usaron este argumento, así que el valor de **cuenta_pulsos** era incrementado en pasos de una unidad. El argumento **step 2** de la instrucción **for** hace que el valor de **cuenta_pulsos** se incremente en pasos de a 2.

```
acelera_adelante:
  for cuenta_pulsos = 0 to 100 step 2
    pulsout 12, 750 - cuenta_pulsos
    pulsout 13, 750 + cuenta_pulsos
    pause 20
  next
```

En el primer paso por el bucle **for...next**, el valor de **cuenta_pulsos** es "0." La segunda vez, **cuenta_pulsos** es incrementado a "2," y la tercera a "4," y así hasta 100. La clave de la aceleración es modificar ligeramente el ancho del pulso cada vez que es enviado al servo pulse, hasta llegar al valor deseado. Incorporando el valor de of **cuenta_pulsos** en el argumento ***duración*** de cada comando **pulsout** se puede lograr variar el ancho del pulso en forma gradual.

Miremos los pulsos enviados al servo derecho. El comando para enviar el pulso es **pulsout 12, 750 – cuenta_pulsos**. Cada pasada por el bucle **for...next**, el valor de **cuenta_pulsos** se incrementa en dos unidades. Esto significa que cada vez que se ejecuta el bucle, el valor del argumento de ***duración*** del comando **pulsout** se reduce en dos unidades, dado que **cuenta_pulsos,** se resta a 750. En las últimas pasadas por el bucle, el valor de **cuenta_pulsos** se acerca a 100, valor que se resta a 750 para obtener como resultado 650. Para el servo izquierdo, el valor de **cuenta_pulsos** se suma a 750 y la última vez que se ejecuta el bucle la duración es 850. Cuando se alcanzan estos valores, los servos se mueven a toda velocidad hacia adelante y el programa puede ejecutar la rutina **adelante,** sin cambios de velocidad.

Cuando finaliza la rutina **adelante**, el desafío es variar lentamente los anchos de los pulsos hacia el valor de reposo 750. Una característica útil del PBASIC es que los bucles **for...next** pueden realizar cuentas descendentes, si el argumento de ***inicio*** es mayor que el argumento ***final***. Este método fue usado para realizar la cuenta regresiva en la rutina **desacelera_adelante**. El bucle **for…next** comenzó con un valor de **cuenta_pulsos** de 100, realizando la cuenta regresiva en pasos de 2 hasta 0. Esto es debido a que se usó el comando **for cuenta_pulsos = 100 to 0 step 2**.

Su Turno

- Desarrolle rutinas que aceleren y desaceleren, para los programas en la Actividad 2 y 3, rutinas **atras**, **giro_der** y **giro_izq**.

Será necesario algún ajuste fino para lograr que el Boe-Bot realice giros de 90° y 180°. Ajustando los argumentos ***paso***, ***inicio*** y ***final*** de la instrucción **for,** se podrán realizar giros más precisos. Para giros de 90°, es posible acelerar y desacelerar sin llegar nunca a la máxima velocidad (ancho de pulso).

Actividad 5: Recordando Listas Largas Usando la EEPROM

El BASIC Stamp almacena una versión tokenizada del programa PBASIC en su memoria de sólo lectura, programable y borrable eléctricamente (EEPROM). Físicamente, la EEPROM es el pequeño chip negro del módulo BASIC Stamp II rotulado "24LC16B." Este componente es fabricado por Microchip Inc. La EEPROM del BASIC Stamp puede almacenar 2048 bytes (2 kB) de información. Lo que no se usa para almacenar el programa (que comienza desde la dirección 2047 hacia la dirección 0) puede ser usado para almacenar datos (comenzando desde la dirección 0 hacia la dirección 2047).

FYI	Si los datos se superponen con su programa, éste no se ejecutará apropiadamente.

La memoria EEPROM es diferente de la de almacenamiento de variables (RAM), en varios aspectos:

- La EEPROM necesita más tiempo para almacenar datos, algunas veces hasta varios milisegundos.
- La EEPROM solamente tolera un número limitado de ciclos de escritura, de aproximadamente 10 millones de ciclos. La RAM tiene capacidades ilimitadas de lectura/escritura.
- La función primaria de la EEPROM es almacenar programas; los datos se almacenan en el espacio sobrante.

Puede ver el contenido de la memoria EEPROM del BASIC Stamp haciendo clic en Run y seleccionando Memory Map. La Figura 2.4 muestra el mapa de memoria del Programa 2.3. Observe el Condensed EEPROM Map

(Mapa de EEPROM Condensado) en la parte inferior central de la figura. La franja horizontal angosta de color rojo representa la pequeña porción de EEPROM que ocupa el Programa 2.3.

Este programa puede haberle parecido grande mientras lo escribía, pero sólo ocupa 128 de los 2048 bytes de memoria de programa disponibles. Hay suficiente lugar para una larga lista de instrucciones. Una de las aproximaciones más simples es almacenar caracteres que indiquen en que sentido ir. Dado que los caracteres ocupan un byte de memoria, hay lugar suficiente para 1920 instrucciones de dirección de un caracter.

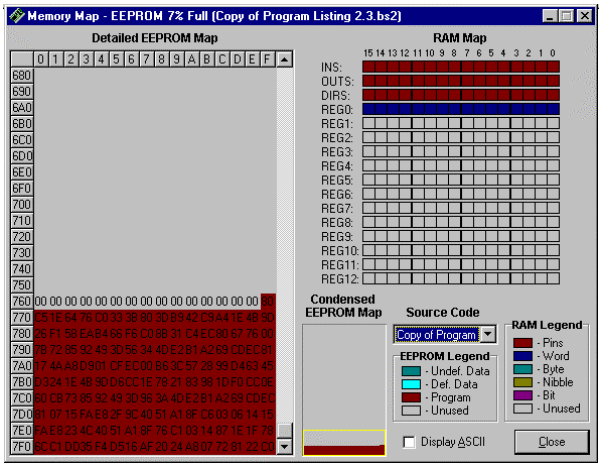


Figura 2.4: Mapa de memoria del BASIC Stamp.

Navegación por EEPROM

La directiva **`data`** es colocada normalmente en la sección **`declaración`** de un programa en PBASIC y es la mejor forma de almacenar listas largas. El Programa 2.5 muestra un ejemplo de cómo la directiva **`data`** puede ser usada para almacenar instrucciones de navegación para el Boe-Bot.

```
' ¡Robótica! v1.5, Programa 2.5: Navegación por EEPROM

'----- Declaración --------------       ' Rótulo para la rutina de declaración.

  cuenta_pulsos     var   word          ' Declara la variable cuenta_pulsos.
  direc_EE          var   byte          ' Almacena e inc. la dirección EEPROM.
  instruccion       var   byte          ' Almacena el caracter recuperado de la EE.

  data         "FFFBBLFFRFFQ"           ' Lista de instrucciones para el Boe-Bot.

'----- Inicialización ------------

  output 2                             ' Fija como salida la línea del parlante.
  freqout 2, 2000, 3000                ' Emite tono indicador de batería baja.
  low 12                               ' Configura P12 como salida baja.
  low 13                               ' Configura P13 como salida baja.

'----- Rutina Principal --------------

principal:                             ' Etiqueta de la rutina principal.
```

```
  read direc_EE, instruccion     ' Lee la direc_EE y almacena la instrucción
  direc_EE = direc_EE + 1        ' Incrementa direc_EE para la siguiente lectura

  if instruccion = "F" then adelante    ' Busca instrucción adelante.
  if instruccion = "B" then atras       ' Busca instrucción atras.
  if instruccion = "R" then giro_der    ' Busca instrucción giro_der.
  if instruccion = "L" then giro_izq    ' Busca instrucción giro_izq.

  stop                                  ' Detiene el programa hasta un reset

'----- Rutinas de Navegación -------

  adelante:                             ' Rutina desp. adelante.
    for cuenta_pulsos = 1 to 60         ' Envía 60 pulsos hacia adelante.
      pulsout 12, 650                   ' Pulsos de 1.3 ms al servo derecho.
      pulsout 13, 850                   ' Pulsos de 1.7 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next
  goto principal                        ' Envía el programa a la rutina principal.

  atras:
    for cuenta_pulsos = 1 to 60         ' Envía 60 pulsos hacia atrás.
      pulsout 12, 850                   ' Pulsos de 1.7 ms al servo derecho.
      pulsout 13, 650                   ' Pulsos de 1.3 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next
  goto principal                        ' Envía el programa a la rutina principal.

  giro_izq:                             ' Rutina giro izquierda.
    for cuenta_pulsos = 1 to 30         ' Envía 30 pulsos de giro.
      pulsout 12, 650                   ' Pulsos de 1.3 ms al servo derecho.
      pulsout 13, 650                   ' Pulsos de 1.3 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next
  goto principal                        ' Envía el programa a la rutina principal.

  giro_der:                             ' Rutina giro derecha.
    for cuenta_pulsos = 1 to 30         ' Envía 30 pulsos de giro.
      pulsout 12, 850                   ' Pulsos de 1.7 ms al servo derecho.
      pulsout 13, 850                   ' Pulsos de 1.7 ms al servo izquierdo.
      pause 20                          ' Pausa de 20 ms.
    next
  goto principal                        ' Envía el programa a la rutina principal.
```

Cómo Funciona el Programa de Navegación por EEPROM

- Busque los comandos **`data`**, **`read`** e **`if...then`** en el Apéndice C: Referencia Rápida de PBASIC o en el BASIC Stamp Manual (en Inglés) antes de continuar.

El Programa 2.5 introduce dos nuevas variables. En lugar de ser de tipo word, son tipo byte, lo que significa que pueden almacenar números entre cero y 255. La primer variable es **`direc_EE`**, que es usada para especificar la dirección de la EEPROM a leer con el comando **`read`**. La segunda variable se llama **`instruccion`**. Esta variable es usada para almacenar el caracter de instrucción leído de la EEPROM.

```
'----- Declaración --------------
  cuenta_pulsos   var   word
  direc_EE        var   byte
  instruccion     var   byte
```

La siguiente declaración es la lista de datos a ser almacenada en la EEPROM. Estos datos son almacenados como una cadena de caracteres. Cuando estos caracteres son almacenados, son guardados como números que corresponden a las letras que ve en la directiva **`data`**. En la sección Su Turno de esta actividad, verá el mapa de memoria y observará los caracteres y su representación numérica en la EEPROM.

```
  data         "FFFBBLFFRFFQ"       ' Datos de movimiento
```

Aunque la rutina de inicialización es la misma que se usó en el Programa 2.3, no ocurre lo mismo con la rutina principal. La rutina principal primero lee el valor en la dirección 0 de la EEPROM y lo almacena en la variable **`instruccion`**. Luego, **`direc_EE`** es incrementado para que en el siguiente ciclo de lectura señale la dirección 1. Una serie de instrucciones **`if...then`** son usadas para decidir qué hacer, basándose en el caracter leído de la EEPROM, mediante la variable **`instruccion`**. Las instrucciones **`if...then`** buscan uno de los cuatro caracteres de instrucción: "F," "B," "R," y "L." Por ejemplo, si el caracter es una "R," las primeras dos instrucciones **`if...then`** son salteadas debido a que ninguna es verdadera. Como la tercer instrucción **`if...then`** es verdadera, el programa salta a la rutina **`giro_der`**.

```
principal:
  read direc_EE, instruccion
  direc_EE = direc_EE + 1

  if instruccion = "F" then adelante
  if instruccion = "B" then atras
  if instruccion = "R" then giro_der
  if instruccion = "L" then giro_izq
stop
```

Cuando el programa ingresa en la rutina **giro_der**, ejecuta 30 pulsos de giro a la derecha. Luego el comando **goto principal** envía el programa hacia la rutina **principal**.

```
giro_der:
  for cuenta_pulsos = 1 to 30
    pulsout 12, 850
    pulsout 13, 850
    pause 20
  next
goto principal
```

Cuando el programa regresa a la rutina **principal**, se lee la siguiente instrucción de la EEPROM y es controlada por las cuatro instrucciones **if...then**. El proceso se repite hasta que el caracter de salida "Q" es leído de la EEPROM. Cuando se carga "Q" en la variable **instruccion**, fallan las cuatro pruebas de **if...then**. En este caso, el programa no salta a ninguna de las rutinas de navegación, sino que ejecuta el comando que se encuentra a continuación de la serie de instrucciones **if...then**, que es el comando **stop**.

Su Turno

- Con el Programa 2.5 activo en el Stamp Editor, haga clic en Run y seleccione Memory Map.

Las instrucciones almacenadas aparecerán resaltadas en azul al principio del Mapa de EEPROM Detallado (Detailed EEPROM Map) como se muestra en la Figura 2.5. Los números mostrados son los códigos hexadecimales ASCII (American Standard Code for Information Interchange = Código Estándar Americano para Intercambio de Información) que corresponden a las letras que ingresó con la instrucción data.

- Haga clic para seleccionar Display ASCII (mostrar ASCII) en la parte inferior de la ventana, al lado del botón Close (cerrar).

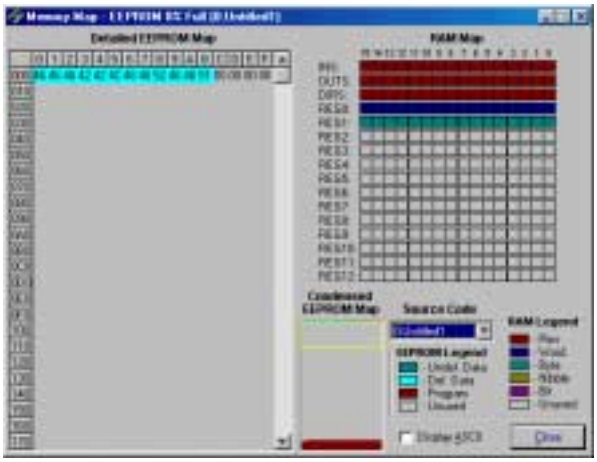


Figura 2.5: Memory Map (Mapa de Memoria) con las instrucciones almacenadas visibles en Detailed EEPROM Map.

Ahora las instrucciones de dirección aparecerán en un formato más familiar, como se muestra en la Figura 2.6. En lugar de códigos ASCII, aparecen los caracteres que almacenó usando la directiva **`data`**.

Como se encuentra escrito, el Programa 2.5 solamente puede almacenar 256 caracteres. Si la variable **`direc_EE`** es re-declarada como una variable word, podría direccionar más ubicaciones que las que posee la EEPROM. Tenga en cuenta que a medida que su programa se hace más largo, el número de direcciones de EEPROM disponibles para almacenamiento de datos, se hace más chico.

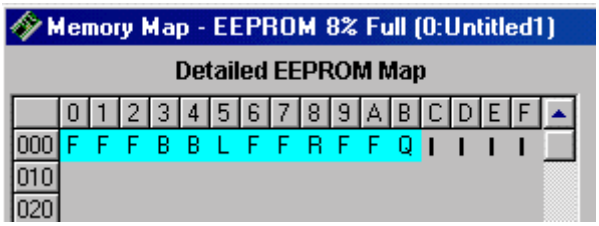


Figura 2.6: Acercamiento del Mapa de EEPROM Detallado, después de haber seleccionado Display ASCII.

Puede mover la cadena de datos existente a un nuevo juego de direcciones. También puede agregar instrucciones **`data`** adicionales. Los datos se almacenan secuencialmente, así que el primer caracter de la segunda cadena de datos se almacena inmediatamente después del último caracter de la primera cadena.

- Pruebe cambiar, agregar y eliminar caracteres de la directiva **`data`**. Recuerde que el último caracter de la directiva **`data`** siempre deberá ser una "Q."

- Pruebe agregar una segunda directiva **`data`**. Recuerde quitar la "Q" del final de la primer directiva **`data`** y agregarla al final de la segunda. De otra forma, el programa solamente ejecutará los comandos de la primer cadena de caracteres.

Actividad 6: Navegación Simplificada con Subrutinas

Todas las rutinas de navegación de la Actividad 5 finalizaron con comandos **`goto`**. Cada comando **`goto`** enviaba el control del programa a la etiqueta **`principal:`**. Una técnica similar involucra el uso de subrutinas y muchos programas de ejemplo de este libro harán uso de ellas. Dado que el programa de ejemplo de la Actividad 7 usa subrutinas, analicemos cómo trabajan antes de usarlas.

Programando Navegación con Subrutinas

Una subrutina es un segmento de código que realiza una tarea específica. Para que la subrutina realice su función, es necesario un comando en la rutina principal que "llame" a la subrutina. El comando para llamar a una subrutina es **`gosub`**, que es similar al comando **`goto`**. Un comando **`goto`** le dice al programa que se dirija a una etiqueta y comience a ejecutar instrucciones. El comando **`gosub`** le dice al programa que se dirija a una etiqueta y comience a ejecutar instrucciones, pero con la diferencia que debe regresar cuando termine. Una subrutina termina cuando se ejecuta el comando **`return`**.

El Programa 2.6 usa una técnica diferente para la navegación. Usa valores variables para la cantidad y el ancho de los pulsos entregados a los servos. Después de asignar valores a las variables, el programa principal llama una subrutina que usa la información almacenada en las variables, para enviar los pulsos a los servos.

```
' ¡Robótica! v1.5, Programa 2.6: Navegación por Subrutinas.

'----- Declaración --------------

cuenta_bucles       var   word          ' Contador de bucle.
ancho_der           var   word          ' Variable para ancho pulso derecho.
ancho_izq           var   word          ' Variable para ancho pulso izquierdo.
cuenta_pulsos       var   byte          ' Número de pulsos a enviar.

'----- Inicialización ------------

output 2                                ' Configura a P2 como salida.
freqout 2, 2000, 3000                   ' Indicador de batería baja.
low 12                                  ' P12 y 13 salidas bajas.
low 13

'----- Rutina Principal --------------

principal:                              ' Rutina principal.

  adelante:                             ' Rutina hacia adelante.
    cuenta_pulsos = 60                  ' Ajusta número de pulsos a 60.
    ancho_der = 650                     ' Ajusta ancho de pulso derecho a 1.3 ms.
    ancho_izq = 850                     ' Ajusta ancho de pulso izquierdo a 1.7 ms.
    gosub pulsos                        ' Llama a la subrutina pulsos.
    pause 500                           ' Pausa de 0.5 s.

  atras:                                ' Rutina hacia atrás.
    cuenta_pulsos = 60                  ' Ajusta número de pulsos a 60.
    ancho_der = 850                     ' Ajusta ancho de pulso derecho a 1.7 ms.
    ancho_izq = 650                     ' Ajusta ancho de pulso izquierdo a 1.3 ms.
    gosub pulsos                        ' Llama a la subrutina pulsos.
    pause 500                           ' Pausa de 0.5 s.

goto principal                          ' Salta a la etiqueta principal.

'----- Subrutinas ---------------

pulsos:                                      ' Subrutina de pulsos.
  for cuenta_bucles = 1 to cuenta_pulsos     ' cuenta_pulsos = cant. de pulsos.
    pulsout 12, ancho_der               ' ancho_der ancho pulso derecho.
```

```
    pulsout 13, ancho_izq              ' ancho_izq ancho pulso izquierdo.
    pause 20                           ' Pausa de 20 ms.
  next
return                                 ' Regresa a donde fue llamada.
```

Cómo Funciona el Programa de Navegación por Subrutinas

- Busque los comandos **`gosub`** y **`return`** en el Apéndice C: Referencia Rápida de PBASIC o en el BASIC Stamp Manual (en Inglés) antes de continuar.

La Figura 2.7 muestra cómo la rutina **`atras`** llama a la subrutina **`pulsos`**. Los comandos de la subrutina **`pulsos`** son ejecutados hasta que el programa se encuentra con el comando **`return`**. El comando **`return`** envía el control del programa de regreso al comando inmediatamente posterior a la instrucción **`gosub pulsos`**, que es **`pause 500`** en este caso. Este mismo proceso se repitió en la rutina **`adelante`**, que se ejecutó antes que la rutina **`atras`**.

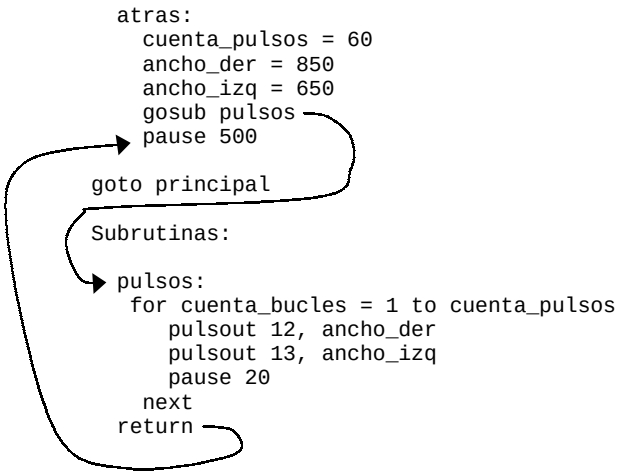


Figura 2.7: Flujo del programa con subrutinas.

Su Turno

- Agregue instrucciones a la rutina principal que fijen los valores para **`giro_der`** y **`giro_izq`**.

Actividad 7: Pongamos Todo Junto

Este capítulo ha introducido varias técnicas de navegación:

- Rutinas de navegación que controlan distancia y ángulo de giro
- aceleración
- almacenado de listas largas
- llamado de subrutinas
- uso de variables para fijar los valores usados por las subrutinas

Todos estos elementos pueden ahora unirse para crear un programa de navegación más robusto.

Programación de Navegación por EEPROM con Aceleración

La directiva **data** del Programa 2.7 almacena dos cosas diferentes en la EEPROM para cada maniobra: la dirección y la cantidad de pulsos a emitir. Ambos valores son almacenados en variables. La subrutina que envía los pulsos a los servos realiza automáticamente la aceleración y desaceleración, para cada maniobra.

```
' ¡Robótica! v1.5, Programa 2.7: Navegación por EEPROM con Aceleración

'----- Declaración --------------

  cuenta_pulsos     var   byte          ' Cantidad de pulsos a entregar.
  cuenta_bucles     var   byte          ' Contador del bucle for...next.
  ancho_der         var   word          ' Almacena ancho del pulso derecho.
  ancho_izq         var   word          ' Almacena ancho del pulso izquierdo.
  der_ant           var   word          ' Almacena ancho del pulso der. anterior.
  izq_ant           var   word          ' Almacena ancho del pulso izq. anterior.
  dif_der           var   word          ' Diferencia entre valor actual y final.
  dif_izq           var   word          ' Diferencia entre valor actual y final.
  direc_EE          var   byte          ' Almacena e inc. la direc. de la EEPROM.
  instruccion       var   byte          ' Almacena la instruc. de la EEPROM.
  direccion         var   byte          ' Almacena número para salto condicional.

  data         "F", 250, "R", 30, "B", 60, "L", 60, "F", 20, "Q"

'----- Inicialización ------------

  output 2                             ' P2 salida.
  freqout 2, 2000, 3000                ' Señal de reset.
  low 12                               ' P12 y 13 salidas bajas.
  low 13
  ancho_der = 750                      ' Fija valores iniciales de variables.
  ancho_izq = 750
  der_ant = 750
  izq_ant = 750
  cuenta_pulsos = 0

'----- Rutina Principal --------------

principal:                             ' Rutina Principal.

  gosub pulsos                         ' Llama a la subrutina pulsos.

  read direc_EE, instruccion           ' Lee la primer instrucción de la EEPROM.
  read direc_EE + 1, cuenta_pulsos     ' Lee la segunda instrucción.
  direc_EE = direc_EE + 2              ' Incrementa direc_EE en 2 unidades.
```

```
  ' La instrucción lookdown es usada junto con la instrucción branch para enviar
  ' el programa hacia la rutina apropiada para fijar los anchos de los pulsos.
  ' Después de fijar los valores, el programa regresa a la rutina principal.
  ' El primer comando de la rutina principal llama a la subrutina pulsos.

  lookdown instruccion, = ["B","L","R","F","Q"], direccion
  branch direccion,[atras,giro_izq,giro_der,adelante,salir]

      atras:        ancho_der = 850: ancho_izq = 650: goto principal
      giro_izq:     ancho_der = 650: ancho_izq = 650: goto principal
      giro_der:     ancho_der = 850: ancho_izq = 850: goto principal
      adelante:     ancho_der = 650: ancho_izq = 850: goto principal

      salir:        stop

'----- Subrutinas ---------------

  ' Los valores de ancho_izq y ancho_der son comparados con sus valores previos.
  ' Las variables dif_der y dif_izq se usan para determinar los anchos
  ' de los pulsos.  Para una explicación mas extensa vea la sección Cómo
  ' Funciona la Navegación por EEPROM con Aceleración.

  pulsos:
    dif_der = ancho_der - der_ant
    dif_izq = ancho_izq - izq_ant

    for cuenta_bucles = 1 to cuenta_pulsos

      dif_der = dif_der - 2 + (4*dif_der.bit15)
      dif_izq = dif_izq - 2 + (4*dif_izq.bit15)

      pulsout 12, ancho_der - dif_der
      pulsout 13, ancho_izq - dif_izq

      pause 20

    next

    der_ant = ancho_der
    izq_ant = ancho_izq

  return
```

Tema Avanzado: Cómo Funciona la Navegación por EEPROM con Aceleración

- Busque cada uno de los siguientes comandos en el Apéndice C: Referencia Rápida de PBASIC o en el BASIC Stamp Manual (en Inglés) antes de continuar: **`lookdown, branch.`**

Seis variables tamaño word fueron agregadas a la sección de declaración del Programa 2.5. Las variables **`izq_ant`** y **`der_ant`** son para almacenar la duración de pulso anterior de cada servo. Las variables **`dif_der`** y **`dif_izq`** cambian ligeramente cada vez que se aplica un pulso. Cuando la diferencia es grande, el valor del ancho de pulso es cercano al valor de los pulsos anteriores. Cuando la diferencia es pequeña, los pulsos se acercan al valor de duración de pulso nuevo.

La directiva **`data`** ha sido reestructurada para que almacene la dirección, seguida por el número de pulsos. De esta forma, el Boe-Bot puede recibir instrucciones como: ir adelante 250 pulsos, luego derecha 30 pulsos, luego atrás 60 pulsos, etc.

Se fijan valores iniciales para las variables que registran el ancho del pulso. Todas se inician con el valor central de 750. El primer comando de la rutina **`principal`** es **`gosub pulsos`**. Dado que la cantidad inicial de pulsos a emitir es "0" y los valores iniciales de las instrucciones de los servos están en el centro, la rutina pulsos no hará nada la primera vez. Todas las veces siguientes que se ejecute la rutina **`principal`**, el comando **`gosub`** llamará a la subrutina **`pulsos`**, que emitirá los pulsos a los servos con los valores determinados en la ejecución anterior de la rutina **`principal`**.

Ahora hay dos comandos **`read`** en la rutina **`principal`**. Esto es debido a que se almacenan los valores de dirección y cantidad de pulsos. El primer comando **`read`** obtiene una letra que representa la dirección del Boe-Bot. El segundo comando **`read`** lee la ubicación **`direc_EE + 1`**, donde se encuentra el número que sigue a cada letra de la instrucción **`data`**. Esta es una forma eficiente de obtener la cantidad de pulsos con un solo comando. Luego, **`direc_EE`** se incrementa dos unidades en lugar de una. De este modo, la próxima vez que el primer comando **`read`** busque una letra, salteará la ubicación del número (de pulsos) y encontrará la siguiente letra en la secuencia.

Ahora que una letra que indica la dirección ha sido almacenada en la variable **`instruccion`** y la cantidad de pulsos en la variable **`cuenta_pulsos`**, es hora de que el BASIC Stamp se encargue de averiguar el significado de la letra almacenada en **`instruccion`**. En lugar de usar una serie de instrucciones **`if...then`**, se usan dos comandos. El primero es el comando **`lookdown`**. Este comando tomará el valor "B" de **`instruccion`** y encontrará que es el primero de la lista. En realidad, en lo que respecta al comando **`lookdown`**, tiene la ubicación cero ya que comienza a contar desde cero. Así, almacena el valor 0 en la variable **`direccion`**. Si **`instruccion`** es "L," se almacenará el número 1 en **`direccion`**. Si **`instruccion`** es "R," el número 2 será almacenado en **`direccion`** y así sucesivamente.

El comando **branch** salta hacia una etiqueta en función del valor de una variable. Si **direccion** es "0," **branch** saltará a la rutina **atras**. Si **direccion** es "1," **branch** saltará a la etiqueta **giro_izq**. Si **direccion** es "2," **branch** saltará a **giro_der** y así sucesivamente.

Tal vez no sabía que podía poner más de un comando, separado por dos puntos, en la misma línea. Las rutinas **atras**, **giro_izq**, **giro_der**, **adelante** y **salir** solamente ocupan una línea. Lo único que hay que tener en cuenta cuando se usan varios comandos PBASIC en la misma línea, es que las etiquetas solamente se pueden usar al principio. Cada una de las rutinas **direccion** fijan los valores de duración **ancho_der** y **ancho_izq**, al igual que en programas anteriores. En lugar de llamar a la rutina pulsos, cada una de las cuatro rutinas de dirección termina con un comando **goto principal**. Observe que tan pronto como el programa llega a **principal,** llama a la subrutina **pulsos**.

La subrutina **pulsos** primero determina la diferencia entre el último valor enviado a cada servo y el valor actual. Tomando el servo derecho como ejemplo, la variable que almacena este valor es **dif_der**, debido a que almacena el ancho de **ancho_der** menos **der_ant**. El proceso para el servo izquierdo es el mismo. Luego comienza el bucle **for...next** que aplica la cantidad de pulsos **cuenta_pulsos**.

Asumiendo que **ancho_der** difiere de **der_ant**, el valor **dif_der** será distinto de cero. Dado que **dif_der** se resta de **ancho_der** cuando se ejecuta el comando **pulsout**, es razonable que el valor inicial de **dif_der** sea grande. Luego, con cada pulso, el valor **dif_der** deberá ser cada vez más pequeño, hasta que el valor del ancho del pulso llegue a su valor **ancho_der**. Un único comando se ocupa de decrementar **dif_der** si es positivo e incrementarlo si es negativo.

```
dif_der = dif_der - 2 + (4*dif_der.bit15)
```

¿Cómo funciona esta expresión? Si **dif_der** es un número positivo, **dif_der.bit15** será cero. Así, si **dif_der** es positivo, la ecuación le resta 2 unidades. Sin embargo, si **dif_der** es negativo, **dif_der.bit15** es "1". Esto es debido al sistema binario de complemento a dos, que usa el BASIC Stamp para trabajar con número negativos. Esto significa que aparece un 4 sumando en la ecuación, dando como resultado un incremento de **dif_der** en dos unidades. El mismo principio se aplica a **dif_izq**.

Luego se generan los pulsos. Observe que cada argumento de duración es una expresión. Para el comando **`pulsout`** de P12, la duración es **`ancho_der – dif_der`** y para P13, es **`ancho_izq – dif_izq`**.

Su Turno

- Modifique la instrucción **`data`** para lograr que el Boe-Bot dibuje in rectángulo de un metro de lado.

Sumario y Aplicaciones

Se armó y verificó un indicador de batería baja, usando un programa en el BASIC Stamp. Este capítulo también introdujo varios programas PBASIC para hacer que el Boe-Bot realice distintas maniobras. Los ejemplos incluyen control sobre distancia y giros del Boe-Bot, junto con los métodos para programar distancias específicas. Se introdujeron ejemplos de aceleración, navegación por EEPROM, así como un ejemplo de integración de algoritmos de navegación.

Ejemplo del Mundo Real

Cuando las pilas del Boe-Bot bajen de cierto nivel, el Boe-Bot enviará una señal indicando que algo está mal, generando una serie de sonidos emitidos por un parlante piezoeléctrico. Estos parlantes son muy comunes. Piense en todos los artefactos que emiten un sonido cuando se presiona una tecla o algo así. Su horno microondas, las registradoras de los supermercados y los teclados de alarma usan esta característica. No debería extrañarnos que cada uno tenga un microcontrolador monitoreando las teclas del teclado. Los sensores de nivel también son de uso común en la industria, donde tensión, presión, temperatura y una gran variedad de condiciones deben ser controladas. Los microcontroladores son usados a menudo para monitorear estas condiciones y hacer sonar una sirena de alarma cuando el proceso se sale de los niveles aceptables.

El movimiento microcontrolado también se encuentra en nuestros alrededores. Aunque puede que aún no haya muchos robots autónomos en su casa, hay muchos otros artefactos con partes con movimientos microcontrolados. Cabezales de impresoras y disqueteras de computadoras son dos ejemplos que usan motores paso a paso. Servos controlados por microcontroladores son usados en muchos lugares. Muchos sistemas de automóviles tienen servos controlando pequeñas piezas móviles en varios sistemas de emisión e ignición. Los servos industriales se emplean en muchos procesos de fabricación, a menudo junto con los sensores de nivel mencionados anteriormente.

Aplicaciones para el Boe-Bot

La navegación programada es la base para muchas actividades del Boe-Bot. En la sección Proyectos, trabajará en la programación de un Boe-Bot, para lograr que se mueva describiendo distintas formas y ajustando algunos algoritmos desarrollados en este capítulo. En el Capítulo 3, usaremos algunas de estas rutinas para responder a las entradas detectadas por los sensores. Las rutinas de navegación desarrolladas en este capítulo serán muy útiles a la hora de evitar obstáculos. Una de las ocupaciones más populares de los robots autónomos, es resolver laberintos llenos de obstáculos. El Apéndice I tiene las reglas de competición para el Boe-Bot. Responder automáticamente a ciertas situaciones puede aumentar la capacidad de navegación del Boe-Bot dentro del laberinto. Por ejemplo, cuando se detecta una esquina, en lugar de intentar dar la vuelta basándose en los datos de los sensores, el Boe-Bot puede acudir a rutinas almacenadas en la EEPROM.

Preguntas y Proyectos

Preguntas

1. ¿Cómo funciona el indicador de pilas bajas del Boe-Bot?
2. Nombre y describa los tres argumentos necesarios para hacer funcionar un comando **`freqout`**.
3. ¿Qué argumento del Programa 2.2 controla la distancia de la rutina **`adelante`**?
4. ¿Cuál es el argumento de duración de **`pause`** antes del **`stop`** en el Programa 2.2?
5. ¿Cuáles son los anchos de pulso de los servo, en ms, para adelante, atras, giro_der y giro_izq?
6. ¿Qué característica del bucle **`for...next`** del PBASIC hace que desacelerar sea tan fácil como acelerar en el Programa 2.4?
7. ¿Cuál es la diferencia entre rutina y subrutina? ¿Qué comando es usado para llamar a una subrutina? ¿Qué sucede cuando finaliza una subrutina? ¿Qué comando indica que finalizó una subrutina y qué es lo que hace?
8. ¿Cuántas posiciones de memoria tiene la EEPROM del BASIC Stamp? ¿En qué posición de memoria EEPROM comienza el programa y en qué sentido avanza cuando el programa se vuelve más grande? ¿En qué posición de la EEPROM comienza el almacenamiento de datos y hacia que posición se almacenan los datos siguientes? ¿Qué comando se utiliza para leer datos de la EEPROM?
9. ¿Qué argumento se coloca después del **`if`** en una instrucción **`if...then`**? ¿Qué argumento se coloca después de **`then`**?
10. ¿Cómo hace el comando **`lookdown`** del Programa 2.7 la transición de letras leídas de la EEPROM a los números que utiliza el comando **`branch`**?

Ejercicios

1. Ajuste el tono generado en el Programa 2.1 para que tenga una frecuencia de 1.5 kHz con una duración de tres segundos.

2. Suponga que su programa se ha vuelto demasiado grande y necesita lugar en la RAM. Modifique la declaración de la variable **`cuenta_pulsos`** del Programa 2.2 para que utilice la mitad de RAM que usa actualmente (8-bits en lugar de 16).

3. Aumente el tiempo de **`pause`** entre giro a la derecha y giro a la izquierda del Programa 2.3 a 2 s.

4. Modifique el Programa 2.4 para que acelere al doble de velocidad.

5. El Programa 2.5 interpreta a cualquier letra distinta de "F," "B," "R," y "L", como fin de programa. Modifíquelo de forma que solamente acepte la letra "Q."

6. Modifique el Programa 2.6 para que pueda moverse por 65534 pulsos en una sola ejecución de subrutina.

7. Modifique el Programa 2.7 de forma que el comando **`lookdown`** ya no sea necesario. Pista: Almacene números en lugar de letras para el control de dirección.

Proyectos

Un aspecto de la navegación del Boe-Bot que no se trató en este capítulo es realizar curvas. Se realiza una curva cuando la velocidad de las ruedas (en el mismo sentido) no es la misma. Las curvas también se producen cuando una rueda acelera a distinto ritmo que la otra. Tenga en cuenta que la cantidad de pulsos entregados, es una forma de controlar la distancia a un ancho de pulso dado (velocidad).

1. El Boe-Bot está siendo usado para transportar material reactivo, en particular sodio sólido y agua. Si reaccionan, explotan, dejando a su Boe-Bot como una pila de escombros. Para transportar cuidadosamente los químicos, necesitará comenzar a moverse con una aceleración pequeña. Cree un programa para que el Boe-Bot se desplace en una zona de un metro cuadrado, con transiciones entre giros muy suaves.

2. Cree un patrón de movimiento simple con varias direcciones, por ejemplo F,B,R,R,F,F,L y finalmente F. Este patrón será almacenado y leído de la EEPROM. Cuando termine de ejecutar el patrón, ejecutará el mismo patrón pero en sentido inverso.

3. Cree el código fuente para los siguientes patrones de movimiento:

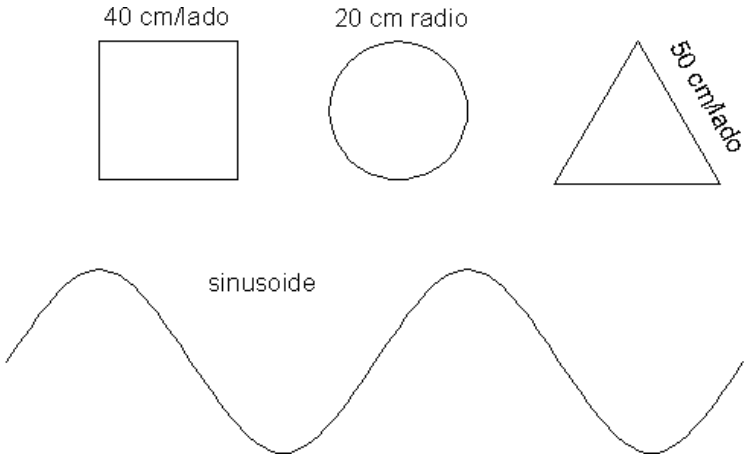


Figura 2.8: Patrones para programar al Boe-Bot.

Capítulo 3: Exploración Táctil con Bigotes

Exploración Táctil

El kit de Bigotes (Whiskers kit) se llama así debido a la apariencia de los interruptores, aunque algunos podrían pensar que parecen antenas. De cualquier modo, los bigotes le dan al Boe-Bot la capacidad de reconocer el mundo exterior con sensores táctiles. El Boe-Bot puede usar estos bigotes para navegar solamente por tacto. Aunque las actividades de este capítulo se enfocan en usar solamente los bigotes, también pueden usarse junto con otros sensores para aumentar la funcionalidad del Boe-Bot.

Actividad 1: Colocar y Probar los Bigotes

Materiales

(2) Resistores de 10 kΩ
(2) Conectores de 3 pines
(2) Separadores macho/hembra de 3/8" 4/40
(2) Tornillos ¼" 4/40
(2) Antenas
(2) Arandelas de nylon Nº 4

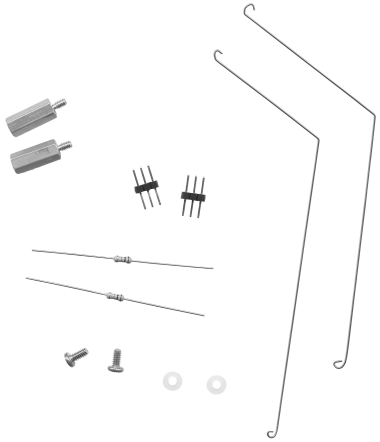


Figura 3.1 Materiales para colocar los Bigotes

¡Constrúyalo!

Necesitará usar su destornillador Phillips y una llave de ¼". Antes de comenzar el montaje de los bigotes, observe atentamente la Figura 3.2. Use estas imágenes como guía para la construcción de la parte mecánica del kit de Bigotes. La Figura 3.3 muestra el diagrama de conexión de los bigotes. Sígalo para realizar las conexiones eléctricas necesarias.

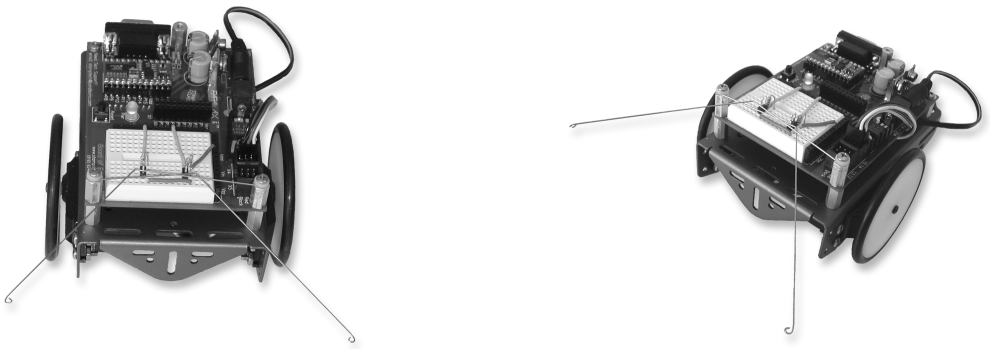


Figura 3.2: Imágenes del Boe-Bot con Bigotes

- Quite los dos tornillos que sujetan la parte delantera de su Plaqueta de Educación a los separadores.
- Atornille en su lugar los separadores que vienen con el kit de Bigotes.

> ✓ **TIP**
>
> Deberá sujetar el separador que está junto al puerto de servos y girar el separador que sostiene la BOE al chasis del Boe-Bot para poder atornillarlos. El separador que sostiene a la BOE no girará hasta que afloje el tornillo que lo sujeta al chasis. Asegúrese de apretarlo cuando termine.

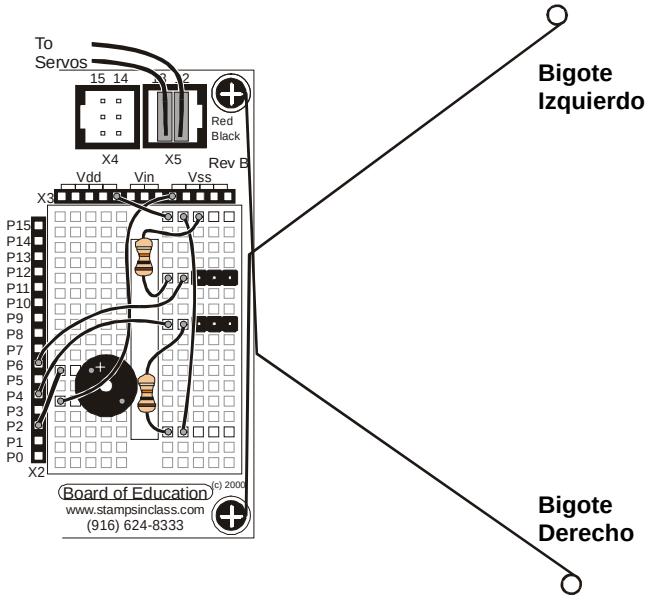


Figure 3.3: Diagrama de conexión de Bigotes.

- Coloque una arandela de nylon en la parte superior de cada separador.
- Pase los tornillos que sacó en el primer paso por el bucle de cada bigote.
- Coloque los tornillos sujetando al bigote entre la arandela de nylon y la cabeza del tornillo. Asegúrese de orientar los bigotes como en las Figuras 3.2 y 3.3.

Los Bigotes como Entradas

La Figura 3.4 es una representación esquemática del circuito que acaba de construir. Cada bigote es una extensión mecánica, así como también la conexión eléctrica a masa de un interruptor normalmente abierto. El BASIC Stamp puede ser programado para detectar cuándo un bigote es presionado. Los pines de E/S conectados a cada interruptor, controlan la tensión en el resistor de pull-up de 10 kΩ. Cuando un bigote no está presionado, la tensión del pin es 5 V (1 lógico). Cuando un bigote es presionado, la línea de E/S es puesta a masa, de forma que se detecta 0 V (0 lógico).

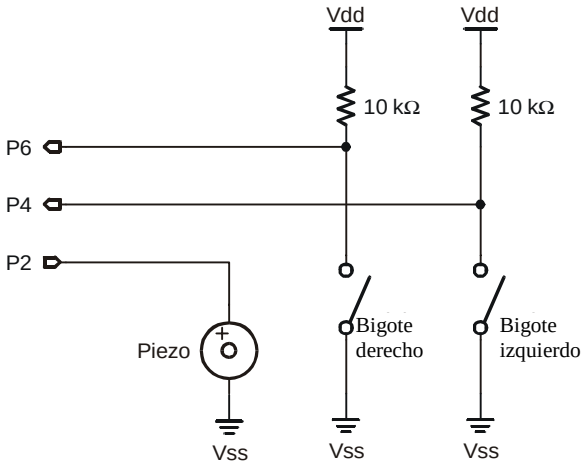


Figure 3.4: Esquema de Bigotes.

Probando los Bigotes

Se puede probar cada bigote con la Debug Terminal. Esta vez, en lugar de imprimir un mensaje, la Debug Terminal es usada para mostrar la tensión de entrada que se observa en los pines de E/S conectados a los bigotes.

Cada pin de E/S del BASIC Stamp está conectado a una posición de memoria RAM. Estas posiciones de memoria pueden verse en el Memory Map (Mapa de Memoria). Esta es la misma ventana usada para ver la EEPROM del BASIC Stamp en el Capítulo 2, Actividad 5.

- Abra el Mapa de Memoria seleccionando Memory Map en el menú Run del Stamp Editor.

Las primeras tres filas de la RAM Map son ubicaciones de memoria conectadas directamente a los pines de E/S. La fila de números de la parte superior indica la dirección de bit para cada posición de una fila dada. Las filas **ins**, **outs** y **dirs** son los registros de E/S.

La primer fila corresponde al registro **ins** y representa una zona de memoria que almacena 16 bits, cada uno de los cuales puede ser 1 o 0. Cada bit del registro **ins** controla el estado de un pin de E/S. Bit-0 corresponde al pin de E/S P0, bit-1 a P1 y así hasta bit-15, que corresponde al pin de E/S P15.

Cada bit del registro **outs**, puede ser usado para fijar el valor de salida de un pin de E/S en 5 V o 0 V. Cuando el valor de un bit del registro **outs** es 1, la salida del pin correspondiente será 5 V. Cuando el valor del bit es 0, la salida será 0 V.

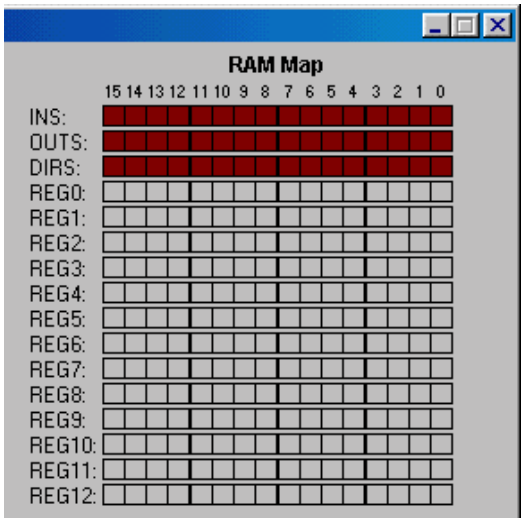


Figura 3.5: Mapa RAM de la Ventana Memory Map.

Dado que un pin de E/S de un BASIC Stamp no puede ser entrada y salida a la vez, la dirección de cada pin es fijada por el registro **dirs**. Un pin de E/S es fijado como entrada o salida, dependiendo del valor del registro **dirs**.

El Programa 3.1 está diseñado para verificar que los bigotes funcionen apropiadamente. Controla y muestra el estado de los pines de E/S del BASIC Stamp conectados a los bigotes. Todos los pines de E/S se configuran como entradas cada vez que se inicia un programa PBASIC. Esto significa que los pines conectados a los bigotes, funcionarán como entradas automáticamente. Como entrada, un pin de E/S conectado a un bigote, hará que su bit correspondiente del registro **ins** muestre un 1 si la tensión es 5 V (bigote no presionado) y 0 si la tensión es 0 V (bigote presionado). La Debug Terminal puede ser usada para mostrar estos valores.

- Ingrese y ejecute el Programa 3.1
- Este programa usa la Debug Terminal, así que deberá dejar conectado el cable serial a la BOE mientras el Programa 3.1 se esté ejecutando.

```
'¡Robótica! v1.5, Programa 3.1: Prueba de Bigotes.

bucle:
  debug home, "P6 = ", bin1 in6, "   P4 = ", bin1 in4
  pause 50
goto bucle
```

- Observe los valores mostrados en la Debug Terminal; debería verse a P6 y P4 iguales a 1.
- Mire la Figura 3.3 para identificar al "bigote izquierdo" y el "bigote derecho".
- Presione el bigote derecho hasta que toque el conector de tres pines de la derecha y observe los valores mostrados en la Debug Terminal nuevamente. Debería leer: P6 = 1 P4 = 0.
- Presione el bigote izquierdo hasta que toque el conector de tres pines y observe los valores mostrados en la Debug Terminal nuevamente. Esta vez debería leer: P6 = 0 P4 = 1.
- Presione ambos bigotes contra los conectores de tres pines. Ahora debería leer P6 = 0 P4 = 0.

Si los bigotes pasaron todas las pruebas, puede continuar; caso contrario, busque errores en su programa y su circuito.

Cómo Funciona el Programa que Prueba los Bigotes

Los únicos elementos nuevos en el Programa 3.2 son **`in6`** y **`in4`**. En lugar de variables tipo word o byte, estas son variables tipo bit. Estas variables tipo bit son especiales debido a que almacenan el valor de entrada de un pin de E/S en particular. La variable **`in4`** se refiere al bit-4 del registro **`ins`** mientras que **`in6`** se refiere al bit-6. Cuando el pin de E/S P4 del BASIC Stamp detecta 0 V, el valor de**`in4`** es 0. Cuando P4 detecta 5 V, el valor de **`in4`** es 1. La variable **`in6`** trabaja del mismo modo, excepto que controla a P6 en lugar de P4.

Su Turno

Suponga que debe probar los bigotes lejos de una computadora. Dado que no dispondrá de la debug terminal, ¿qué puede hacer? Una solución sería programar el BASIC Stamp para que emita una señal de salida que corresponda a la señal de entrada que recibe. Una forma de hacer esto podría ser con un par de LEDs y un programa que encienda y apague los LEDs en base a las entradas de los bigotes. La Figura 3.6 muestra los componentes necesarios junto con sus símbolos eléctricos.

Componentes Extra
(2) LEDs Rojos
(2) Resistores de 470 Ω

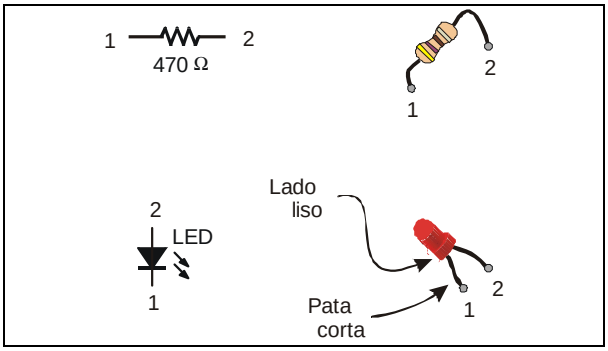


Figura 3.6 Componentes extra

FYI	LED es la sigla de light emitting diode (diodo emisor de luz). El terminal rotulado 1 en la figura 3.6 es el cátodo del LED y el otro es el ánodo. Normalmente puede identificar cuál de las dos patas del LED es el cátodo, debido a que es la más corta.

✓ **TIP**	Si se hubiesen cortado las patas del LED por algún motivo, perderemos la referencia anterior. Por este motivo, el encapsulado plástico tiene un rebaje del lado del cátodo. Observe con cuidado el borde que tiene el LED en su base y notará que del lado del cátodo, éste desaparece.

- ❑ Construya el circuito que se muestra en la Figura 3.7.

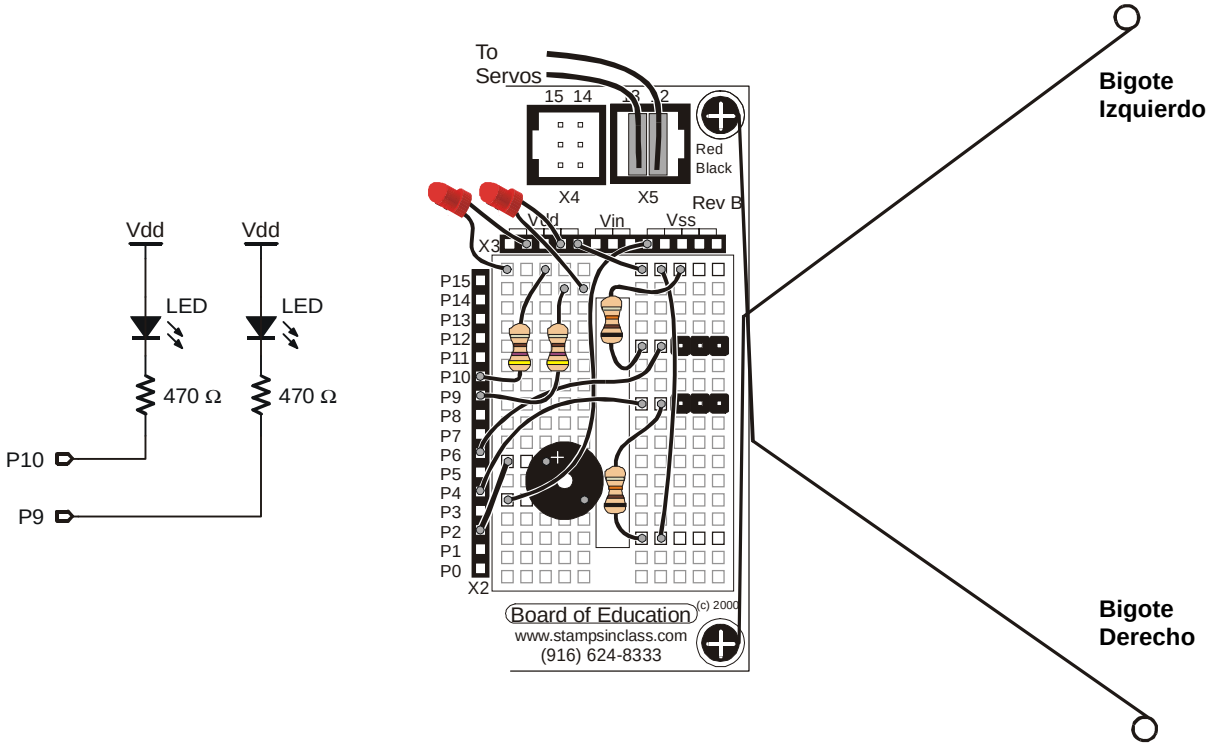


Figura 3.7 (a): agregue este circuito, (b) para que los bigotes se vean así cuando termine.

- Agregue estos comandos al principio del Programa 3.1, arriba de la etiqueta **`bucle:`**

```
output 9
output 10
```

Estos comandos cambian la dirección de P9 y P10 de entrada a salida. Ahora en lugar de recibir señales, se encargarán de emitir señales.

- Además, en el Programa 3.1, agregue estos comandos debajo de la línea **`debug home...:`**

```
out9 = in4
out10 = in6
```

Estas instrucciones igualan los valores de salida de P9 y P10 a los valores que se presenten en la entrada de **in4** e **in6** respectivamente. Si **in4=1**, **out9** valdrá 1. Esto significa que cuando **in4** vea 5 V, **out9** emitirá 5 V. Si **in4** es 0, lo que significa que detecta 0 V, entonces **out9** también será 0, emitiendo 0 V.

- Ejecute su versión modificada del Programa 3.1 y pruebe los bigotes usando los LEDs para indicar que bigote del BASIC Stamp ha sido presionado.

Actividad 2: Exploración con Bigotes

En la Actividad 1, el BASIC Stamp fue programado para detectar si algún bigote había sido presionado. En esta actividad, el BASIC Stamp será programado para usar esa información para guiar al Boe-Bot. Cuando el Boe-Bot está avanzando y un bigote es presionado, significa que se ha topado con algún obstáculo. Un programa de navegación debe recibir esta información, decidir que significa y llamar a una rutina de navegación que pueda evitar el obstáculo.

Programación del Boe-Bot para que Explore Basándose en los Bigotes

El Programa 3.2 hace que el Boe-Bot vaya hacia adelante hasta que encuentra un obstáculo. Es este caso, el Boe-Bot sabe que encontró un obstáculo, al chocar con él con uno o ambos bigotes. Tan pronto como los bigotes detectan el obstáculo, se usan las rutinas y subrutinas de navegación desarrolladas en el Capítulo 2 para que el Boe-Bot retroceda y gire. Luego, el Boe-Bot reasume su movimiento hacia adelante hasta chocar con otro obstáculo.

Cuando se presiona un bigote, debido a un obstáculo, se cierra el interruptor del tipo normalmente abierto. Los pines P6 y P4 se configuran como entradas y se usan para monitorear el estado de los bigotes. Los dos bigotes pueden estar en uno de cuatro estados posibles:

(1) Ambos en 1 – ningún objeto detectado
(2) Izquierdo 0, derecho 1 – objeto detectado a la izquierda
(3) Derecho 0, izquierdo 1 – objeto detectado a la derecha
(4) Ambos en 0 – indica colisión frontal contra un objeto grande (una pared por ejemplo)

El Programa 3.2 muestra un ejemplo de cómo pueden usarse los bigotes para seleccionar la rutina de navegación del Boe-Bot apropiada. Por ejemplo, estado 1 significa que el Boe-Bot puede seguir adelante. Estado 2 significa que el Boe-Bot debería retroceder y luego girar a la derecha. Estado 3 significa que el Boe-Bot debería retroceder y girar a la izquierda y estado 4 sería un buen momento para retroceder y hacer un giro en U.

- Ejecute el Programa 3.2 y vea cómo se comporta el Boe-Bot cuando choca contra una pared.

```
' ¡Robótica!, Programa 3.2: Exploración con Bigotes.

'----- Declaración --------------

  cuenta_pulsos  var   byte                    ' Contador del for...next.

'----- Inicialización ------------

  output 2                                     ' Fija a P2 como salida.
  freqout 2, 2000, 3000                        ' Indicador de reset.
  low 12                                       ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:

  control_bigotes:                             ' Controla cada bigote.

    if in6  = 0 and in4 = 0 then giro_u        ' Atrás si tocan ambos.
    if in6  = 0 then giro_der                  ' Derecha si toca el izquierdo.
    if in4  = 0 then giro_izq                  ' Izquierda si toca el derecho.

  adelante:                                    ' Si no detecta, un pulso adelante.
    pulsout 12,650
    pulsout 13,850
    pause 20

goto principal                                 ' Controla nuevamente.

'----- Rutinas de Navegación -------

  giro_izq:                                    ' Rutina giro izquierda.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
  goto principal

  giro_der:                                    ' Rutina giro derecha.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
```

```
      pulsout 12, 850
      pulsout 13, 850
      pause 20
    next
  goto principal

  giro_u:                                          ' Rutina de giro en U.
    gosub atras                                    ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 60
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
  goto principal

'----- Subrutina de Navegación -----

  atras:                                           ' Usada por cada rutina de
    for cuenta_pulsos = 0 to 60                    ' navegación.
      pulsout 12, 850
      pulsout 13, 650
      pause 20
    next
  return
```

El diseño mecánico de los bigotes no es a prueba de fallas. La Tabla 3.1 lista algunos problemas comunes que podrá encontrar, junto con las soluciones sugeridas.

Tabla 3.1: Solución de Problemas con los Bigotes

Problema	Pruebe esto
El Boe-Bot retrocede poco o demasiado.	Ajuste los argumentos de **`for. . . next`** del programa. Pueden variarse para regular la distancia recorrida por el Boe-Bot cuando retrocede o gira.
El Boe-Bot choca lateralmente contra la pared y los bigotes resbalan sin poder detectarla.	Modifique la resistencia de los bigotes contra la superficie de un objeto, doblando los bigotes con un ángulo diferente. También puede probar bañando el bigote con un material pegajoso.
Los Bigotes no detectan objetos estáticos al frente.	Doble los bigotes hacia adentro.
Los interruptores no funcionan apropiadamente.	Repita la Actividad 1.

Cómo Funciona Exploración con Bigotes

Las instrucciones **`if...then`** de la rutina **`principal`** primero revisan el estado de los bigotes. Si ambos están presionados, llama la rutina **`giro_u`**. Si solamente se encuentra presionado el bigote izquierdo, se activa la entrada P6 y se llama la rutina **`giro_der`**. Si se presiona el bigote derecho, se activa la entrada P4, llamando a la rutina **`giro_izq`**.

```
principal:
  control_bigotes:
    if in6  = 0 and in4 = 0 then giro_u
    if in6  = 0 then giro_der
    if in4  = 0 then giro_izq
```

Si no se presiona ninguno de los bigotes, la acción por defecto es ejecutar una versión modificada de la rutina **`adelante`**. La rutina **`adelante`** ha sido modificada de forma que solamente entrega un pulso, antes de regresar a verificar el estado de los bigotes. La clave para entender como está estructurada la rutina **`principal,`** es que controla los bigotes entre cada pulso que se envía a los servos, cuando el Boe-Bot se mueve hacia adelante. Observe que las instrucciones **`pulsout`** y **`pause`** en la rutina adelante no están anidadas dentro de un bucle **`for...next`**. Solamente se envía un único pulso a los servos. Después del pulso, el programa salta a la etiqueta **`principal:`** y controla los bigotes nuevamente. Este proceso se produce tan rápidamente que no es necesario reducir el valor de **`pause`** en la rutina adelante.

```
adelante:

  pulsout 12,650
  pulsout 13,850
  pause 20

goto principal
```

Cuando los bigotes detectan un objeto, el Boe-Bot ya está demasiado cerca del mismo. Para evadir el objeto, el Boe-Bot primero debe retroceder. Esto es fácil de solucionar con una subrutina. Por ejemplo, la rutina **`giro_izq`** que se muestra a continuación llama a la subrutina **`atras`** usando el comando **`gosub atras`**. La ventaja de las subrutinas es que regresan el control del programa a la línea inmediatamente posterior a la llamada de la subrutina. Así, después de ejecutar **`atras`**, se inicia el bucle **`for...next`** de la rutina **`giro_izq`**. Otra ventaja de tener una subrutina **`atras`** es que regresa el control del programa al lugar correcto. Si **`giro_der`** llama a la subrutina, en lugar de **`giro_izq,`** el control del programa es regresado a la rutina **`giro_der`**.

```
giro_izq:
    gosub atras
    for cuenta_pulsos = 0 to 30
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
  goto principal
```

La subrutina **`atras:`** es una versión modificada de la rutina **`atras:`** desarrollada en el Capítulo 2, Actividad 2. En lugar de finalizar con el comando **`goto principal`**, finaliza con **`return`**. El comando **`return`** trabaja correctamente en esta situación, debido a que regresa el control del programa a cualquiera de las tres rutinas que llamaron a la subrutina.

```
atras:
  for cuenta_pulsos = 0 to 60
    pulsout 12, 850
    pulsout 13, 650
    pause 20
  next
return
```

Su Turno

El argumento ***`final`*** de los bucles **`for...next`** de las rutinas **`giro_der`** y **`giro_izq`** puede ser ajustado para lograr giros mayores o menores y la rutina atras puede tener su valor ***`final`*** ajustado para que retroceda menos y poder navegar en lugares más reducidos.

- Experimente con distintos valores del argumento ***`final`*** de los bucles **`for...next`** del Programa 3.2.

Actividad 3: Controlando Entradas Múltiples como Números Binarios

Esta es otra forma de interpretar los Estados 1 a 4 tratados en la Actividad 2. En lugar de verlos como estados, podemos verlos como números binarios de 2-bits. Para aquellos que no conocen estos números, digamos que los números binarios 00, 01, 10 y 11 representan a los decimales a 0, 1, 2 y 3. La Tabla 3.2 muestra como los bigotes pueden generar estos números binarios.

Tabla 3.2: Estado de los Bigotes en Binario

Valor de Estado Binario	Valor de Estado Decimal	Acción de Salto Basada en el Valor de Estado
0000	0	**in6=0** y **in4=0** Ambos bigotes detectan un objeto, ejecuta **giro_u**
0001	1	**in6=0** y **in4=1** Bigote izquierdo detecta un objeto, ejecuta **giro_der**
0010	2	**in6=1** y **in4=0** Bigote derecho detecta un objeto, ejecuta **giro_izq**
0011	3	**in6=1** y **in4=1** Ningún bigote detecta un objeto, ejecuta **adelante**

Variante del Programa de Exploración con Bigotes

El Programa 3.3 hace lo mismo que el Programa 3.2. Sin embargo, procesa las entradas de los bigotes colocándolas en los bits de menor peso de una variable. Esta variable almacena un número binario que luego es usado por un comando branch (salto condicional) para saltar a la rutina de navegación apropiada.

```
' ¡Robótica! v1.5, Programa 3.3: Variante de Exploración con Bigotes.

'----- Declaración --------------

  cuenta_pulsos     var   byte                ' Contador del bucle for...next.
  estado            var   nib

'----- Inicialización ------------

  output 2                                    ' Fija a P2 como salida.
  freqout 2, 2000, 3000                       ' Indicador de reset.
  low 12                                      ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:

  control_bigotes:

    estado.bit1 = in6                         ' Crea un número binario con in6 en
    estado.bit0 = in4                         ' estado.bit1 e in4 en estado.bit0.

    ' Salto condicional a rutina de navegación basándose en el valor de estado.
```

```
    branch estado, [giro_u, giro_der, giro_izq, adelante]

    adelante:                                   ' Entrega un pulso adelante.
      pulsout 12,650
      pulsout 13,850
      pause 20

    goto principal

'----- Rutinas de navegación -------

  giro_izq:                                     ' Rutina de giro a la izquierda.
    gosub atras                                 ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
  goto principal

  giro_der:                                     ' Rutina giro derecha.
    gosub atras                                 ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 850
      pulsout 13, 850
      pause 20
    next
  goto principal

  giro_u:                                       ' Rutina de giro en U.
    gosub atras                                 ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 60
      pulsout 12, 850
      pulsout 13, 850
      pause 20
    next
  goto principal

'----- Subrutina de Navegación -----

  atras:                                        ' Usada por cada rutina de
    for cuenta_pulsos = 0 to 60                 ' navegación.
      pulsout 12, 850
      pulsout 13, 650
      pause 20
    next
  return
```

Cómo Funciona el Programa Variante de Exploración con Bigotes

- Busque el comando **branch** en el Apéndice C: Referencia Rápida de PBASIC o en el BASIC Stamp Manual (en Inglés) antes de continuar.

Una variable nibble (**nib**) llamada **estado** se agregó al sector de declaraciones del Programa 3.3. Un nibble (pedacito) consta de 4-bits y puede almacenar un número entre 0 (binario 0000) y 15 (binario 1111). Los dos bits de menor peso del nibble se usan para almacenar el estado de cada bigote. Dos bits pueden ser usados para contar de 0 a 3 (cuatro valores).

```
declaracion:
  cuenta_pulsos  var   byte
  estado         var   nib
```

La rutina **principal** contiene dos instrucciones seguidas por un comando **branch**. La primera instrucción iguala el valor del bit-1 de la variable **estado,** con el valor binario de **in6**. La segunda iguala el valor del bit-0 de la variable **estado** al valor de **in4**.

```
principal:

  control_bigotes:                          ' controla cada bigote
    estado.bit1 = in6
    estado.bit0 = in4

    branch estado, [giro_u, giro_der, giro_izq, adelante]
```

Su Turno

- Inserte este comando **debug** antes del comando **branch** en el Programa 3.3.

```
debug home, "binario ", bin4 estado, " = decimal ", dec1 estado
```

- Apoye el Boe-Bot sobre algún objeto cuando se ejecuta el programa, para evitar que las ruedas toquen cualquier superficie. De esta forma podrá dejar conectado el Boe-Bot al cable serial, mientras ejecuta el programa.

- Ejecute el programa y observe la ventana Debug Terminal a medida que prueba los bigotes.

Actividad 4: Inteligencia Artificial. Decidir Cuándo Está Atrapado

Tal vez haya notado que el Boe-Bot tiende a quedar atrapado en las esquinas. A medida que el Boe-Bot se acerca a una esquina, supongamos que su bigote toca la pared de la izquierda, entonces dobla a la derecha. Cuando el Boe-Bot se vuelve a mover hacia adelante, su bigote derecho tocará la otra pared, así que girará a la izquierda. Luego tocará la pared izquierda y volverá a girar a la derecha, donde se encontrará con la otra pared y así sucesivamente, hasta que alguien lo rescate de este predicamento.

Programa Para Liberarse de las Esquinas

El problema de los rincones puede solucionarse agregando un contador que registre las transiciones entre los estados 1 y 2 en el Programa 3.3. El Programa 3.4 que aparece abajo, es una versión modificada del Programa 3.3, que detecta y cuenta la cantidad de veces que los bigotes detectan un obstáculo después de otro.

```
' ¡Robótica! v1.5, Programa 3.4: Escape de Rincones.

'----- Declaración -------

  cuenta_pulsos  var   byte             ' Contador del bucle for...next.
  estado         var   nib              ' Almacena datos de bigotes en binario.
  estado_ant     var   nib              ' Guarda estado del pulso anterior.
  contador       var   nib              ' Contador de secuencia de eventos.

'----- Inicialización -----

  output 2                              ' Fija a P2 como salida.
  freqout 2, 2000, 3000                 ' Indicador de reset.
  low 12                                ' P12 y 13 salidas bajas.
  low 13
  estado_ant = %0001                    ' Inicializa estado_ant.

'----- Rutina Principal -------

principal:

  control_bigotes:

    estado.bit1 = in6                   ' Almacena bigotes en estado.
    estado.bit0 = in4

    ' Más adelante en el programa, un contador se incrementa cada vez que se pre-
    ' sionan bigotes alternadamente.  La instrucción if...then controla que no se
    ' trate del mismo bigote.  Si es así, reinicia la variable contador.
```

```
    if estado_ant <> estado then sin_reset     ' Si repite mismo bigote, reset.
    contador = 0                               ' Pone a cero el contador.

    sin_reset:                                 ' Rótulo para que el if... salte.

    ' Si estado_ant o-exclusiva estado no es igual al binario-0011, salta al rótu-
    ' lo continuar:.  De otra forma incrementa el contador y decide si es hora de
    ' dar un giro en U.

    if estado_ant ^ estado <> %0011 then continuar

      contador = contador +1
      estado_ant = estado                      ' Guarda una copia de estado.
      if contador = 4 then giro_u              ' Rutina de escape de esquina.


    continuar:             ' Salto condicional según el valor de estado.

      branch estado, [giro_u, giro_der, giro_izq, adelante]

    adelante:
      pulsout 12,650
      pulsout 13,850
      pause 20

    goto principal

'----- Rutinas de Navegación -------

  giro_izq:                                    ' Rutina de giro a la izquierda.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
  goto principal

  giro_der:                                    ' Rutina giro derecha.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 850
      pulsout 13, 850
      pause 20
    next
  goto principal
```

```
  giro_u:                                          ' Rutina de giro en U.
    gosub atras                                    ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 60
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
    contador = 0                                    ' Pone a cero el contador.
  goto principal

'----- Subrutina de Navegación -----

  atras:                                           ' Usada por cada rutina de
    for cuenta_pulsos = 0 to 60                     ' navegación.
      pulsout 12, 850
      pulsout 13, 650
      pause 20
    next
  return
```

Cómo Funciona el Programa Escape de Rincones

Las variables tipo nibble llamadas **`estado_ant`** y **`contador`** se agregaron a la sección de **`declaraciones`** para mantener un registro de los giros. La variable **`estado_ant`** guarda el estado previo antes de que la variable **`estado`** sea actualizada con los valores actuales de los bigotes. La variable **`contador`** registra la cantidad de transiciones de **`estado`** de uno a dos y viceversa.

```
      declaraciones:
        cuenta_pulsos  var   byte
        estado         var   nib
        estado_ant     var   nib
        contador       var   nib
```

La rutina **`control_bigotes`** está diseñada para controlar los bigotes como antes, pero se ha agregado código para detectar la condición "atrapado en una esquina". Esta condición es indicada por cuatro transiciones derecha-izquierda-derecha (o izquierda-derecha-izquierda). Antes de actualizar la variable **`estado`** con los valores nuevos de los bigotes, la instrucción **`estado_ant = estado`** guarda una copia del estado previo, como referencia. Luego la variable **`estado`** es actualizada con los valores actuales, como en el Programa 3.3.

```
        control_bigotes:
          estado_ant = estado
```

¿Qué es...

O-Exclusiva (XOR)

El término "o-exclusiva" (XOR) se refiere a una de las operaciones del Álgebra Booleana, que es usada con expresiones y números binarios. Esta tabla muestra el uso de XOR:

```
0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0
```

El operador en PBASIC para la operación XOR es el caracter ^. Cuando tiene una expresión como variable_1 ^ variable 2, significa que cada bit de la variable_1 se comparará mediante el operador "o-exclusiva" con el bit correspondiente de la variable_2. En otras palabras, el bit-0 de la variable_1 se compara con el bit_0 de variable_2. El bit-1 de la variable_1 se compara con el bit-1 de la variable_2 y así sucesivamente.

El operador NOT es introducido en la sección de Proyectos y Preguntas de este capítulo. El BASIC Stamp Manual muestra más operadores Booleanos en la sección del mismo nombre. Se realiza una explicación detallada en la sección que explica el comando **`if...then`**.

```
    estado.bit1 = in6
    estado.bit0 = in4
```

Asumimos que la condición para que esté atrapado en la esquina, es que se presionen en forma alternada los bigotes, en varias oportunidades. Sin embargo, si el Boe-Bot encuentra un obstáculo dos veces seguidas con el mismo bigote, es porque no debe estar atrapado en una esquina. Si esto sucede, el programa debería empezar a contar desde cero nuevamente. El comando **`estado_ant <> estado then sin_reset`** verifica si el mismo bigote fue presionado dos veces seguidas. Si es así, reinicia el contador, caso contrario, el programa salta a la etiqueta **`sin_reset:`**.

```
    if estado_ant <> estado then sin_reset
    contador = 0

    sin_reset:
```

Luego, el programa controla si los bigotes se han presionado en forma alternada mediante el operador o-exclusiva (**^**). Si el resultado de **`estado_ant`** o-exclusiva **`estado`** nuevo, no es igual al número binario 0011, significa que el sensor opuesto no fue tocado. En este caso, la instrucción **`if...then`** envía el programa a la etiqueta **`continuar:`**.

```
    if estado_ant ^ estado <> %0011 then continuar
```

Por otro lado, si la comparación es falsa (si **`estado_ant`** o-exclusiva **`estado`** es igual a 0011), entonces se incrementa la variable **`contador`**. Otro **`if...then`** controla si **`contador`** ha llegado a 4. Si es así, se ejecuta la rutina **`giro_u`**. Caso contrario el programa salta a la maniobra apropiada basándose en el valor de **`estado`**.

```
controla_contador:
 contador = contador +1
 if contador = 4 then giro_u

continuar:
  branch estado, [giro_u, giro_der, giro_izq, adelante]
```

Cuando el contador de contactos en bigotes opuestos llega a 4, se llama la rutina **giro_u**. Esta rutina ha sido modificada para que reinicie el contador cada vez que es llamada. De esa forma, el Boe-Bot comienza a contar de 0 cada vez que se escapa de una esquina.

```
giro_u:
  gosub atras               ' Llama a atras: antes de girar.
  for cuenta_pulsos = 0 to 60
    pulsout 12, 650
    pulsout 13, 650
    pause 20
  next
  contador = 0
goto principal
```

Su Turno

Una de las instrucciones **if...then** del Programa 3.4 controla si el contador ha llegado a 4. Pruebe reducir e incrementar el argumento ***condición*** de esta instrucción **if...then y** observe su efecto. También observe si reducir el valor tiene algún efecto sobre el comportamiento en la navegación normal. Reducirlo a 0, 1, o 2 podría causar problemas interesantes.

Sumario y Aplicaciones

En este capítulo, en vez de ejecutar una lista preprogramada, el Boe-Bot se programó para explorar basándose en las entradas de los sensores. En este caso, los sensores de entrada fueron los bigotes. El BASIC Stamp fue programado para controlar estos sensores y mostrar los resultados por dos medios, la Debug Terminal y los LEDs.

Cuando se conecta todo apropiadamente, estos interruptores pueden mostrar una tensión (5 V) cuando están abiertos y una tensión diferente (0 V) cuando están cerrados. Tensiones de 5 y 0 V corresponden a niveles de lógica transistor-transistor (TTL) y las entradas del BASIC Stamp interpretan estos niveles como "1" y "0," respectivamente.
Los programas PBASIC se desarrollaron para lograr que el BASIC Stamp controle los bigotes entre cada pulso de los servos. Basándose en el estado de los bigotes, las rutinas de **`principal:`** harán que el Boe-Bot continúe adelante o llamarán a rutinas de navegación, que fueron desarrolladas en el capítulo anterior, para que el Boe-Bot evite los obstáculos. Como ejemplo de inteligencia artificial, se desarrolló un programa que logró que el Boe-Bot detectara cuándo se hallaba atorado en una esquina. En este caso, la tarea fue realizada por expresiones de lógica Booleana con o-exclusiva XOR y un contador.

Ejemplo del Mundo Real

Es común encontrar sensores automatizados a nuestro alrededor. Cuando va a un supermercado, los sensores se encargan de hacer abrir la puerta por usted. Los microcontroladores controlan teclados de forma similar a como los programas del BASIC Stamp controlan los bigotes, para detectar si han sido presionados o no. La información es procesada y como resultado produce una salida. En el caso de una puerta automática, el resultado es una puerta controlada por motor o servo que es abierta.
La maquinaria robótica de muchas formas y tamaños, también incluye una gran variedad de interruptores táctiles, conectados de manera similar a los bigotes. Los brazos robóticos usan a menudo estos interruptores para detectar cuándo han encontrado el objeto que están programados para manipular. Las fábricas usan estos interruptores para contar objetos en una línea de producción y también para alinear componentes para procesos industriales. En todos estos casos, los interruptores proveen las entradas que generan algún tipo de respuesta programada. Sea una calculadora, robot o una línea de producción, los interruptores de entrada son monitoreados electrónicamente. Basándose en el estado de los interruptores, la calculadora muestra resultados, el brazo robótico toma un objeto o la línea de producción de una fábrica actúa sobre motores y servos, para guiar la fabricación de un producto en una forma preprogramada.

Aplicaciones para el Boe-Bot

Este capítulo introdujo la exploración del Boe-Bot basándose en sensores. Los siguientes tres capítulos se enfocarán en el uso de diferentes tipos de sensores para darle visión al Boe-Bot. La visión y el tacto abren muchas posibilidades para la exploración en ambientes complejos.

Preguntas y Proyectos

Preguntas

1. ¿Qué tipo de conexión eléctrica es un bigote?

2. ¿Cuáles son los tres tipos de registros de E/S del BASIC Stamp ?

3. Si un pin de E/S se configura como salida, ¿qué registro logra esto?

4. Cuando se presiona un bigote, ¿qué tensión se produce en el pin de E/S que lo está monitoreando? ¿Qué valor binario se presentará en el registro de entrada? Si se usa P8 para controlar el pin de entrada, ¿qué valor tiene **`in8`** cuando se presiona un bigote y cuál será si no se presiona?

5. ¿Qué dirección debe tener un pin de E/S para poder hacer funcionar a un LED?

6. ¿A qué bigote está conectada **`in6`**? ¿Y que hay sobre **`in4`**?

7. ¿Qué comandos puede usar para reemplazar varias instrucciones **`if...then`**?

8. ¿Cuáles son los tres elementos de programación PBASIC que fueron usados en este capítulo para registrar los eventos y tomar las decisiones correspondientes?

Ejercicios

1. Suponga que el Programa 3.1 toma 1 ms para mostrar cada caracter y otro ms para ejecutar el bucle. ¿Cuál es la velocidad de muestreo a la que el BASIC Stamp controla cada bigote?

2. En el Programa 3.2, determine la velocidad de muestreo cuando el Boe-Bot se mueve hacia adelante. También determine la velocidad de muestreo mientras el Boe-Bot está realizando maniobras. Pista: la segunda parte de este ejercicio es una "pregunta tramposa".

3. Suponga que la etiqueta de la rutina **`giro_izq`** fuese cambiada por **`izq_giro`**. Escriba todos los cambios necesarios a los comandos de los Programas 3.2, 3.3 y 3.4.

4. Describa la pantalla **`debug`** después de la modificación de la sección Su Turno de la Actividad 3.

Proyectos

1. El operador XOR (o-exclusiva) fue introducido como operador Booleano en este capítulo. El operador NOT de PBASIC es otro operador Booleano que se representa con el caracter "~". En los teclados de computación ingleses se escriben presionando Shift y la tecla que está debajo de Esc. En los teclados españoles deberá tener presionada la tecla Alt y teclear el número 126. Así trabaja NOT:

   ```
   NOT  0  =  1
   NOT  1  =  0
   ```

   En la sección Su Turno de la Actividad 1, usó LEDs para mostrar el estado de los bigotes. Cuando lo hizo, los LEDs se encendían cada vez que se presionaban los bigotes. Use el operador PBASIC NOT (~) para lograr que cada LED esté encendido mientras el bigote no esté presionado. El LED debería apagarse cuando se presiona el bigote.

   Modifique aún más el programa, de forma que el LED parpadee a 2 Hz cuando se presiona el bigote. Pista, necesitará usar un bucle e instrucciones **`pause`**. Pruebe tres modificaciones más al Programa 3.1 y ejecute el programa en cada paso. Haga las modificaciones para que el LED titile a 4 Hz, 10 Hz y 100 Hz cuando el bigote es presionado.

2. Modifique el Programa 3.2 de forma que el Boe-Bot se mueva hacia atrás lentamente mientras ambos bigotes estén presionados. Caso contrario, permanecerá estático. Modifique aún más el programa, de forma que el Boe-Bot rote en sentido antihorario cuando se presiona el bigote izquierdo y en sentido horario cuando se presione el derecho. Cuando el programa esté terminado, ajuste la velocidad de respuesta de forma que parezca que usted está empujando al Boe-Bot desde sus bigotes.

3. Desafío: Modifique el Programa 3.4 para que el Boe-Bot viaje en un patrón circular. Modifíquelo de tal forma que si toca el bigote interno, la trayectoria circular reduzca 5 cm su diámetro y si toca el bigote externo del Boe-Bot su trayectoria circular aumente el diámetro en 5 cm.
   Cuando pueda realizar el control sobre el diámetro del circulo con los bigotes, programe el Boe-Bot para que recuerde este diámetro, incluso después de un reset. El comando **`write`** puede ser usado para almacenar datos en la EEPROM (no volátil). Mientras que la RAM del BASIC Stamp se borra con cada reset (volátil), la EEPROM puede almacenar los datos para que sean usados la próxima vez que el BASIC Stamp recupere la alimentación. El formato del comando **`write`** es:

   ```
   write dirección, datos
   ```

Debido a que no usamos ninguna instrucción **`data`** es este programa, la dirección 0 de la EEPROM puede ser usada para almacenar el valor de un parámetro PBASIC que determine el diámetro de la circunferencia.

Capítulo 4: Navegación Sensible a la Luz con Fotorresitores

Su BOE-Bot, ¿es Fotófilo o Fotofóbico?

Los Fotorresitores de su kit pueden ser usados para que su Boe-Bot detecte variaciones en el nivel de luz. Con un poco de programación, su Boe-Bot puede ser transformado en fotófilo (una criatura que es atraída por la luz), o fotofóbico (una criatura que evita la luz).

Para medir la presencia e intensidad de luz, construirá un par de circuitos con Fotorresitores en su Boe-Bot. Un fotorresitor es un resistor cuyo valor depende de la intensidad de luz (light-dependent resistor o LDR), que cubre un espectro similar al del ojo humano. Los elementos activos de estos Fotorresitores están hechos de Sulfuro de Cadmio (CdS). La luz entra en la capa semiconductora que está aplicada sobre un sustrato cerámico y produce portadores de carga libres. Se produce una resistencia eléctrica definida, que es inversamente proporcional a la intensidad de iluminación. En otras palabras, la oscuridad produce valores altos de resistencia y la claridad produce valores pequeños.

Los Fotorresitores incluidos en el kit del Boe-Bot son EG&G Vactec (#VT935G). Si necesita más Fotorresitores, puede obtenerlos en Parallax, así como también en cualquier negocio de venta de componentes electrónicos. Vea el Apéndice A: Boe-Bot Lista de Componentes y Suministros. Las especificaciones de estos Fotorresitores se muestran e la Figura 4.1:

Figure 3.1: Especificaciones del Fotoresistor EG&G Vactec (en Inglés)

Photoresistor Specifications

Resistance (Ohms)					Peak Spectral Response nm	V_{MAX}	Response Time @ 1 fc (ms, typ.)	
10 Lux 2850K			Dark					
Min	Typ.	Max.	Min.	Sec.			Rise (1-1/e)	Fall (1/e)
20K	29.0K	38K	1M	10	550	100	35	5

Luminancia es el nombre específico de la medición de luz incidente. La unidad de medición de luminancia es el “pie-candela” (foot-candle) en el sistema Inglés y el "lux" en el sistema métrico. Mientras usamos fotorresistores no necesitamos conocer los niveles de “luxs”, sino simplemente si la luminancia es mayor o menor en ciertas direcciones. Basándose en estos datos el Boe-Bot girará hacia la luz. Para más información sobre medición de la luz con un Microcontrolador, eche una mirada al Experimento 4 de Mediciones Ambientales.

Actividad 1: Construcción y Prueba de Ojos Fotosensibles

Componentes

La Figura 4.2 muestra los componentes introducidos en este experimento, junto con sus símbolos esquemáticos. Debajo hay una lista de los componentes que necesitará. Todos los componentes son no polarizados, lo que quiere decir que los terminales 1 y 2 pueden ser invertidos sin afectar el circuito.

(1) piezoeléctrico
(2) fotorresistores
(2) capacitores 0.1 uF
(2) capacitores 0.01 uF
(2) resistores 220 Ohm
(varios) cables de interconexión

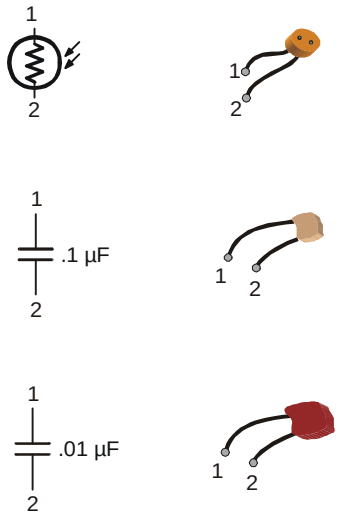
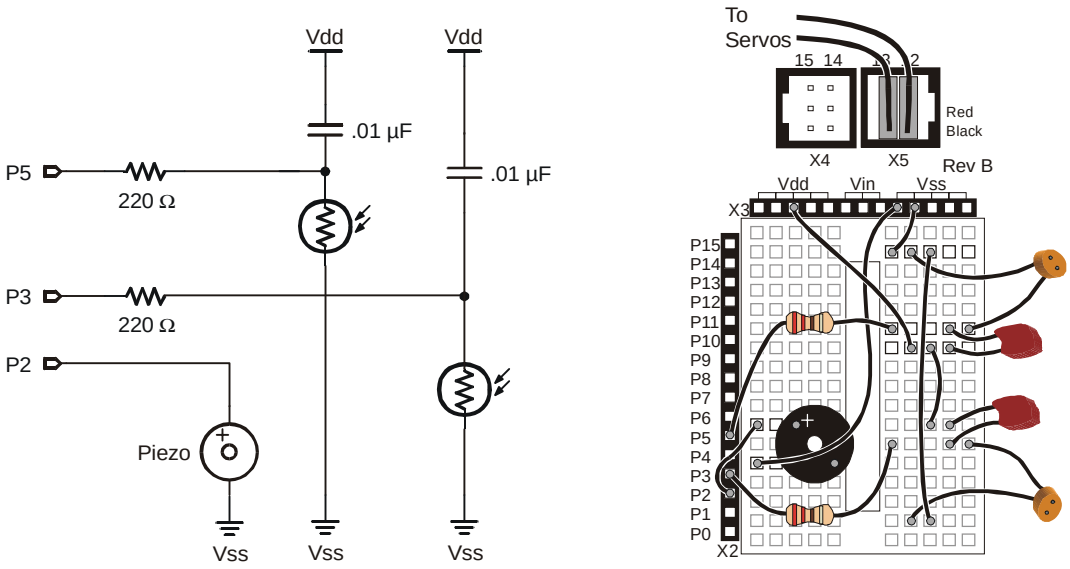


Figura 4.2: Símbolos y componentes para armar el circuito.

¡Constrúyalo!

La Figura 4.3 muestra (a) el circuito resistor / capacitor (RC) para cada fotorresitor y (b) un ejemplo de armado sobre la protoboard. Un fotorresitor es un dispositivo analógico. Su valor cambia en forma continua de acuerdo a la luminancia, que es otro valor analógico. La resistencia del fotorresitor es muy baja cuando se expone directamente a la luz del sol. A medida que desciende el nivel de luz, su resistencia aumenta. En completa oscuridad, la resistencia del fotorresitor puede alcanzar valores de hasta 1 MΩ. Aunque el fotorresitor es analógico, esto no quiere decir que sea lineal. Esto significa que si la fuente de luz (luminancia) varía a una proporción constante, el valor del fotorresitor no necesariamente variará a una proporción constante.



(a) **(b)**

Figura 4.3: (a) Dos circuitos RC para medir resistores que varían con la luz y (b) ejemplo de distribución en la protoboard.

> ✓ **TIP** Recuerde: Los circuitos de los servos no se muestran en los esquemas, pero aún se los puede ver en los diagramas de la protoboard. Todas las actividades a partir del Capítulo 2, fueron diseñadas de forma que los servos quedaran conectados en los puertos para servos 12 y 13.

Programa para Medir la Resistencia

El circuito de la Figura 4.3 (a) fue diseñado para ser usado con el comando **`rctime`**. Este comando puede ser usado con un circuito RC donde uno de los valores, R o C, varía, mientras el otro permanece constante. El comando **`rctime`** puede medir valores variables debido a que toma ventaja de las variaciones de tiempo que generan los circuitos RC. El tiempo que demora un circuito RC en variar una tensión depende de $R \times C$, llamada constante de tiempo RC. La constante de tiempo RC a menudo es representada por la letra griega Tau (τ).

Para uno de los circuitos RC mostrados en la Figura 4.3 (a), el primer paso para tomar una lectura con **`rctime`** es descargar el capacitor, colocando 5 V en la placa inferior. Si configuramos como salida en estado alto al pin de E/S correspondiente, en pocos ms lograremos esto. Luego, el comando **`rctime`** puede ser usado para medir el tiempo que le lleva al capacitor cargarse, hasta que la tensión en el pin cambie de 5 a 1.4 V. ¿Por qué 1.4 V? Debido a que este es el umbral de tensión de los pines del BASIC Stamp. Cuando la tensión de un pin es mayor de 1.4 V, el valor del bit de registro de entrada conectado a ese pin de E/S es "1". Cuando la tensión cae por debajo de 1.4 V, el valor del bit de registro de entrada es "0."

El comando **`rctime`** del circuito que se muestra en la Figura 4.3 mide cuánto tiempo demora la tensión de la placa inferior de capacitor en bajar de 5 a 1.4 V. Este tiempo varía con la fórmula:

$$\frac{t}{R \times C} = \ln(\frac{V_{inicial}}{V_{final}})$$

$$\frac{t}{R \times 0.01 \times 10^{-6}} = \ln(\frac{5\,V}{1.4\,V}) \quad s$$

$$t = \ln(3.57) \times R \times 0.01 \times 10^{-6} \quad s$$

$$t = 1.27 \times 10^{-8} \times R \quad s \qquad (4.1)$$

La Ecuación 4.1 indica que el tiempo que tarda en caer la tensión de la placa inferior de uno de los capacitores de los circuitos RC de la Figura 4.3 (a), es directamente proporcional a la resistencia del fotorresitor. Dado que la resistencia varía con la luminancia (exposición a niveles variados de luz), así también lo hace el tiempo. Midiendo este tiempo, se puede inducir la exposición de luz relativa.

El comando **`rctime`** cambia la dirección de un pin de E/S, de salida a entrada. Tan pronto como el pin se convierte en entrada, la tensión de la placa inferior del capacitor empieza a caer, de acuerdo a la ecuación temporal anterior. El BASIC Stamp comienza a contar en incrementos de 2 μs, hasta que el nivel de tensión de la placa inferior del capacitor cae por debajo de 1.4 V.

✓ **TIP**	Para obtener mejores resultados: Elimine la luz directa del sol; es demasiado brillante para los circuitos de los fotorresitores.

- Ejecute el Programa 4.1. Éste demuestra cómo usar el comando **`rctime`** para leer fotorresitores.
- Este programa usa la Debug Terminal, así que deberá dejar conectado el cable serial a la BOE mientras se esté ejecutando el Programa 4.1.


```
' ¡Robótica! Programa 4.1: Fotorresitores y rctime

'----- Declaración --------------

  foto_izq   var word                          ' Almacenan los tiempos RC de los
  foto_der  var word                           ' fotorresitores izq. y derecho.

'----- Inicialización ------------

  debug cls                                    ' Abre y limpia la Debug Terminal.

'----- Rutina Principal --------------

principal:

  ' Mide el tiempo RC del fotorresitor derecho.

  high 3                                       ' Fija a P3 como salida alta.
  pause 3                                      ' Pausa de 3 ms.
  rctime 3,1,foto_der                          ' Mide tiempo RC en P3.

  ' Mide el tiempo RC del fotorresitor izquierdo.

  high 5                                       ' Fija a P5 como salida alta.
  pause 3                                      ' Pausa de 3 ms.
  rctime 5,1,foto_izq                          ' Mide tiempo RC en P5.

  ' Muestra las mediciones de tiempo RC usando la Debug Terminal.

  debug home, "I  ", dec5 foto_izq, "  D  ", dec5 foto_der

goto principal
```

Cómo Funciona Fotorresitores y rctime

Dos variables tipo word, **`foto_izq`** y **`foto_der`** se declaran para almacenar los valores de los tiempos RC de los fotorresitores izquierdo y derecho. La rutina **`principal`** mide y muestra los valores para cada circuito RC. El código para leer el circuito RC derecho se muestra a continuación. Primero, el pin de E/S P3 se fija

como salida en estado alto. Luego, una pausa de 3 ms le permite al capacitor descargarse. Luego de los 3 ms, la placa inferior del capacitor está suficientemente cerca de 5 V y está lista para la medición de **`rctime`**. El comando **`rctime`** mide el tiempo RC en el pin P3, comenzando en el estado "1" (5 V) y almacena el resultado en la variable **`foto_der`**. Recuerde, el valor almacenado en **`foto_der`** es un número. Este número le dice cuántos incrementos de 2 µs pasaron antes que la tensión de la placa inferior del capacitor cayera por debajo del umbral de 1.4 V.

```
high 3
pause 3
rctime 3,1,foto_der
```

Su Turno

- Reemplace uno de los capacitores de 0.01 µF con uno de 0.1 µF. ¿Qué circuito se comporta mejor bajo luz brillante, el del capacitor grande (0.1 µF) o el del pequeño (0.01 µF)? ¿Cuál es el efecto a medida que los alrededores se vuelven más oscuros? ¿Observa algún síntoma que le indique qué capacitor trabajaría mejor en un ambiente oscuro?

- Asegúrese de retornar el circuito a su estado original antes de pasar a la siguiente actividad.

Actividad 2: Un Compás de Luz

Si enfoca un haz de luz en el frente del Boe-Bot, el circuito y las técnicas de programación recién discutidas pueden ser usadas para que el Boe-Bot gire y se dirija hacia la luz. Asegúrese de que los fotorresitores estén colocados de forma que puedan realizar una comparación de luz incidente. Deberían estar a un ángulo de 45° de la línea central del Boe-Bot hacia afuera y a 45° por debajo de la horizontal. En otras palabras, apunte las caras de los fotorresitores hacia la mesa. Luego, use una luz brillante para hacer que el Boe-Bot siga la dirección de la luz.

Programe el Boe-Bot para que Apunte hacia la Luz

Para lograr que el Boe-Bot siga una fuente de luz, compararemos en el programa los valores medidos en cada fotorresitor. Recuerde que a medida que la intensidad de luz se debilita, el valor del fotorresitor aumenta. Así, si el valor del fotorresitor de la derecha es mayor que el de la izquierda, significa que hay más claridad a la izquierda. Dada esta situación, el Boe-Bot debería girar a la izquierda. Por otro lado, si **`rctime`** del fotorresitor de la izquierda es mayor que el de la derecha, el lado derecho es el más luminoso, así que el Boe-Bot debería girar a la derecha.

Para evitar que el Boe-Bot cambie de dirección muy seguido, se introduce un intervalo (deadband) en el que no debería haber intentos de corrección. Si los números van por encima o por debajo de este intervalo, el sistema efectuará las correcciones necesarias. La forma más conveniente de medir este intervalo es restar el **`rctime`** izquierdo del **`rctime`** derecho, o viceversa y luego tomar el valor absoluto. Si este valor absoluto está dentro de los límites del intervalo, no se generará respuesta; caso contrario, programe el ajuste apropiado.

- Ingrese y ejecute el Programa 4.2.
- Encienda una luz brillante en el frente del Boe-Bot. Cuando mueva la luz, el Boe-Bot debería rotar para quedar apuntando a la fuente de luz.
- En lugar de usar una linterna, use su mano o una hoja para provocar una sombra en uno de los fotorresitores. El Boe-Bot debería rotar para alejarse de la sombra.

```
' ¡Robótica! v1.5, Programa 4.2: Compás de Luz

'----- Declaración --------------

  foto_izq   var word                          ' Almacenan los tiempos RC de los
  foto_der  var word                           ' fotorresitores izq. y derecho.

'----- Inicialización ------------

  output 2                                     ' Fija a P2 como salida.
  freqout 2, 2000, 3000                        ' Indicador de reset.
  low 12                                       ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:

  ' Mide el tiempo RC del fotorresitor derecho.

    high 3                                     ' Fija a P3 como salida alta.
    pause 3                                    ' Pausa de 3 ms.
    rctime 3,1,foto_der                        ' Mide tiempo RC en P3.

  ' Mide el tiempo RC del fotorresitor izquierdo.

    high 5                                     ' Fija a P5 como salida alta.
    pause 3                                    ' Pausa de 3 ms.
    rctime 5,1,foto_izq                        ' Mide tiempo RC en P5.
```

```
  ' Toma la diferencia entre foto_der y foto_izq, luego decide qué hacer.

    if abs(foto_izq-foto_der) < 2 then principal
    if foto_izq > foto_der then pulso_der
    if foto_izq < foto_der then pulso_izq

'----- Rutinas de Navegación -------

pulso_izq:                                  ' Aplica un pulso a la izq., luego
  pulsout 12, 650
  pulsout 13, 650
  pause 10
goto principal                              ' regresa a la rutina principal.

pulso_der:                                  ' Aplica un pulso a la der., luego

  pulsout 12, 850
  pulsout 13, 850
  pause 10
goto principal                              ' regresa a la rutina principal.
```

Cómo Trabaja el Compás de Luz

El Programa 4.2 mide los tiempos RC y controla si la diferencia entre los valores obtenidos por los comandos **`rctime`** están dentro del intervalo de tolerancia (en el que no toma ninguna acción):

```
      if abs(foto_izq - foto_der) < 2 then principal
```

Si la diferencia está dentro del intervalo, el programa salta a la etiqueta **`principal:`**. Si la diferencia medida supera el valor del intervalo, dos instrucciones **`if...then`** deciden a qué rutina llamar, **`pulso_izq`** o **`pulso_der`**.

```
        if foto_izq > foto_der then pulso_der
        if foto_izq < foto_der then pulso_izq
```

La rutina **`pulso_izq`** se muestra a continuación. Observe que los comandos **`pulsout`** y el comando **`pause`** no están anidados dentro del bucle **`for...next`**. En lugar de eso, se emite un único pulso con una pausa un poco más pequeña de lo normal y luego el control regresa a la rutina **`principal`**. Esto le permite al programa controlar y actualizar los valores de los fotorresitores entre pulsos de los servos. Observe que la pausa no es de 20 ms. Esto es debido a que cada comando **`rctime`** lleva cierta cantidad de tiempo, que puede restarse de la pausa. Para las condiciones de luz usadas en este ejemplo, el promedio de las pausas causadas por **`rctime`** fue de 10 ms. Sus condiciones probablemente sean diferentes.

```
pulso_izq:
  pulsout 12, 650
  pulsout 13, 650
  pause 10
goto principal
```


Su Turno

En un sector oscuro, los valores de los fotorresitores no solamente serán mayores, sino que además aumentará la diferencia entre ambos. Tal vez deba incrementar el intervalo (deadband) para luz ambiente escasa, para que el Boe-Bot no responda a pequeñas variaciones en la intensidad de la luz. A menores niveles de luz, menores deberán ser las instrucciones **`pause`**. Si el rendimiento del Boe-Bot comienza a decrecer, probablemente sea debido a que el tiempo entre pulsos ha excedido los 40 ms. La primer línea de defensa para este problema es reducir el tiempo de **`pause`** en cada subrutina a cero. La segunda línea de defensa es controlar los fotorresitores de a uno por vez, en forma alternada. De esa forma, después del primer pulso, el fotorresitor derecho podría ser controlado. Luego, después del segundo pulso, se podría revisar el fotorresitor izquierdo. Puede probar sus habilidades de programación desarrollando un código que haga esto en la sección de Proyectos de este capítulo.

El valor actual de la deadband (intervalo) es "2" en la expresión:

```
if abs(foto_izq-foto_der) < 2 then principal
```

- Experimente con diferentes niveles de luz ambiente y su efecto en la deadband (intervalo) en áreas luminosas y oscuras. En zonas luminosas, el valor de la deadband puede reducirse, incluso puede ser cero. En zonas oscuras, el valor de la deadband debería incrementarse.

- Invierta las condiciones de la segunda y tercera instrucción **`if...then`** del Programa 4.2. luego vuelva a ejecutar el programa. Ahora su Boe-Bot apunta hacia la oscuridad.

Actividad 3: Seguir la Luz

Simplemente agregando movimiento hacia adelante a su Boe-Bot, puede convertirlo en un robot seguidor de luz, un fotófilo. Un experimento interesante es tratar de programar el Boe-Bot para que se mueva hacia adelante y busque la luz. Luego, colocarlo en una habitación oscura con la puerta abierta hacia un ambiente iluminado. Asumiendo que no haya obstáculos en su camino, el Boe-Bot se dirigirá a la puerta y saldrá de la habitación oscura.

Programa para Seguir la Luz

Programar el Boe-Bot para que siga la luz, solamente requiere unas pocas modificaciones al Programa 4.2. La principal modificación es que una medición que caía dentro de la deadband, daba como resultado cero movimiento en el Programa 4.2. En el Programa 4.3, cuando las diferencias en rctime caen dentro de la deadband, dan como resultado movimiento hacia adelante. Veamos cómo se hace.

```
' ¡Robótica! v1.4, Programa 4.3: Seguidor de luz.

'----- Declaración --------------

  foto_izq   var word                          ' Almacenan los tiempos RC de los
  foto_der  var word                           ' fotorresitores izq. y derecho.

'----- Inicialización ------------

  output 2                                     ' Fija a P2 como salida.
  freqout 2, 2000, 3000                        ' Indicador de reset.
  low 12                                       ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:

  ' Mide el tiempo RC del fotorresitor derecho.

  high 3                                       ' Fija a P3 como salida alta.
  pause 3                                      ' Pausa de 3 ms.
  rctime 3,1,foto_der                          ' Mide tiempo RC en P3.

  ' Mide el tiempo RC del fotorresitor izquierdo.

    high 5                                     ' Fija a P5 como salida alta.
    pause 3                                    ' Pausa de 3 ms.
    rctime 5,1,foto_izq                        ' Mide tiempo RC en P5.

  ' Controla si la diferencia entre los tiempos RC cae dentro de la deadband,
  ' 2 en este caso. Si es así, adelante.  Sino a la subrutina control_dir.

  if abs(foto_izq-foto_der) > 2 then control_dir

  adelante_pulso:
```

```
    pulsout 12, 650
    pulsout 13, 850
    pause 20
  goto principal

  ' Salta hacia giro_der o giro_izq dependiendo del mayor tiempo RC.

  control_dir:
    if foto_izq > foto_der then pulso_der
    if foto_izq < foto_der then pulso_izq

'----- Rutinas de Navegación -------

pulso_izq:                                  ' Aplica un pulso a la izq., luego
  pulsout 12, 650
  pulsout 13, 650
  pause 20
goto principal                              ' regresa a la rutina principal.

pulso_der:                                  ' Aplica un pulso a la der., luego

  pulsout 12, 850
  pulsout 13, 850
  pause 20
goto principal                              ' regresa a la rutina principal.
```

Cómo Funciona el Programa Seguidor de Luz

Como en el programa anterior, la primer instrucción **`if...then`** controla si la diferencia en las mediciones del tiempo RC cae dentro de la deadband. Esta instrucción ha sido modificada para que saltee la rutina **`adelante_pulso`** si la diferencia de tiempos RC cae fuera de la deadband. Por otro lado, si la diferencia del tiempo RC cae en la deadband, se ejecuta un pulso adelante. Luego, el programa vuelve a **`principal`** y controla nuevamente los tiempos RC.

```
if abs(foto_izq-foto_der) > 2 then control_dir

  adelante_pulso:
      pulsout 12, 650
      pulsout 13, 850
      pause 20
  goto principal
```

Si la diferencia entre tiempos RC no cae dentro de la deadband, el programa salta a la etiqueta **`control_dir`**. Las instrucciones **`if...then`** que están a continuación de la etiqueta **`control_dir`** se usan

para decidir si se aplicará un pulso a la izquierda o un pulso a la derecha, dependiendo de la diferencia entre los valores **foto_der** y **foto_izq**. De esta forma, el programa aplica un pulso hacia adelante o un pulso de giro, cada vez que se controlan los fotorresitores.

```
control_dir:
  if foto_izq > foto_der then pulso_der
  if foto_izq < foto_der then pulso_izq
```

Su Turno

- Repita el ejercicio de la sección Su Turno anterior. Puede guiar a su Boe-Bot con una linterna.

- En lugar de apuntar los fotorresitores hacia la superficie al frente del Boe-Bot, apúntelos hacia arriba como en la Figura 4.3. Con los fotorresitores configurados de esta forma, el Boe-Bot se moverá por el piso tratando siempre de mantenerse en el lugar más iluminado.

Dependiendo del gradiente (variación) de luminancia, tal vez deba incrementar la deadband, para suavizar el comportamiento del Boe-Bot. Por el contrario, puede ser necesario reducir la deadband para aumentar la respuesta a los cambios de luz.

Actividad 4: Seguir una Línea

Si el Boe-Bot puede ser programado para seguir un haz de luz enfocado frente a él, sobre una mesa, ¿qué le impediría seguir una línea blanca sobre un fondo negro? La respuesta es: nada se lo impediría. El Boe-Bot puede hacerlo y es el proyecto que realizará en la sección Proyectos de este capítulo. De la misma forma, el Boe-Bot podría seguir una línea negra sobre un fondo blanco. Independientemente del color de la línea, esta actividad se llama genéricamente, "seguimiento de línea".

El ancho recomendado para la cinta negra es de 5 cm. El papel de construcción o la cinta aisladora servirán. Con un poco de calibración bajo condiciones controladas de iluminación, el Boe-Bot es un buen seguidor de líneas.

- Las sombras y luces brillantes pueden confundir al BOE-Bot, así que trate de mantener la iluminación lo más uniforme posible. Por ejemplo, tubos fluorescentes sin luz entrando por las ventanas, será lo mejor.

- Además, asegúrese de doblar los fotorresitores hasta llegar lo más cerca de la línea posible. En otras palabras, deberá orientarlo parecido a cuando seguía la luz del piso, pero más cerca aún.

Programa para Seguir la Línea

Cambiando un valor y tres parámetros del programa del ejemplo anterior, el Boe-Bot podrá seguir una línea negra sobre fondo blanco. El Programa 4.4 demuestra esto. En los comentarios al principio del programa, se muestran los datos obtenidos en las pruebas del Programa 4.1. Las lecturas sobre blanco de **`rctime`** se tomaron mientras los fotorresitores miraban hacia una superficie blanca al frente del in Boe-Bot. Las lecturas de **`rctime`** sobre negro fueron tomadas mientras los fotorresitores apuntaban hacia una superficie negra. La diferencia media entre lecturas fue de 77 es este ejemplo. El promedio puede ser tan bajo como 45 y el ejemplo aún funcionará sin problemas. Cuando la diferencia sea aún más pequeña, el valor de la deadband deberá reducirse. Cuando la diferencia sea mayor, la deadband deberá ser incrementada, para mejorar el rendimiento.

- Pruebe los fotorresitores usando el Programa 4.1. Use la información obtenida para estimar un valor de la deadband útil para seguimiento de línea.

- Ajuste el valor de la deadband según sus predicciones y luego ejecute su versión modificada del Programa 4.4. Pruebe distintos valores de deadband hasta que encuentre uno que mejore el rendimiento de su Boe-Bot mientras sigue la línea.

✓ **TIPS**

Si sus mediciones son significativamente mayores que las que se muestran en la sección de comentarios del Programa 4.4, deberá incrementar su deadband. Si sus mediciones, por el contrario, son significativamente menores, disminuir la deadband sería recomendable.

En ambientes muy iluminados, disminuir la deadband puede no ser suficiente. Los capacitores de 0.1 µF pueden ser reemplazados por los de 0.01 µF en los circuitos RC del Boe-Bot. Esto incrementará los tiempos RC por un factor de 10. Tenga esto en cuenta cuando ajuste la deadband.

```
' ¡Robótica! v1.5, Programa 4.4: Seguidor de línea negra

' Programa 4.1 lecturas con el Boe-Bot mirando a superficies negras/blancas.

' color             izquierdo    derecho
' blanco            58           67
' negro             127          152

'----- Declaración --------------

  foto_izq   var word                          ' Almacenan los tiempos RC de los
```

```
  foto_der  var word                            ' fotorresitores izq. y derecho.

'----- Inicialización ------------

  output 2                                      ' Fija a P2 como salida.
  freqout 2, 2000, 3000                         ' Indicador de reset.
  low 12                                        ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:

  ' Mide el tiempo RC del fotorresitor derecho.

    high 3                                      ' Fija a P3 como salida alta.
    pause 3                                     ' Pausa de 3 ms.
    rctime 3,1,foto_der                         ' Mide tiempo RC en P3.

  ' Mide el tiempo RC del fotorresitor izquierdo.

    high 5                                      ' Fija a P5 como salida alta.
    pause 3                                     ' Pausa de 3 ms.
    rctime 5,1,foto_izq                         ' Mide tiempo RC en P5.

  ' Controla si la diferencia entre los tiempos RC cae dentro de la deadband,
  ' 7 en este caso. Si es así, adelante.  Sino a la subrutina control_dir.

  ' IMPORTANTE: El valor de la deadband (actualmente 7) debería reducirse en am-
  ' bientes muy iluminados y agrandarse en oscuros.  Caso contrario, el Boe-Bot
  ' no detectará la línea.  Se necesitarán ajustes finos para optimizar la habili-
  ' dad del Boe-Bot de seguir una línea.

    if abs(foto_izq-foto_der) > 7 then control_dir

    adelante_pulso:
      pulsout 12, 650
      pulsout 13, 850
    pause 20
    goto principal

  ' Salta hacia giro_der o giro_izq dependiendo del mayor tiempo RC.

    control_dir:
      if foto_izq < foto_der then pulso_der
      if foto_izq > foto_der then pulso_izq
```

```
'----- Rutinas de Navegación -------

pulso_izq:                                      ' Aplica un pulso a la izq., luego
  pulsout 12, 650
  pulsout 13, 650
  pause 20
goto principal                                  ' regresa a la rutina principal.

pulso_der:                                      ' Aplica un pulso a la der., luego

  pulsout 12, 850
  pulsout 13, 850
  pause 20
goto principal                                  ' regresa a la rutina principal.
```

Cómo Funciona el Programa para Seguir la Línea Negra

Las tres líneas que se muestran abajo son las únicas que se cambiaron del Programa 4.3. La deadband se incrementó de "2" a "7," y los signos de las desigualdades (< y >) de las últimas dos instrucciones **`if...then`** fueron intercambiados. Como se dijo antes, esto es todo lo que se necesitaba para cambiar un seguidor de luz (fotófilo) en un buscador de oscuridad (fotofóbico). Evitar la luz es la clave para este programa.

```
        if abs(foto_izq-foto_der) > 7 then control_dir
                  .
                  .
                  .
          if foto_izq < foto_der then giro_der
          if foto_izq > foto_der then giro_izq
```

Su Turno

Pruebe el programa con una línea negra que tenga una curva de 45° en la mitad del recorrido.

Inténtelo nuevamente, pero con un giro de 90° y busque el valor de la deadband que logre realizarlo.

Recuerde, puede tener que ajustar su deadband para poder realizar estas maniobras.

Para cualquiera o ambas maniobras anteriores, encuentre los valores límites, máximo y mínimo, de la deadband con las que el Boe-Bot puede maniobrar exitosamente.

Sumario y Aplicaciones

Este capítulo se enfocó en la medición de diferencias en la intensidad de la luz y su uso como guía para el Boe-Bot. El comando **`rctime`** se usó junto con un circuito RC para medir cada fotorresitor. No se tuvo en cuenta el valor de la resistencia de cada fotorresitor, sino que se remarcó la diferencia relativa entre los dos valores. La diferencia se obtiene con una simple resta, pero puede ser usada para determinar en qué dirección es más fuerte la iluminación.

Ejemplo del Mundo Real

La luz tiene muchas aplicaciones en la robótica y el control industrial. Algunos ejemplos incluyen detectar el extremo de un rollo de tela en la industria textil, determinar cuándo encender la vía blanca (alumbrado público) en diferentes momentos a lo largo del año, cuándo tomar una foto, o cuándo regar un cultivo.

La deadband es un problema común en los sistemas de control de navegación. En términos de control de maquinaria, la deadband puede resultar de los errores de medición, debidos a conexiones mecánicas. El resultado es que la deadband es un área de incerteza y deberíamos desarrollar formas creativas para minimizar su valor. Por otro lado, la deadband es usada en algunos termostatos. Al intentar mantener una temperatura, el control diferencial y el control por histéresis usan una región deadband donde no se realiza ninguna corrección de temperatura.

Aplicaciones para el Boe-Bot

Como puede ver, el Boe-Bot puede hacer muchos trucos con un par de fotorresitores como guía. Puede apuntar hacia la luz, moverse hacia un lugar luminoso, seguir un haz de luz y seguir una línea negra con curvas, sobre una superficie blanca. Nada mal para un par de fotorresitores, capacitores y resistores baratos.

Preguntas y Proyectos

4

Preguntas

1. Nombre y describa el elemento en los fotorresistores que cambia su resistencia en respuesta a la luminancia.

2. ¿Cómo hace el BASIC Stamp para medir la resistencia de un circuito RC? ¿Qué valor debe permanecer fijo en un circuito RC para medir una resistencia variable? ¿Por qué?

3. ¿Cuál es el valor del incremento temporal de **`rctime`**?

4. Cuando el valor del fotorresitor aumenta, ¿qué indica?

5. ¿En que se diferencian el programa que sigue la luz y el que se oculta de la luz?

6. ¿Qué relación hay entre la deadband y la tendencia del Boe-Bot a moverse hacia delante? ¿Y con la tendencia del Boe-Bot a girar?

Ejercicios

1. Si tiene un capacitor de 10 µF y su valor de **`rctime`** es 150, ¿cuál es la resistencia del fotorresitor? Pista: Use la ecuación 4.1.

2. Mire la instrucción **`if...then`** que controla la deadband en el Programa 4.2. Si quiere una deadband que ignore las diferencias en la medición del tiempo RC entre –6 y +6, ¿qué argumento modificará y en qué instrucción?

3. Deduzca la ecuación 4.1 pero usando un capacitor de 0.1 µF. ¿Qué tipo de problemas aparecen si el capacitor de 0.1 µF reemplaza al de 0.01 µF? ¿Qué efecto tiene el valor incrementado de RC sobre el tiempo de la medición? ¿Qué efecto tiene el tiempo de la medición sobre el rendimiento de los servos?

4. Dado el rango de valores de **`rctime`** mostrados en el Programa 4.4, ajuste el tiempo de **`pause`** para que los servos reciban los pulsos con una separación de 20 ms. Asuma 1 ms para el tiempo de proceso, además de las pausas y retardos conocidos. Use un valor de tiempo RC promedio para sus correcciones.

5. Desafío: Escriba un código que controle las mediciones de tiempo RC y fije un tiempo de **`pause`** acorde.

Proyectos

1. Desarrolle un programa que siga la línea pero con un único fotorresitor. Pista: En lugar de tomar la diferencia entre las lecturas de dos fotorresitores, tendrá que fijar valores típicos para mediciones de tiempo RC sobre blanco y negro y tomar decisiones basándose en esos valores.

2. Agregue los Bigotes al Boe-Bot. Arme una pista de seguimiento de línea, pero con obstáculos en el camino. Programe el Boe-Bot para que siga la línea y controle los bigotes para detectar obstáculos. Desarrolle rutinas que guíen el Boe-Bot alrededor de los obstáculos y lo devuelvan a la línea.

Asegúrese de aislar cualquier parte del bigote que pueda tocar con otros circuitos, usando cinta aisladora. Lo único que se le debe dejar tocar a un bigote son los obstáculos y su conector de tres pines.

3. Uno de los temas interesante en el uso de la deadband para seguir la línea, es que se pueden realizar ajustes totalmente por software. Este proyecto explora la relación entre el ajuste de la deadband y el ancho de la cinta.

   Repita los ejercicios de Su Turno, Actividad 4 con una cinta negra de 3.75 cm de ancho. No ajuste la separación de sus fotorresitores; solamente modifique el valor de la deadband. Repita esta actividad para una cinta de 2.5 cm de ancho. Tome notas de los límites superior e inferior de los valores de la deadband para cada ancho de cinta. En otras palabras, encuentre los valores más altos y más bajos de la deadband que tienen éxito en el seguimiento de la línea. Grafique sus resultados. ¿Hay alguna relación matemática aparente entre la deadband y el ancho de la cinta? Use el gráfico para buscar una aproximación lineal y desarrolle una ecuación para la deadband. Pruebe la ecuación sobre una cinta de 4.4 cm de ancho.

Capítulo 5: Detección de Objetos Usando Infrarrojo

¿Qué es...

Infrarrojo

Infra significa "por debajo", así que infra-rojo es luz (o radiación electromagnética) que tiene frecuencia más baja, o longitud de onda más larga que la luz roja. Nuestros LEDs IR y sensores trabajan a 980 nm (nanómetros) que es considerado infrarrojo cercano. Visores nocturnos y sensores de temperatura IR usan longitudes de onda de infrarrojo lejano de 2000-10,000 nm dependiendo de la aplicación.

Color	Longitud de onda aprox.
Violeta	400 nm
Azul	470
Verde	565
Amarillo	590
Naranja	630
Rojo	780 nm
Infrarrojo cercano	800-1000 nm
Infrarrojo	1000-2000
Infrarrojo lejano	2000-10,000nm

Uso de Luces Infrarrojas para Ver el Camino

Los productos actuales más exitosos parecen tener sólo una cosa en común: comunicación inalámbrica. Organizadores personales y calculadoras científicas irradian datos en computadoras personales y controles remotos inalámbricos nos permiten cambiar los canales. Con unos pocos componentes baratos y ampliamente disponibles, BASIC Stamp puede usar un LED infrarrojo y un receptor para detectar objetos adelante y al costado de su Boe-Bot.

Afortunadamente, la detección de obstáculos no requiere tanta sofisticación como la visión. Un sistema mucho más simple bastará. Algunos robots usan RADAR o SONAR (algunas veces llamado SODAR cuando se usa en aire en vez de agua). Un sistema aún más simple es usar luz infrarroja para iluminar el camino del robot y determinar cuando se refleja en un objeto. Gracias a la proliferación de controles remotos infrarrojos (IR), iluminadores IR y detectores son baratos y fáciles de conseguir.

La detección infrarroja de objetos del Boe-Bot tiene muchos usos. El Boe-Bot puede usar el infrarrojo para detectar objetos sin necesidad de chocarlos. Como con los fotorresitores, el infrarrojo puede usarse para detectar la diferencia entre blanco y negro para seguimiento de línea. El infrarrojo también puede ser usado para determinar la distancia entre un objeto y el Boe-Bot. El Boe-Bot puede usar esta información para seguir objetos a una distancia fija, o detectar y evitar obstáculos.

Luces Infrarrojas

El sistema infrarrojo de detección de objetos que construiremos en el Boe-Bot es parecido, en varios aspectos, a las luces de un automóvil. Cuando las luces del auto se reflejan en un obstáculo, sus ojos lo detectan y su cerebro procesa la información, haciendo que su cuerpo maneje el auto en forma acorde. El Boe-Bot usa LEDs infrarrojos como se muestran en la Figura 5.1. Ellos emiten infrarrojo y en algunos casos, el infrarrojo se refleja en los objetos y regresa hacia el Boe-Bot. Los ojos del Boe-Bot son los detectores infrarrojos. Éstos envían señales al BASIC Stamp de acuerdo a si reciben o no reflexiones de infrarrojo. El cerebro del Boe-Bot, el BASIC Stamp, toma las decisiones y maneja los servomotores de acuerdo a sus entradas.

Los detectores de IR tienen filtros ópticos internos que dejan pasar muy poca luz excepto el infrarrojo de 980 nm, que es el que queremos que detecte en su fotodiodo interno. También tienen un filtro electrónico que solamente permite el paso a señales de aproximadamente 38.5 kHz. En otras palabras, el detector solamente busca una luz infrarroja que parpadee 38.500 veces por segundo. Esto evita la interferencia de fuentes comunes de IR como la luz del sol y el alumbrado de las casas. La luz del sol es una interferencia de CC (0 Hz) y las luces de las casas parpadean a 100 o 120 Hz, dependiendo de cada país. Dado que los 120 Hz están bastante lejos de la banda de paso de 38.5 kHz del filtro, es para todos los fines prácticos, completamente ignorada por los detectores IR.

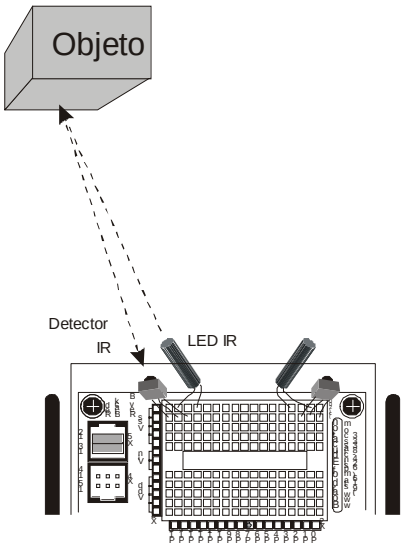


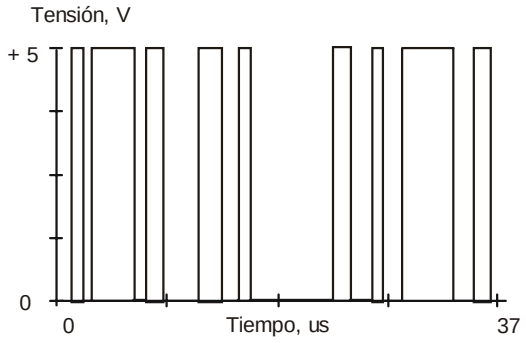
Figura 5.1: Detección de objetos con luces IR.

El Truco con Freqout

Dado que los detectores IR solamente distinguen señales IR de aproximadamente 38.5 kHz, los LEDs IR deben parpadear a esa frecuencia. Un temporizador 555 puede ser usado para este propósito, pero este circuito es más complejo y menos funcional que el que usaremos en este capítulo y el siguiente. Por ejemplo, el método de la detección IR introducido aquí puede ser usado para medir distancia, mientras que el temporizador 555 necesitaría hardware adicional para hacerlo.

Un par de entusiastas del BOE-Bot encontraron un truco interesante que hizo innecesario el temporizador 555. Este esquema usa el comando **`freqout`** sin el filtro RC que normalmente se usa para suavizar la señal y lograr una sinusoide. Incluso aunque la máxima frecuencia que puede enviar **`freqout`** es 32768 Hz, la salida sin filtrar de **`freqout`** contiene armónicos que lo hacen compatible con un detector IR de 38.5 kHz IR. Más útil aún es el hecho de que puede usar una instrucción como **`freqout pin, duración, 38500`** para enviar un armónico de 38.5 kHz que el sensor IR detectará.

La Figura 5.2 muestra (a) la señal enviada por el comando **`freqout`** pin, duración, 27036. Los filtros electrónicos pueden detectar componentes de esta señal, llamados armónicos. Los dos armónicos de baja frecuencia dominantes de la señal **`freqout`** son mostrados en las Figuras 5.2 (b) y (c). La Figura 5.2 (b) muestra la armónica fundamental y la Figura 5.2 (c) la tercer armónica de la señal **`freqout`**. La tercer armónica puede ser controlada directamente, ingresando comandos tales como **`freqout pin, duración, 38500`** (en lugar de **`27036`**) para 38.5 kHz, o **`freqout pin, duración, 40000`** para 40 kHz, etc.



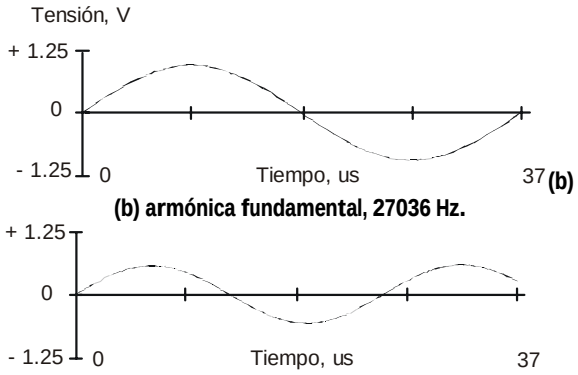


Figura 5.2: (a) Pulsos sin filtrar enviados por `freqout pin, duración, 27036`

Incluso aunque el truco de freqout funcione, hay un problema adicional. El BASIC Stamp no soporta multitarea. Esto es un problema porque el detector IR solamente envía una señal de estado bajo, indicando que detectó un objeto, mientras está recibiendo el IR de 38.5 kHz. De otro modo, emite un estado alto. Afortunadamente, el detector tarda tanto en cambiar de estado que el BASIC Stamp puede capturar el valor. La razón por la que la salida del detector tarda tanto en cambiar de estado, es debido a su respuesta lenta a señales con tiempos en estado alto y bajo distintos, como la de la Figura 5.2 (a).

Actividad 1: Construcción y Prueba del Nuevo Transmisor/Detector de IR

Componentes

(1) Parlante piezoeléctrico
(2) LEDs IR con tubo plástico
(2) Detectores IR
(2) Resistores 220 Ω
(varios) Cables de interconexión

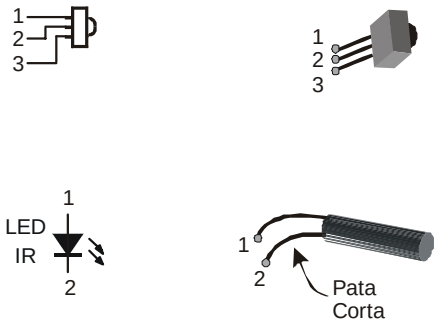


Figura 5.3: Arriba detector IR , abajo LED IR.

¡Constrúyalo!

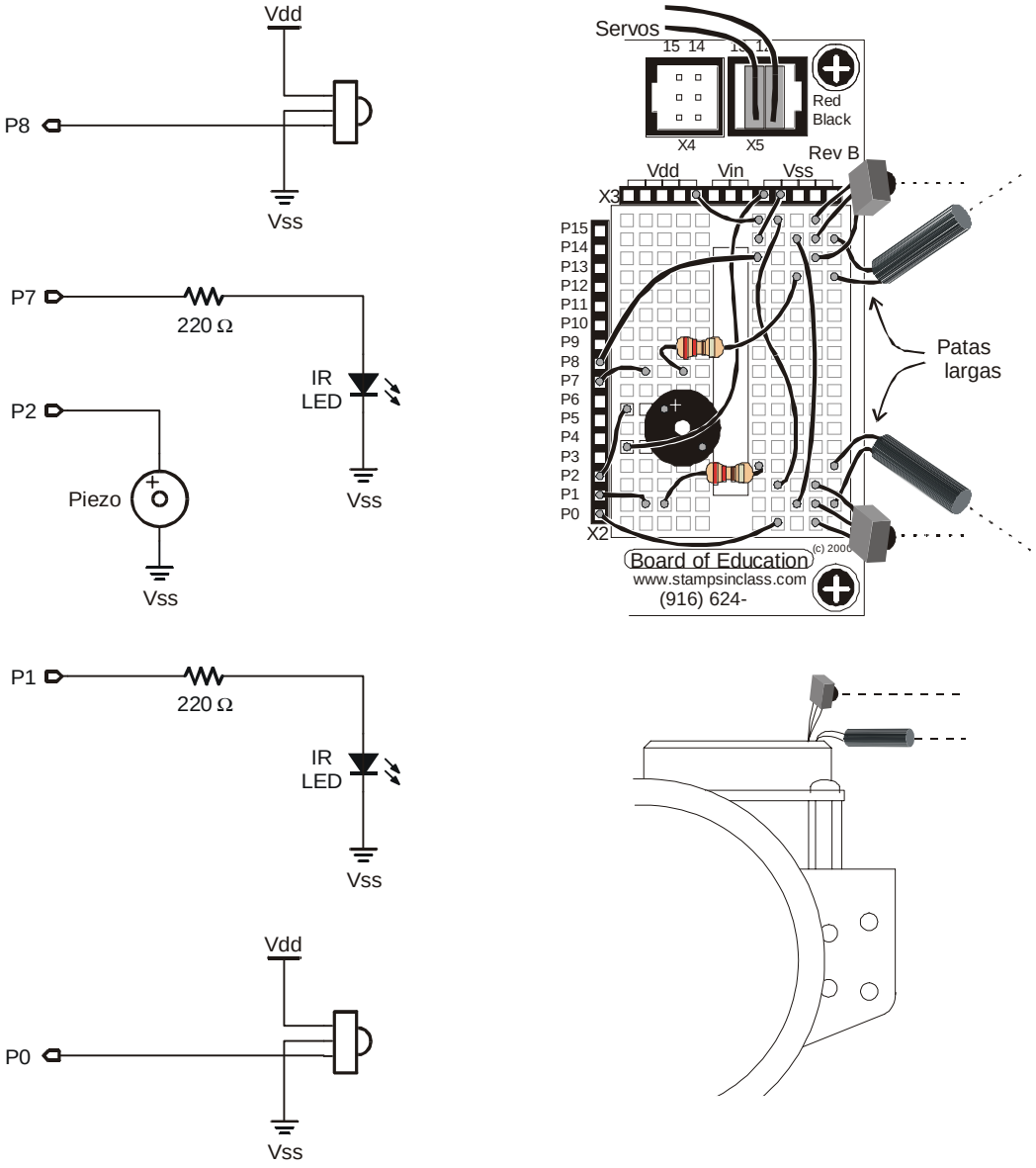


Figura 5.4: IR (a) Esquema y **(b) distribución de componentes.**

Un par IR (LED IR y detector) se monta en cada esquina de la protoboard. La Figura 5.4 muestra el circuito IR y la distribución de componentes. Arme el circuito como se muestra en la figura.

Probando los Pares IR

La clave para hacer funcionar cada par IR es enviar 1 ms de armónicos sin filtrar de 38.5 kHz con **`freqout`**, e inmediatamente verificar el detector IR, guardando su valor de salida. El estado de salida normal del detector IR cuando no encuentra ninguna señal IR, es alto. Cuando el detector IR ve los armónicos de 38500 Hz enviados por el LED IR, su salida cambiará de estado alto a bajo. Por supuesto, si el IR no se refleja en algún objeto, la salida del detector IR quedará en estado alto. El Programa 5.1 muestra un ejemplo para este método de lectura.

- Ingrese y ejecute el Programa 5.1.
- Este programa usa la Debug Terminal, así que deje conectado el cable serial a la BOE mientras ejecuta el Programa 5.1.

```
'¡Robótica! v1.5, Programa 5.1: Lectura de pares IR.

'----- Declaración --------------

  det_IR_izq   var  bit                       ' Dos variables tipo bit para alma-
  det_IR_der  var  bit                        ' cenar las salidas de detect. IR.

'----- Inicialización ------------

  output 2                                    ' Config. todas las E/S que envían
  output 7                                    ' señales freqout como salidas.
  output 1

'----- Rutina Principal --------------

principal:
                                              ' Detecta objeto a la izquierda.
  freqout 7, 1, 38500                         ' Envía freqout al LED IR izq.
  det_IR_izq = in8                            ' Guarda salida detector IR en RAM.
                                              ' Detecta objeto a la derecha.
  freqout 1, 1, 38500                         ' Repite para el par IR derecho.
  det_IR_der = in0

  ' Muestra las salidas IR en la Debug terminal para su control.

  debug home, "Izquierda= ", bin1 det_IR_izq, "   Derecha= ", bin1 det_IR_der

goto principal
```

- Mientras se esté ejecutando el Programa 5.1, apunte los detectores IR de forma que el infrarrojo no pueda rebotar sobre ninguna superficie. La mejor forma de hacer esto es apuntar el Boe-Bot hacia el cielorraso. La ventana Debug debería mostrar los valores izquierdos y derechos iguales a "1."

- Si coloca su mano frente a un par IR, debería cambiar el valor mostrado en la Debug Terminal para ese par, de "1" a "0." Si quita su mano, el par debería regresar al estado "1". Esto se debe cumplir para cada par de sensores, en forma individual, e incluso podría poner su mano frente a ambos detectores y hacer que sus salidas cambien de "1" a "0."

- Si los pares IR pasaron todas las pruebas, está listo para continuar; caso contrario, busque errores en el programa y el circuito.

Cómo Funciona el Programa Lectura de pares IR

Dos variables tipo bit se declaran para almacenar el valor de la salida de cada detector IR. El primer comando **`freqout`** de la rutina **`principal`** es diferente. El comando **`freqout 7, 1, 38500`** envía el patrón que se muestra en la Figura 5.2, haciendo parpadear rápidamente el circuito del LED IR izquierdo. Los armónicos contenidos en esta señal pueden o no, rebotar sobre algún objeto. Si rebotan contra un objeto y el detector IR reconoce la señal, cambiará a estado bajo el pin de E/S P8. Si no rebotan, el detector IR mantendrá en estado alto a P8. El comando inmediatamente posterior a **`freqout`** es el que comprueba el estado de la salida del detector IR, almacenándolo en una variable en la RAM. La instrucción **`det_IR_izq = in8`** controla a P8 y almacena el valor ("1" para alto o "0" para bajo) en el bit de la variable **`det_IR_izq`**. Este proceso se repite para el otro par IR, guardando el valor en la variable **`det_IR_der`**. El comando **`debug`** luego muestra los valores en la ventana de depuración (debug).

Su Turno

- Experimente modificando las frecuencias de sus pares IR con valores mayores que 38.5 kHz. Por ejemplo, pruebe con 39.0, 39.5, 40.0, 40.5 y 41 kHz. Observe la máxima distancia a la que se detecta un objeto, acercándolo progresivamente a los pares IR y tomando notas de la distancia a la que los pares cambian de "1" a "0."

Actividad 2: Detección y Evasión de Obstáculos

Lo interesante de los detectores IR es que sus salidas son como las de los bigotes. Cuando no se detecta ningún objeto, la salida es alta; cuando se detecta un objeto, la salida es baja. En esta actividad, el Programa 3.2: Exploración con Bigotes, es modificado para que trabaje con los detectores IR.

Convirtiendo un Programa de Bigotes para Detección / evasión IR de Objetos

Exploración con Bigotes puede ser modificado para que cada circuito de LED IR reciba la señal **`freqout`**. Inmediatamente después de enviar la señal, el estado de la salida de cada detector IR debe ser controlado y almacenado. Una vez que se ha almacenado la información, puede ser comparada usando las mismas instrucciones **`if...then`** y rutinas de navegación usadas en el programa de exploración con bigotes original (Programa 3.2).


```
'¡Robótica! v1.5, Programa 5.2: Exploración con bigotes ajustado para pares IR.

'----- Declaración --------------

  cuenta_pulsos  var   byte                  ' Contador del bucle for...next.
  det_IR_izq   var  bit                      ' Dos variables tipo bit para alma-
  det_IR_der  var  bit                       ' cenar las salidas de detect. IR.

'----- Inicialización ------------

  output 2                                   ' Config. todas las E/S que envían
  output 7                                   ' señales freqout como salidas.
  output 1
  freqout 2, 2000, 3000                      ' Indicador de reset.
  low 12                                     ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:
                                             ' Detecta objeto a la izquierda.
  freqout 7, 1, 38500                        ' Envía freqout al LED IR izq.
  det_IR_izq = in8                           ' Guarda salida detector IR en RAM.
                                             ' Detecta objeto a la derecha.
  freqout 1, 1, 38500                        ' Repite para el par IR derecho.
  det_IR_der = in0

  ' Con la excepción de que se usan los valores almacenados en la RAM en lugar
  ' de los valores de los registros de entrada, el proceso de decisión es el mismo
  ' que el usado en el Programa 3.2.

  if det_IR_izq  = 0 and det_IR_der = 0 then giro_u
  if det_IR_izq  = 0 then giro_der
  if det_IR_der  = 0 then giro_izq

  ' Los comandos desde este punto en adelante son idénticos al
  ' Programa 3.2: Exploración con Bigotes.
```

```
  adelante:                                    ' Si no detecta, un pulso adelante.
    pulsout 12,650
    pulsout 13,850
    pause 20

goto principal                                 ' Controla nuevamente.

'----- Rutinas de Navegación -------

  giro_izq:                                    ' Rutina giro a la izq.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 650
      pulsout 13, 650
      pause 20
    next
  goto principal

  giro_der:                                    ' Rutina giro a la der.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 30
      pulsout 12, 850
      pulsout 13, 850
      pause 20
    next
  goto principal

  giro_u:                                      ' Rutina de Giro en U.
    gosub atras                                ' Llama a atras: antes de girar.
    for cuenta_pulsos = 0 to 60
      pulsout 12, 850
      pulsout 13, 850
      pause 20
    next
  goto principal

'----- Subrutina de Navegación -----

  atras:                                       ' Usada por todas las rutinas.
    for cuenta_pulsos = 0 to 60
      pulsout 12, 850
      pulsout 13, 650
      pause 20
    next
  return
```

Cómo Funciona el Programa Exploración con Bigotes Modificado para Pares IR

Dos variables tipo bit, **det_IR_izq** y **det_IR_der**, se agregan para tomar y retener los estados de salida de los detectores IR.

```
declaraciones:
  cuenta_pulsos  var   byte
  det_IR_izq   var  bit
  det_IR_der  var  bit
```


La rutina **principal** tiene cuatro comandos adicionales, dos para controlar la salida de cada detector IR. Cada comando **freqout** envía una señal sin filtrar de 1 ms al circuito del LED IR del par. El valor de la entrada del BASIC Stamp que está conectado a la salida del detector IR se almacena como variable tipo bit en la RAM. Por ejemplo, el comando **freqout 7, 1, 38500** es seguido inmediatamente por la instrucción **det_IR_izq = in8**. Este comando asigna el valor presente en la entrada P8 a **det_IR_izq,** almacenando el estado de la salida del detector IR izquierdo.

```
principal:

  control_pares_IR:

    freqout 7, 1, 38500
    det_IR_izq = in8

    freqout 1, 1, 38500
    det_IR_der = in0
```

Los bits almacenados por cada detector IR pueden ser usados del mismo modo que el programa de navegación con bigotes lo hacía (con los bigotes). Con una excepción, las rutinas de navegación que se ejecutan de acuerdo a las instrucciones **if...then** son idénticas a las que usaba originalmente el Programa 3.2: Exploración con Bigotes. Las instrucciones **if...then** se acomodaron para usar los valores almacenados de cada detector IR; mientras que las instrucciones **if...then** del programa original usaban directamente los valores de entrada.

```
    if det_IR_izq  = 0 and det_IR_der = 0 then giro_u
    if det_IR_izq  = 0 then pulso_der
    if det_IR_der  = 0 then pulso_izq
```

Su Turno

Como en el Programa 3.2, puede ajustar los argumentos finales de los bucles **for...next** para mejorar el comportamiento del Boe-Bot en los giros y el retroceso.

Actividad 3: Exploración por Números en Tiempo Real

En el Programa 5.2, el Boe-Bot controlaba los sensores cada vez que emitía un pulso **`adelante`**, para ver si podía seguir moviéndose en ese sentido. Cuando el Boe-Bot realizaba maniobras, normalmente estaban predeterminadas por el programa. Otra forma de navegación IR es controlar los sensores, aplicar un solo pulso basándose en esas entradas y luego controlar los sensores otra vez. El Boe-Bot se comporta muy distinto usando esta técnica.

Exploración IR en Tiempo Real

El Programa 5.3 controla los pares IR y entrega uno de cuatro pulsos diferentes basándose en los sensores. Cada una de las rutinas de navegación consta de un único pulso hacia **`adelante`**, **`giro_izq`**, **`giro_der`** o **`atras`**. Después de aplicar el pulso, los sensores se controlan nuevamente, luego se aplica otro pulso, etc. Este programa también usa técnicas de programación que encontrará muy útiles.

```
' ¡Robótica! v1.5, Programa 5.3: Exploración IR por Números en Tiempo Real

'----- Declaración --------------
                                          ' Se usan los 2 bits de menor peso
  sensores var nib                        ' para almacenar los valores del.
                                          ' detector IR.
'----- Inicialización ------------

  output 2                                ' Config. todas las E/S que envían
  output 7                                ' señales freqout como salidas.
  output 1
  freqout 2, 2000, 3000                   ' Indicador de reset.
  low 12                                  ' P12 y 13 salidas bajas.
  low 13

'----- Rutina Principal --------------

principal:
                                          ' Detecta objeto a la izquierda.
   freqout 7,1,38500                      ' Envía freqout al LED IR izq.
   sensores.bit0 = in8                    ' Guarda salida detector IR en RAM.
                                          ' Detecta objeto a la derecha.
   freqout 1,1,38500                      ' Repite para el par IR derecho.
   sensores.bit1 = in0

   pause 18                               ' Pausa de 18 ms (2 ms en freqout).

   ' Al cargar los valores de salida del detector IR en los 2 bits de menor peso
```

```
   ' de sensores, se genera un número entre 0 y 3 que será usado por branch.

   branch sensores,[atras,giro_izq,giro_der,adelante]

'----- Rutinas de Navegación -------

      adelante:     pulsout 13,850: pulsout 12,650: goto principal
      giro_izq:   pulsout 13,650: pulsout 12,650: goto principal
      giro_der:  pulsout 13,850: pulsout 12,850: goto principal
      atras:    pulsout 13,650: pulsout 12,850: goto principal
```


Cómo Funciona Exploración IR por Números en Tiempo Real

- Busque el comando **`branch`** en el Apéndice C: Referencia Rápida PBASIC o en el BASIC Stamp Manual.

Este programa declara la variable **`sensores`**, como tipo nibble (4 bits) de RAM. De los cuatro bits de la variable sensores, solamente se usarán los dos de menor peso. El bit-0 se usa para almacenar la salida del detector izquierdo y el bit-1 es usado para almacenar la salida del detector derecho.

```
declaraciones:
  sensores var nib
```

Los pines de E/S P7, P1 y P2 se declaran como salidas. P2 se declara como salida para enviar señales al parlante con freqout. P7 y P1 se declaran como salidas para encender los circuitos de los LEDs IR derecho e izquierdo.

```
inicializacion:
  output 7
  output 1
  output 2
  freqout 2,2000,3000
```

La rutina principal comienza con los comandos **`freqout`** usados para enviar señales IR, pero los comandos que siguen a cada freqout son ligeramente diferentes de los del programa anterior. En lugar de guardar el valor del bit del pin de entrada en una variable tipo bit, cada valor es almacenado como un bit en la variable **`sensores`**. Al bit-0 de **`sensores`** se le asigna el valor binario de **`in8`** y al bit-1 de la variable sensores se le asigna el valor binario de **`in0`**. Después de configurar los valores de los dos bits de menor peso de la variable sensores, obtendremos un valor decimal entre "0" y "3." El comando **`branch`** usa estos números para determinar a qué etiqueta enviar el control del programa.

```
principal:

   freqout 7,1,38500
   sensores.bit0 = in8

   freqout 1,1,38500
   sensores.bit1 = in0

   pause 18

   branch sensores,[atras,giro_izq,giro_der,adelante]
```

Los cuatro números binarios posibles se muestran en la Tabla 5.1. También se muestra la acción que producirá branch basándose en ese valor.

Tabla 5.1: Detector de Estados IR como Números Binarios

Valor Binario de sensores	Valor Decimal de sensores	Qué indica el valor, Acción de branch sensores
0000	0	**in8 = 0** e **in0 = 0**, Ambos detectores IR encuentran un objeto, pulso hacia **atras**.
0001	1	**in8 = 0** e **in0 = 1**, El detector IR izquierdo encuentra un objeto, pulso **giro_der**
0010	2	**in8 = 1** e **in0 = 0**, El detector IR derecho encuentra un objeto, pulso **giro_izq**
0011	3	**in8 = 1** e **in0 = 1**, Ningún detector IR encuentra un objeto, pulso hacia **adelante**.

Dependiendo del valor de la variable **sensores**, el comando **branch** envía el programa a una de cuatro rutinas: **adelante**, **giro_izq**, **giro_der**, o **atras**. Cualquiera sea la rutina ejecutada, los servos recibirán un único pulso, en la dirección apropiada, después del cual, la rutina envía el control del programa a la etiqueta **principal** para controlar el estado de los sensores nuevamente.

```
rutinas:

      adelante:       pulsout 13,850: pulsout 12,650: goto principal
      giro_izq:     pulsout 13,650: pulsout 12,650: goto principal
      giro_der:    pulsout 13,850: pulsout 12,850: goto principal
      atras:      pulsout 13,650: pulsout 12,850: goto principal
```

✓ **TIP**

Cada rutina consta de una etiqueta, seguida por tres comandos, todos en la misma línea. Cuando ponga más de un comando en la misma línea, debe separarlos con el caracter dos puntos (:). Las etiquetas solamente pueden aparecer al principio de una línea con varios comandos PBASIC.

Su Turno

Puede reordenar las etiquetas del comando branch para que el Boe-Bot realice diferentes maniobras cuando encuentra un obstáculo. Una actividad interesante es reemplazar la etiqueta **`atras`** por la etiqueta **`adelante`**. Habrá dos casos en que se elija **`adelante`** dentro del comando **`branch`**, pero esto no es un problema. También, intercambie las etiquetas **`giro_izq`** y **`giro_der`**.

- Realice los cambios que acabamos de mencionar.
- Ejecute la versión modificada del Programa 5.3 y haga que el Boe-Bot siga su mano a medida que lo guía hacia algún lugar.

Si deja su mano quieta, el Boe-Bot la chocará. Debido a esto, un Boe-Bot no puede ser programado para seguir a otro sin algún control de distancia. Si el que está adelante se detiene, el de atrás lo chocará. Este problema se solucionará en un ejemplo del próximo capítulo.

Sumario y Aplicaciones

Este capítulo cubrió la técnica para la detección infrarroja de objetos. Iluminando el camino del Boe-Bot con infrarrojo y observando su reflexión, se pueden detectar objetos. Los circuitos de LEDs IR se usaron para enviar señales de 38.5 kHz usando una única y poco conocida propiedad de **`freqout`**. Esta propiedad permite controlar los armónicos de la señal PWM que envía **`freqout`** a los LEDs IR.

Las técnicas de programación en PBASIC también se mejoraron, para minimizar el tiempo que se pierde en la captura de las señales de salida de los detectores IR. La señal de salida de cada detector tiene un retardo que hace posible que el BASIC Stamp lea salida del detector IR incluso después de que la señal de **`freqout`** se dejó de transmitir.

También se introdujeron técnicas de navegación con control de sensores entre pulsos de los servos y procesamiento binario de las salidas de los sensores. Usando estas dos técnicas juntas se logró un rendimiento óptimo del Boe-Bot.

Ejemplo del Mundo Real

El uso de infrarrojo es muy popular en los productos electrónicos. Controles remotos de TV, asistentes personales o palmtop computers y calculadoras científicas, usan el infrarrojo para sus comunicaciones. Existen muchos sistemas para transmitir datos. Un control remoto de TV, por ejemplo, envía una señal de estado alto emitiendo pulsos IR a 38.5 kHz. Una señal de estado bajo equivale a no enviar IR. Los detectores de algunos televisores, video caseteras, etc., son idénticos a los usados en el Boe-Bot.

Los circuitos de detección infrarroja de equipos de apertura automática de puertas y equipos automatizados en supermercados, se basan en la misma teoría de operación usada en el Boe-Bot. Cada vez que acciona una de esta puertas automáticas, es debido a que reflejó el haz IR hacia el detector. Los detectores infrarrojos también se usan en las cintas transportadoras. Las fábricas los usan para contar la cantidad de productos a medida que pasan y los supermercados los usan para detectar cuándo la mercadería llega al final de la cinta transportadora, donde está la caja registradora. Estas cintas mueven automáticamente la mercadería hacia la caja, para que el empleado pueda facturarla. Para evitar que la mercadería se apile al final de la cinta, contra el scanner (láser que lee los códigos de barras), un detector IR montado en el final de la cinta, controla el motor que la mueve, deteniéndolo cuando un producto llega hasta el extremo.

Aplicaciones para el Boe-Bot

Lo más importante sobre el uso de los detectores IR es que le permiten al Boe-Bot detectar objetos sin tocarlos. En competencias de laberintos, donde se pierden puntos por tocar las paredes, es una gran ventaja.

Preguntas y Proyectos

Preguntas

1. ¿Qué significa infrarrojo? ¿En qué se diferencia el infrarrojo, del infrarrojo cercano?
2. ¿Qué filtros poseen los detectores IR usados en el Kit ¡Robótica!? ¿Qué hace cada uno?
3. Describa qué representan las salidas de los detectores IR.
4. ¿Por qué es importante controlar y guardar el estado del detector IR inmediatamente después de enviar la señal de 38.5 kHz?
5. ¿Qué sucede si envía una señal de 39.5 kHz en lugar de una de 38.5 kHz?
6. ¿En qué se parece el Programa 5.2 al Programa 3.2? ¿En que se diferencian?
7. ¿Qué valores espera ver almacenados en la variable **sensores** del Programa 5.3?
8. ¿Qué caracter se usa para separar varios comandos PBASIC escritos en la misma línea? ¿Qué cuidado especial se debe tener con las etiquetas, cuando se usan varios comandos en la misma línea?

Ejercicios

1. Si se quiere enviar una armónica infrarroja de 35 kHz mediante los LEDs IR del Boe-Bot, ¿qué comando usaría?
2. Modifique el Programa 5.2 de forma que el par IR derecho se controle antes que el izquierdo.
3. Modifique el Programa 5.2 para que el Boe-Bot siga objetos en lugar de evitarlos. Describa los problemas que encuentra, si es que hay alguno.

Proyectos

1. Hay dos formas de bajar la sensibilidad del par IR, logrando así que detecte los objetos a menor distancia. La primera involucra un cambio físico: el reemplazo de los resistores de 220 Ω con resistores de 470 Ω. La segunda involucra un desajuste de la frecuencia de trabajo, enviando un armónico que no sea de la frecuencia central del detector IR de 38.5 kHz.

   - Pruebe la máxima distancia de detección de cada par IR colocando un objeto frente al par mientras ejecuta el Programa 5.1. Aleje suavemente el objeto del par IR, hasta que el valor mostrado en la ventana debug cambie de "0" a "1" y anote la distancia
   - Reemplace los resistores de 220 Ω por resistores de 470 Ω.
   - Repita el paso anterior y anote los cambios en la distancia máxima de detección para cada par IR.
   - Vuelva a colocar los resistores de 220 Ω en su lugar.
   - Coloque el objeto usado para medir la distancia de detección máxima, a la distancia que se determinó con los resistores de 470 Ω.
   - Modifique los argumentos de ***`frecuencia`*** de **`freqout`** sumándole 25, cada vez que los altera. Por cada modificación, no olvide volver a ejecutar el Programa 5.1. Cuando comience a observar un cambio de 0 a 1 en la Debug Terminal, indica que el objeto está en el límite máximo de alcance.

2. Uno de los inconvenientes con la detección IR de objetos es que los pares IR del Boe-Bot no detectan el negro. Esto es debido a que el negro no refleja el IR sino que lo absorbe. El Boe-Bot tiende a chocarse contra objetos negros cuando navega con los pares IR, debido a que no puede verlos. Agregue los bigotes a su Boe-Bot y modifique el Programa 5.2 para que controle el estado de los bigotes antes de controlar los pares IR. De esta forma el Boe-Bot podrá verificar si ha chocado contra un objeto negro.

Recuerde: Cubra las partes de los bigotes que pudieran tocar otros componentes eléctricos, con una cinta aisladora.

Capítulo 6: Determinación de la Distancia Usando Barrido de Frecuencia

¿Qué es un Barrido de Frecuencia?

En general, un barrido de frecuencia es lo que usted hace cuando busca una estación de radio que le guste. Fija la radio en una frecuencia y controla la salida. Si no le gusta la canción que están pasando, cambia la frecuencia y controla la salida nuevamente.

Actividad 1: Probando el Barrido de Frecuencia

El Boe-Bot puede ser programado para enviar diferentes frecuencias de IR y controlar si detecta un objeto a cada frecuencia. Se puede determinar a qué distancia se encuentra un objeto si registramos a que frecuencia fue encontrado por el detector IR. El eje vertical del gráfico de la Figura 6.1 muestra cómo disminuye la sensibilidad de los detectores IR debido a sus filtros electrónicos, cuando recibe frecuencias mayores de 38.5 kHz. El filtro hace que el detector IR se vuelva menos sensible a estas frecuencias. Otra consecuencia de esto es que a una frecuencia menos sensitiva, detectará los objetos a una distancia menor. Dado que el detector es menos sensible, necesitará un rebote de IR más potente para reconocer la señal.

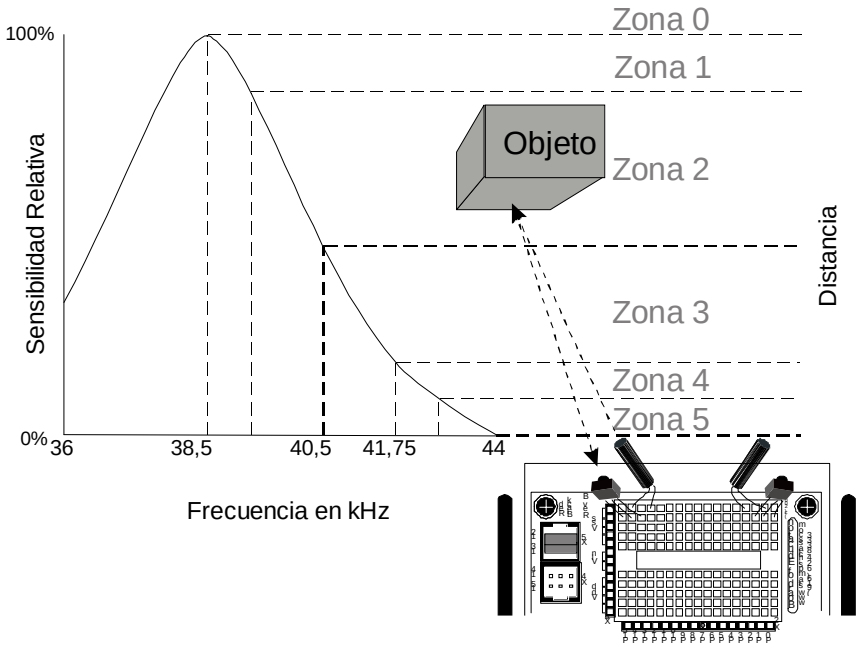


Figura 6.1: El eje izquierdo del gráfico compara la frecuencia de IR con la sensibilidad relativa del detector IR. El lado derecho del gráfico muestra cómo se relaciona la sensibilidad relativa del detector IR con la distancia. A medida que disminuye la sensibilidad del detector con el incremento de la frecuencia, el objeto debe estar más cerca para que la señal IR sea detectada. ¿Por qué más cerca? Cuando los detectores se hacen menos sensibles por el cambio de la frecuencia del emisor, necesitan una señal de mayor intensidad para poder detectarla. Si el IR rebota contra un objeto más cercano, la intensidad recibida en el detector es mayor.

El eje derecho de la Figura 6.1 muestra cómo se pueden usar diferentes frecuencias para indicar la zona en la que fue detectado un objeto. Comenzando con una frecuencia de 38.5 kHz, se puede determinar si hay un objeto dentro de las Zonas 1 a 5. Si no se detecta ningún objeto, puede estar mas allá del límite del detector (Zona 0). Si un objeto es detectado, verificando a 39.25 kHz se puede obtener la primer información sobre su distancia. Si se detectó el objeto a 38.5 kHz pero no a 39.25 kHz, entonces el rebote se originó en la Zona 1. Si el objeto fue detectado a las dos frecuencias, pero no a 40.5 kHz, sabemos entonces que está en la Zona 2. Si las tres frecuencias anteriores detectaron el objeto, pero no sucedió lo mismo a 41.75 kHz, entonces sabemos que está en la Zona 3. Si las cuatro frecuencias lo detectaron, pero 42.5 kHz no, sabremos que está en la Zona 4. Si absolutamente todas las frecuencias lo detectaron, entonces está en la Zona 5.

FYI

La técnica del barrido de frecuencia usada en este capítulo trabaja bastante bien en el Boe-Bot y los componentes que se usan, cuestan una fracción de los sensores de distancia IR comunes. La desventaja es que la exactitud de este método también es una fracción de la precisión de los sensores de distancia IR comunes. Para las tareas básicas del Boe-Bot que requieran alguna percepción de la distancia, tales como seguir a otro Boe-Bot, esta técnica interesante funcionará. Además de agregarle percepción de distancia de baja resolución a los sentidos del Boe-Bot, también provee una introducción a conceptos sobre filtros y respuesta en frecuencia.

¡Constrúyalo!

- Use el mismo circuito de detección IR del Capítulo 5, mostrado en la Figura 5.4, para esta actividad.

Programando el Medidor de Distancia IR

Programar al BASIC Stamp para que envíe diferentes frecuencias involucra el uso del bucle **`for...next`**. La variable **`contador`** puede ser usada para darle al comando **`freqout`** diferentes frecuencias. Este programa introduce el uso de arreglos (vectores). Los arreglos se usan en el Programa 6.1 para almacenar las salidas de los detectores IR a diferentes frecuencias. Para la variable **`valores_izq`**, la salida de la Zona 0 es almacenada en el bit-0 de **`valores_izq`**. La salida de la Zona 1 es almacenada en el bit-1, **`valores_izq.bit1`** y así hasta la Zona 5, que es almacenada en el bit-5 de **`valores_izq`**. Las mismas mediciones se realizan con **`valores_der`**.

- Ingrese y ejecute el Programa 6.1.
- Este programa usa la Debug Terminal, así que debe dejar conectado el cable serial a la BOE mientras se esté ejecutando el Programa 6.1.

```
' ¡Robótica! v1.5, Programa 6.1: Medidor de Distancia IR.

'----- Declaración --------------
```

```
  contador          var          nib          ' Variable multipropósito.
  valores_izq       var          byte         ' Variables para guardar las salidas
  valores_der       var          byte         ' del barrido de frec.
  IR_freq           var          word         ' Almacena la frecuencia de freqout.

'----- Inicialización ------------

  output 7                                    ' Config. todas las E/S que envían
  output 1                                    ' señales freqout como salidas.

'----- Rutina Principal --------------

  principal:

    valores_izq = 0                           ' Fija valores_izq y valores_der a 0
    valores_der = 0

    ' Carga las salidas de los sensores en valores_izq y valores_der usando un
    ' bucle for...next, una tabla lookup y direccionamiento de bit.

    for contador = 0 to 4

      lookup contador,[37500,38250,39500,40500,41500], IR_freq

      freqout 7,1, IR_freq
      valores_izq.lowbit(contador) = ~in8

      freqout 1,1, IR_freq
      valores_der.lowbit(contador) = ~in0

    next

    ' Muestra valores_izq y valores_der en formato binario y ncd.

    debug home, cr, cr, "Lecturas Izquierda     Lecturas Derecha", cr
    debug "  ",bin8 valores_izq, "          ", bin8 valores_der, cr
    debug "  ",dec5 ncd(valores_izq), "            ", dec5 ncd(valores_der), cr, cr

  goto principal
```

Cuando el Boe-Bot se enfrenta a una pared (entre 3 y 5 cm), la Debug Terminal debería mostrar algo similar a la Figura 6.2. A medida que el Boe-Bot se acerca y aleja de la pared, los números mostrados por la Debug Terminal deberían aumentar y disminuir.

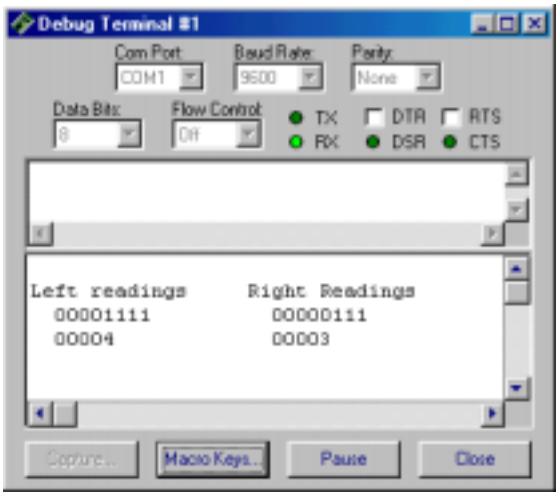


Figura 6.2: Datos de barrido de frecuencia en formato binario y ncd.

- Coloque el Boe-Bot de forma que los LEDs IR queden a aproximadamente 1 cm de la pared. Las lecturas izquierda y derecha deberían estar en "4" o "5". Si no es así, asegúrese de que cada detector IR esté mirando en la misma dirección que su LED IR. Además, asegúrese de usar resistores de 220 Ω.

- Gradualmente aleje al Boe-Bot de la pared. A medida que lo hace, las lecturas izquierda y derecha deberían disminuir hasta llegar a "0."

- Si cualquiera o ambos lados permanecen fijos con todos ceros o todos unos, indica un posible error en el conexionado o en el programa. Si este es el caso, desconecte el porta pilas de la BOE. Luego, busque errores en su conexionado o en el código PBASIC.

FYI

La distancia de detección máxima es de 20 a 30 cm., dependiendo de la reflectividad de la pared. Pueden hacer falta algunos ajustes sobre la dirección a la que apuntan los pares IR, para que ambos entreguen los mismos resultados a una distancia dada. Un alto nivel de precisión NO ES necesario para estas actividades. El Apéndice H: Ajuste de la Detección de Distancia IR, presenta un método para calibrar los detectores de distancia del Boe-Bot. Este método puede llevar mucho tiempo y no es necesario para realizar las Actividades 2 o 3.

✓
TIP

Use un pelacables para retirar 1 cm de aislamiento de uno de los cables de interconexión que vienen con la BOE. Deslice este aislamiento por una de las patas del LED IR. Esto evitará que las patas del LED hagan un cortocircuito cuando se realizan ajustes.

Cómo Funciona el Programa Medidor de Distancia IR

- Busque el comando **lookup** en el Apéndice C: Referencia Rápida de PBASIC o en el BASIC Stamp Manual (en Inglés) antes de continuar.

Contador es una variable tipo nibble (4 bits) que es usada como índice del bucle **for...next**. Este bucle es usado para controlar los detectores IR a varias frecuencias. Las variables **valores_izq** y **valores_der** almacenan las salidas de los detectores IR izquierdo y derecho a todas las frecuencias usadas. Cada variable almacena cinco mediciones binarias. Dado que las salidas de los detectores IR se controlan a varias frecuencias, **IR_freq** es una variable que almacena el valor de la frecuencia que se envía en cada paso por el bucle de barrido de frecuencias.


```
declaraciones:

  contador        var         nib
  valores_izq     var         byte
  valores_der     var         byte
  IR_freq         var         word
```

La rutina principal contiene dos rutinas, una para el barrido de frecuencia y otra para mostrar los datos obtenidos. El primer paso en el barrido de frecuencia es fijar los valores de **valores_izq** y **valores_der** en cero. Esto es importante debido a que se manipularán individualmente los bits de cada variable. Al borrar el contenido de **valores_izq** y **valores_der** cada variable comienza en un estado limpio. Luego se pueden modificar los bits individualmente a "1" o "0", dependiendo de la medición de los detectores IR.

```
principal:

  valores_izq = 0
  valores_der = 0
```

El bucle **for…next** es donde se produce el barrido de frecuencia. El comando **lookup** controla el valor del **contador** para determinar qué frecuencia copiar en la variable **IR_freq**. Cuando **contador** es "0", 37500 es copiado en **IR_freq**. Cuando **contador** es "1", 38250 es copiado en **IR_freq**. A medida que el valor de **contador** se incrementa de "0" a "4" en el bucle **for...next**, cada valor sucesivo de la tabla de **lookup** es copiado en **IR_freq**.

```
  for contador = 0 to 4

     lookup contador,[37500,38250,39500,40500,41500],IR_freq
```

Observe que la tabla lookup comienza el barrido de frecuencia a 37500 (más sensitivo) y termina a 41500 (menos sensitivo). Se preguntará por qué los números de la tabla lookup no son los mismos que los valores de frecuencia de la Figura 6.1. Es cierto que si el BASIC Stamp pudiese transmitir un tren de pulsos con un ciclo de trabajo del 50% (pulsos con el mismo tiempo en alto y en bajo) a estas frecuencias, deberíamos trabajar a las frecuencias especificadas por los filtros del detector IR. Sin embargo, el comando **`freqout`** introduce otros factores que afectan la amplitud de los armónicos transmitidos por los LEDs IR. La matemática involucrada en el cálculo de los argumentos de ***`frecuencia`*** óptimos es muy avanzada y está muy lejos del alcance de este libro. Aún así, las mejores frecuencias para una distancia dada pueden determinarse experimentalmente y la Actividad 2 introducirá un método para la deducción y análisis de los datos del barrido de frecuencia, para calibrar los detectores de distancia.

El sensor izquierdo es controlado usando **`freqout`** para enviar el valor actual de **`IR_freq`**. Luego, el argumento **`.lowbit()`** es usado para direccionar cada bit sucesivo en **`valores_izq`**. Cuando **`contador`** es "0", el argumento **`.lowbit(contador)`** direcciona el bit-0 de **`valores_izq`**. Cuando **`contador`** es "1", el argumento **`.lowbit(contador)`** direcciona el bit-1 de **`valores_izq`**, etc. Antes de escribir el valor de **`in8`** en **`valores_izq.lowbit(contador)`**, el operador NOT (~) es usado para invertir el valor del bit. El mismo proceso se repite para **`valores_der`**. Luego de repetir cinco veces el bucle **`for...next`**, los bits de datos IR han sido cargados en **`valores_izq`** y **`valores_der`**.

```
            freqout 7,1,IR_freq
            valores_izq.lowbit(contador) = ~in8

            freqout 1,1,IR_freq
            valores_der.lowbit(contador) = ~in0

         next
```

La subrutina **`mostrar`** usa muchos modificadores y cadenas de textos para mostrar las variables **`valores_izq`** y **`valores_der`**. La primera fila de la pantalla es el encabezado de texto que indica qué medición corresponde al detector IR derecho y cuál al izquierdo. Recuerde que la referencia para izquierdo y derecho se toma como si usted estuviese sentado al volante del Boe-Bot.

```
mostrar:
  debug home, cr, cr, " Lecturas Izquierda     Lecturas Derecha ", cr
```

La segunda fila muestra a **`valores_izq`** y **`valores_der`** en formato binario. Esto permite observar como se modifican los valores de los bits en **`valores_izq`** y **`valores_der`** a medida que la distancia aparente a un objeto cambia.

```
  debug "  ",bin8 valores_izq, "          ", bin8 valores_der, cr
```

La tercer fila muestra el valor **ncd** de cada variable. El operador **NCD** entrega un valor que corresponde a la ubicación de bit más significativo (de mayor peso) de una variable. Si una variable está compuesta de ceros, **ncd** entrega cero. Si el bit menos significativo contiene un "1" y el resto de los bits son ceros, **NCD** entrega un "1". Si el bit-1 contiene un "1", pero todos los bits a la izquierda del bit-1 son ceros, **ncd** entrega un "2" y así sucesivamente. El operador **NCD** es un práctico indicador de cuántos bits se han cargado en **valores_izq** y **valores_der**. Lo bueno de esto es que **ncd** indica directamente en que zona se ha detectado el objeto.

```
debug "  ",dec5 ncd(valores_izq), "       ", dec5 ncd(valores_der), cr, cr
```

Cuando la rutina mostrar termina de enviar datos a la Debug Terminal, el control del programa regresa a la etiqueta **principal**.

```
goto principal
```


Su Turno

- Con el Programa 6.1 ejecutándose, coloque al Boe-Bot de frente a una pared, con los LEDs IR a aproximadamente 1.5 cm de la pared. Para obtener mejores resultados, pegue una hoja blanca contra la pared.
- Tome nota de las lecturas de los pares izquierdo y derecho.
- Comience a alejar el Boe-Bot de la pared.
- Cada vez que el valor de alguno de los sensores disminuya, tome nota de la distancia. De esta forma puede determinar las zonas para cada uno de los pares IR de su Boe-Bot.
- Si las medidas de un lado son mucho mayores que las del otro, puede desviar ligeramente el LED IR del par que registra mayores distancias. Por ejemplo, si el par IR izquierdo siempre entrega lecturas mayores que las del par IR derecho, desvíe el LED IR y el detector un poco más a la izquierda.

Actividad 2: El Detector de Bordes

Una aplicación de los detectores de distancia es controlar la presencia de bordes. Por ejemplo, si el Boe-Bot se está desplazando sobre una mesa, puede cambiar de dirección si ve el borde de la misma. Todo lo que debe hacer es apuntar los pares IR hacia abajo de forma que apunten hacia la porción de mesa que se encuentra al frente del Boe-Bot. Se puede usar un programa de medición de distancia para detectar cuándo se aproxima el borde de la mesa. Cuando el Boe-Bot se acerca al borde, uno o ambos detectores avisarán que no ven más mesa por delante. Esto significa que es momento de huir del abismo.

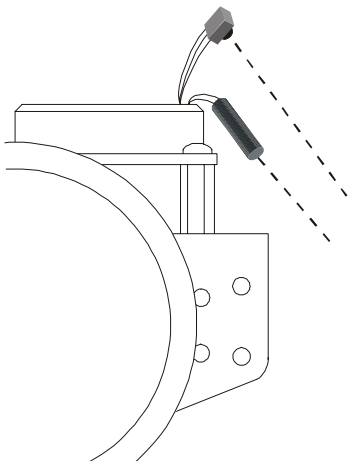


Figura 6.3: LED IR ajustado para detectar bordes.

- Apunte sus pares IR a la superficie de la mesa al frente del Boe-Bot como se muestra en la Figura 6.3. Los pares IR deberían apuntarse a 45° por debajo de la horizontal y a 45° hacia fuera del eje central del Boe-Bot.
- Realice las pruebas de abajo con el Programa 6.1 antes de cargar el Programa 6.2.

> ✓ **TIP** En las actividades del Capítulo 6 use primero los números que aparecen en cada programa de ejemplo. Si usó el Apéndice H para calibrar los sensores de distancia de su Boe-Bot, coloque los valores que determinó únicamente después de usar los valores originales de los programas de ejemplo.

- Registre las salidas de los pares IR cuando el Boe-Bot está sobre la mesa. Si los valores de los pares IR son de "3" o más, indica que los detectores están viendo lo que se suponía que vieran.
- Registre las salidas de los pares IR cuando el Boe-Bot se encuentra con el borde de la mesa. Si estos valores son menores que "3", el Boe-Bot está listo para el Programa 6.2.
- Si el Boe-Bot no entrega lecturas consistentes de "3" o más cuando está sobre la mesa, intente ajustar la dirección a la que apuntan los pares IR. Además, si el Boe-Bot no tiene lecturas menores de "3" cuando se enfrenta al borde de la mesa, también deberán realizarse ajustes sobre los pares IR.

- Si los sensores indican "3" o más cuando está sobre la mesa y "2" o menos cuando se enfrenta con el borde, el Boe-Bot está listo para el Programa 6.2.

- Si los ajustes sobre los componentes no dan ningún resultado, pruebe las instrucciones del Apéndice H y luego repita los pasos anteriores.

Asegúrese de no descuidar su Boe-Bot cuando ejecuta el Programa 6.2, siempre esté listo para atraparlo cuando se acerca al borde de la mesa. Si el Boe-Bot no reconoce el borde y se acerca demasiado con claras intenciones de no detenerse, deténgalo antes de que salte al vacío. De otra forma su Boe-Bot se convertirá en un No-Bot.

Cuando supervisa su Boe-Bot mientras esquiva los bordes de la mesa, esté listo para tomarlo desde arriba. Caso contrario, el Boe-Bot verá sus manos y no el borde y no se comportará como se esperaba.

Programación de Detección de Bordes

El Programa 6.2 usa versiones modificadas de las rutinas adelante, giro_der, giro_izq y atrás, que han sido usadas y vueltas a usar en cada capítulo, desde el Capítulo 2. La cantidad de pulsos de cada rutina ha sido ajustada para obtener un mejor rendimiento en las cercanías del borde de la mesa. La subrutina **`control_sensores`** toma mediciones de distancia reciclando el código del Programa 6.1: Medidor de Distancia IR.

- Ejecute y pruebe el Programa 6.2. Recuerde, esté permanentemente dispuesto a levantar su Boe-Bot si intenta salirse de la mesa.

```
' ¡Robótica! v1.5, Programa 6.2: Detector de Bordes

'----- Declaración --------------

  contador          var    nib          ' Índice del bucle for...next.
  valores_izq       var    word         ' Amacena valores sensores derecha.
  valores_der       var    word         ' Amacena valores sensores izquierda.
  frec_IR_i         var    word         ' Frecuencia de IR izquierdo.
  frec_IR_d         var    word         ' Frecuencia de IR derecho.

'----- Inicialización ------------

   low 13                               ' Valores iniciales de los servos.
```

```
   low 12
   output 2                          ' Declara pines freqout como salidas.
   output 7
   output 1
   freqout 2,500,3000                ' Indicador de reset.

'----- Rutina Principal --------------

principal:                           ' Rutina principal

  ' El comando “gosub control_sensores” envía el programa a una subrutina que
  ' carga los valores de distancia en valores_izq y valores_der.  Así, cuando el
  ' programa regresa de la subrutina control_sensores, los valores están
  ' actualizados listos para tomar decisiones que involucren distancia.

  gosub control_sensores

  ' Las distancias se controlan con cuatro desigualdades distintas.  Dependiendo de
  ' la condición que se cumpla, el programa salta a la rutina de navegación
  ' adelante, giro_izq, giro_der o atras.

  if valores_izq >= 3 and valores_der >= 3 then adelante
  if valores_izq >= 3 and valores_der < 3 then giro_izq
  if valores_izq < 3 and valores_der >= 3 then giro_der
  if valores_izq < 3 and valores_der < 3 then atras

goto principal                       ' Repite el proceso.

'----- Rutinas de Navegación -------

   adelante:                         ' Emite un único pulso adelante, luego
      pulsout 13,850
      pulsout 12,650
      pause 10
   goto principal                    ' regresa a la etiqueta principal:.

   giro_izq:                         ' Emite ocho pulsos a la izq., luego
      for contador = 0 to 8
         pulsout 13,650
         pulsout 12,650
         pause 20
      next
   goto principal                    ' regresa a la etiqueta principal:.

   giro_der:                         ' Emite ocho pulsos a la derecha, luego
      for contador = 0 to 8
         pulsout 13,850
```

```
         pulsout 12,850
         pause 20
      next
   goto principal                      ' regresa a la etiqueta principal:.

   atras:                              ' Emite ocho pulsos hacia atrás, luego
      for contador = 0 to 8
         pulsout 13,650
         pulsout 12,850
         pause 20
      next
   goto principal                      ' regresa a la etiqueta principal:.

'----- Subrutinas ---------------

  ' La subrutina de control de sensores es una versión modificada del Programa 6.1
  ' sin usar la Debug Terminal.  En lugar de mostrar valores_izq y valores_der,
  ' la rutina principal los usa para determinar en que sentido moverse.

  control_sensores:

    valores_izq = 0                   ' Pone a valores_izq y valores_der en 0.
    valores_der = 0

    ' Carga las salidas de los sensores en valores_izq y valores_der usando un
    ' bucle for...next, una tabla lookup y direccionamiento de bits.

      for contador = 0 to 4

         control_sensor_izq:
            lookup contador,[37500,38250,39500,40500,41500],frec_IR_i
            freqout 7, 1, frec_IR_i
            valores_izq.lowbit(contador) = ~ in8

         control_sensor_der:
            lookup contador,[37500,38250,39500,40500,41500],frec_IR_d
            freqout 1, 1, frec_IR_d
            valores_der.lowbit(contador) = ~ in0

      next

    ' Convierte a valores_izq y valores_der de binario a formato ncd.

      valores_izq = ncd valores_izq
      valores_der = ncd valores_der
```

```
    ' Ahora valores_izq y valores_der almacenan un número entre 0 y 5 que repre-
    ' senta la zona donde se detectó un objeto.  El programa ahora puede regresar a
    ' la parte de la rutina principal que toma las decisiones basándose en estas
    ' mediciones de distancia.

  return
```

Cómo Funciona el Programa Detector de Bordes

Lo primero que hace la rutina **`principal`** es llamar a la subrutina **`control_sensores`**. Observe que **`control_sensores`** es simplemente el Programa 6.1 sin la Debug Terminal en una subrutina. En lugar de mostrar los valores **`NCD`** de **`valores_izq`** y **`valores_der`**, éstos se convierten en valores **`ncd`** con las instrucciones:

```
valores_izq = ncd valores_izq
```

y

```
valores_der = ncd valores_der
```

Después de llamar la subrutina **`control_sensores`**, **`valores_izq`** y **`valores_der`** son números entre "0" y "5". Estos valores reemplazan al "1" y "0" usados en los programas de los bigotes. Cuando el programa regresa de la subrutina **`control_sensores`**, se controlan **`valores_izq`** y **`valores_der`** para ver si se ha detectado algún borde de la mesa.

```
if valores_izq >= 3 and valores_der >= 3 then adelante
if valores_izq >= 3 and valores_der < 3 then giro_izq
if valores_izq < 3 and valores_der >= 3 then giro_der
if valores_izq < 3 and valores_der < 3 then atras
```

Los argumentos de ***`inicio`*** y ***`final`*** de los bucles **`for...next`** que emiten los pulsos en las cuatro rutinas de navegación, han sido ligeramente modificados. Se han reubicado en la sección **`Rutinas de Navegación`**. Esto simplifica los ajustes en el comportamiento de las rutinas debido a que están todas en el mismo lugar.

Su Turno

- Ajuste los argumentos ***`inicio`*** y ***`final`*** de los bucles **`for...next`** de las rutinas de navegación, hasta optimizar el comportamiento de evasión de bordes.

Actividad 3: Boe-Bot Seguidor

Para que un Boe-Bot siga a otro, el Boe-Bot seguidor, debe conocer a qué distancia está el vehículo que va adelante. Si el seguidor se retrasa, debe detectarlo y apurar el paso. Si el seguidor se acerca demasiado al otro vehículo, debe detectarlo y aminorar su marcha. Si se encuentra a la distancia correcta, puede esperar a que las mediciones le indiquen que está muy lejos o muy cerca nuevamente.

Se puede usar control proporcional para realizar esta tarea, tomando la diferencia entre la distancia deseada y la medida, ajustando luego los anchos de pulsos enviados a los servos, para corregir esta diferencia. La Figura 6.4 muestra cómo trabaja el control proporcional. La distancia medida se resta de la prefijada, que es llamada set point (o punto de ajuste). Efectuando la resta, se obtiene el error de distancia. Para lograr que la distancia medida iguale a predeterminada, hay que multiplicar el error por un número llamado constante de proporcionalidad, K_p.

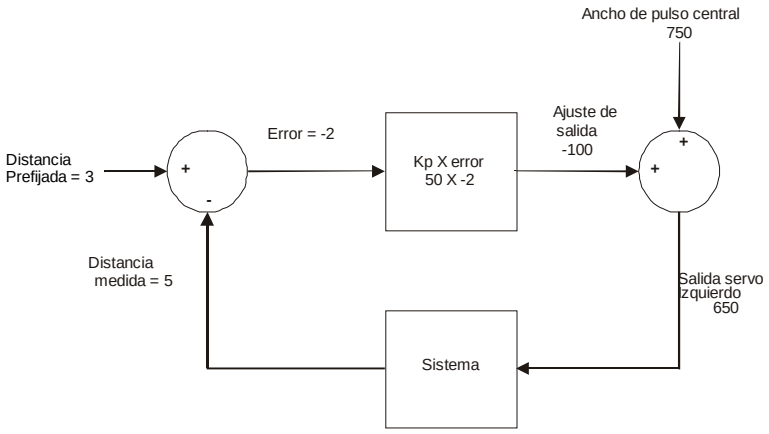


Figura 6.4: Diagrama en bloques del control proporcional del Boe-Bot.

La Figura 6.4 nuestra un ejemplo específico para el par IR y el servo izquierdos. El set point es "3"; la distancia es "5" y la constante de proporcionalidad es $K_p = 50$. Comenzamos por el círculo de la izquierda del diagrama en bloques, donde la distancia medida es restada de la distancia predeterminada. Dado que la distancia set point es "3" y la medida es "5," el valor del error es–2. Una señal de error de –2 significa que el Boe-Bot seguidor debe alejarse del líder, debido a que está ¡peligrosamente cerca! El error entra en un bloque donde se lo multiplica por la constante de proporcionalidad $K_p = 50$. Así que para nuestro ejemplo se realiza la siguiente operación: $50 \times (-2) = -100$. El círculo de la derecha suma –100 más el ancho de pulso central del servo izquierdo. El resultado es que se le resta 100 al valor del ancho del pulso central. El efecto resultante es una ancho de pulso para el servo izquierdo de 650, que representa máxima velocidad hacia atrás. La matemática para la operación de este control proporcional es bastante simple.

$$\text{ancho\_izq} = \text{centro} + (K_p \times \text{error})$$
$$= 750 + (50 \times (-2))$$
$$= 650$$

El servo derecho y su par IR tienen un algoritmo similar, a excepción de que el valor del ajuste de salida se resta en lugar de sumarse a la duración del ancho de pulso central de 750 (1.5 ms). Asumiendo que se haya leído el mismo valor en el par IR derecho, la salida ajusta el ancho del pulso a 850. El resultado final es que el Boe-Bot aplica un pulso de máxima velocidad hacia atrás. Este pulso en reversas es parte de un sistema complejo que depende de factores desconocidos relacionados con la velocidad de movimiento del vehículo líder. La realimentación se cumple cuando el Boe-Bot seguidor vuelve a medir la distancia. Luego el bucle de control se repite una y otra vez.

Programando el Boe-Bot Seguidor

El Programa 6.3 reproduce el bucle de control proporcional que se acaba de explicar entre pulsos del servo. En otras palabras, antes de cada pulso, se mide la distancia y se determina una señal de error. Luego el error es multiplicado por K_p y el valor resultante se suma o resta a los anchos de pulso de los servos izquierdo o derecho.

- Ejecute el Programa 6.3.
- Apunte el Boe-Bot hacia una hoja de papel colocada enfrente de él, como si fuera una pared. El Boe-Bot debería mantener una distancia fija entre él y la hoja de papel.
- Rote ligeramente la hoja de papel. El Boe-Bot debería rotar junto a ella.
- Mueva la hoja de papel para guiar el Boe-Bot por los alrededores. El Boe-Bot debería seguirla.

```
' ¡Robótica! v1.5, Programa 6.3: Vehículo Seguidor

'----- Declaración --------------
                                ' Constantes
  kp_der      con           50    '     Constante proporcional servo derecho.
  kp_izq      con           50    '     Constante proporcional servo izquierdo.
  set_point   con           3     '     Fija valor de distancia entre 0 y 5.
                                ' Variables
  contador          var     nib   '     Índice del bucle for...next.
  valores_izq       var     word  '     Amacena valores sensores izquierda.
  valores_der       var     word  '     Amacena valores sensores derecha.
  frec_IR_i         var     word  '     Frecuencia de IR izquierdo.
  frec_IR_d         var     word  '     Frecuencia de IR derecho.
```

```
'----- Inicialización ------------

   output 13                    ' Declara salidas.
   output 12
   output 2
   output 7
   output 1

   freqout 2,500,3000           ' Indicador de reset.

'----- Rutina Principal --------------

principal:                      ' Rutina principal

  gosub control_sensores        ' Obtiene valores de distancia para cada sensor

  valores_izq = kp_izq * (set_point - valores_izq)   ' Control proporcional izq.
  valores_der = kp_der * (set_point - valores_der)   ' Control proporcional der.

  pulsout 13,750 + valores_izq   ' Pulsos a los servos
  pulsout 12,750 - valores_der

  pause 10                      ' Pausa de 10 ms.

goto principal                  ' Bucle Infinito.

'----- Subrutina(s) -------------

control_sensores:

  valores_izq = 0               ' Pone distancias en 0.
  valores_der = 0
  ' Toma 5 mediciones de distancia en cada par IR.  Si ajustó sus frecuencias en
  ' la Actividad 2, use sus valores en las tablas lookup.

  for contador = 0 to 4
    control_sensor_izq:
      lookup contador,[37500,38250,39500,40500,41000],frec_IR_i
      freqout 7,1,frec_IR_i
      valores_izq.lowbit(contador) = ~in8

    control_sensor_der:
      lookup contador,[37500,38250,39500,40500,41000],frec_IR_d
      freqout 1,1,frec_IR_d
      valores_der.lowbit(contador) = ~in0

  next
```

```
  valores_izq = ncd valores_izq     ' Valor (0 a 5) para distancia.
  valores_der = ncd valores_der

return
```

Cómo Funciona el Programa Vehículo Seguidor

El Programa 6.3 declara tres constantes, **Kp_der**, **Kp_izq** y **set_point** usando la directiva **con**. Cada vez que vea **set_point**, en realidad se trata del número "3" (una constante). Por lo mismo, cada vez que vea **Kp_der** o **Kp_izq** en el programa, en realidad se trata del número 50. La conveniencia del uso de declaración de constantes es que cuando se cambia el valor de la declaración, automáticamente se actualizan los valores que están distribuidos por todo el programa. Por ejemplo, si se cambia el valor de la directiva **Kp_izq con** de 50 a 40, cada lugar del programa en el que aparezca **Kp_izq** cambia de 50 a 40. Esto es extremadamente útil cuando se experimenta o ajusta el funcionamiento de los bucles de control proporcional izquierdo y derecho.

```
declaraciones:
  Kp_der          con         50
  Kp_izq          con         50
  set_point       con         3
```

Lo primero que hace la rutina **principal** es llamar a la subrutina **control_sensores**. Cuando finaliza la subrutina **control_sensores**, **valores_izq** y **valores_der** contienen un número que corresponde a la zona en la que detectaron un objeto los pares IR izquierdo y derecho.

```
principal:
  gosub control_sensores
```

Las dos líneas siguientes de código implementan los cálculos de error y corrección de cada servo. El cálculo de error de cada lado está representado por la resta que está entre paréntesis y el factor de corrección se obtiene al multiplicar los términos anteriores por **Kp_izq** y **Kp_der**.

```
  valores_izq = kp_izq * (set_point - valores_izq)
  valores_der = kp_der * (set_point – valores_der)
```

El cálculo de ajuste de salida se realiza directamente en los argumentos ***duración*** de **pulsout**.

```
  pulsout 13,750 + valores_izq
  pulsout 12,750 - valores_der
```

El tiempo de la pausa se reduce para tener en cuenta el tiempo que tomaron las mediciones.

```
pause 10
```

Luego el control del programa regresa a la etiqueta **`principal:`** y el bucle de realimentación proporcional se repite.

```
goto principal
```


Su Turno

La Figura 6.5 muestra un Boe-Bot líder seguido por un Boe-Bot seguidor. El Boe-Bot líder está ejecutando una versión modificada del Programa 5.3: Exploración IR y el Boe-Bot seguidor una versión modificada del Programa 6.3: Vehículo Seguidor. El control proporcional hace que el Boe-Bot seguidor sea muy eficiente. Un Boe-Bot líder puede guiar un trencito de 6 o 7 Boe-Bots. Simplemente agregue los paneles laterales y de cola al Boe-Bot líder y a los que tengan otro detrás y agregue el resto de los Boe-Bots seguidores formando el tren.

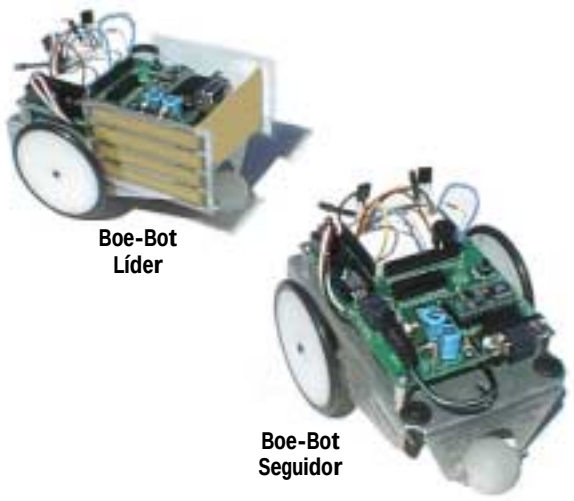


Figura 6.5: Boe-Bot líder y seguidor.

- Si usted no es miembro de una clase y tiene solamente un Boe-Bot, ejecute el Programa 6.3 con K_p igual a 100. Observe la diferencia en la respuesta del Boe-Bot cuando siga la hoja de papel. Observe también si sus correcciones son más abruptas.

- Si es miembro de una clase, coloque paneles de papel a los costados y en la parte trasera de su Boe-Bot como se muestra en la Figura 6.5.

- Programe el Boe-Bot líder para evitar objetos usando el Programa 5.3 con las siguientes modificaciones a las **`Rutinas de Navegación`**:

  - Aumente todos los argumentos de ***`duración`*** de ancho de pulso de 650 a 700.
  - Reduzca todos los argumentos de ***`duración`*** de ancho de pulso de 850 a 800.

- Modifique el Programa 6.3 ajustando los valores de **`Kp_der`** y **`Kp_izq`** a 100.

- Cuando ambos programas se estén ejecutando en los Boe-Bots, coloque el Boe-Bot detrás del Boe-Bot líder. Con un poco de calibración, el Boe-Bot seguidor será muy eficiente.

- Si el Boe-Bot seguidor se mantiene demasiado alejado del líder, apunte los pares IR un poco más hacia afuera. El par IR derecho debería ser puesto un poco más hacia la derecha. El par IR izquierdo debería ser puesto un poco más a la izquierda.

- Algunas de las calibraciones manuales que acaba de hacer, se pueden realizar por software. Modifique el programa del Boe-Bot seguidor, cambiando el valor de la variable **`set_point`** de "3" a "2" pruebe su efecto. También pruebe con un **`set_point`** de "4".

- Modifique el Programa 6.3 volviendo a poner a **`Kp_der`** y **`Kp_izq`** en 50.

La ventaja de reducir las constantes de proporcionalidad es que suavizará los cambios de dirección y ajustes de velocidad del Boe-Bot seguidor. La desventaja es que se corre peligro de que el Boe-Bot seguidor no sea capaz de alcanzar al líder, si éste está configurado para funcionar a máxima velocidad.

- Experimente con la velocidad de exploración del Boe-Bot líder. Cambie los valores de las constantes de proporcionalidad del Boe-Bot seguidor a 50, 75 y 100. Pruebe incrementar las ***`duraciones`*** de ancho de pulso en las **`Rutinas de Navegación`** del Boe-Bot líder, en incrementos de "5" (comenzando con las ***`duraciones`*** de **`pulsout`** de 800/700) para cada una de las tres constantes de proporcionalidad usadas en el Boe-Bot seguidor.

Sumario y Aplicaciones

Se presentó una técnica usada frecuentemente en electrónica llamada barrido de frecuencia. El Boe-Bot fue programado para realizar un barrido de frecuencia sobre los detectores IR, que tienen filtros pasa-banda internos. Cada filtro pasa-banda tiene una frecuencia central de 38.5 kHz. La respuesta en frecuencia de cada filtro pasa-banda se usó para indicar la distancia a los objetos encontrados por el detector.

Se trató también la realimentación a lazo cerrado y el control proporcional. El control proporcional en un sistema de lazo cerrado es un algoritmo donde el error es multiplicado por una constante de proporcionalidad para determinar la salida del sistema. El error se obtiene de restar el set point menos la salida del sistema de medición. En el Boe-Bot, la salida del sistema y el set point estaban en términos de distancia. El BASIC Stamp fue programado en PBASIC para realizar esta operación, tomando repetidas mediciones y recalculando el ajuste de salida. Cada 20 ms, el Boe-Bot hace un muestreo de la distancia y recalcula una salida para el servo proporcional a la señal de error.

Ejemplo del Mundo Real

Los filtros se usan en una gran variedad de aplicaciones de radio, televisión, comunicaciones celulares, audio de alta calidad y comunicaciones IR, entre otras. Se dispone de una gran variedad de filtros con muchas características de respuesta en frecuencia diferentes. La mecatrónica, electrónica analógica y digital, e incluso la mecánica usan muchas aplicaciones de filtros electrónicos y sus técnicas de diseño. Un ejemplo de mecatrónica donde un filtro nos sería útil, es en un detector de vibración. Asumiendo que ciertas frecuencias de vibración podrían dañar las partes móviles de una máquina, podemos detectarlas con un sensor de vibración común, cuya salida entra a un filtro electrónico que busca la presencia de la frecuencia dañina. Cuando se detecta la vibración, el filtro pasa la señal a un subsistema que se encargará de efectuar la corrección automática.

Hablando de corrección automática, el control proporcional es el tema inicial de un curso completo de control con realimentación a lazo cerrado aplicado a una gran variedad de sistemas industriales, electrónicos y mecatrónicos. El control on-off es la primera línea de defensa cuando se trata de mantener un sistema en un nivel determinado, como por ejemplo el líquido dentro de un tanque, un sistema de control de presión o de temperatura. El control proporcional es la segunda línea de defensa. Luego vienen varias mezclas de control proporcional, integral y derivativo que se introducen en el manual Control Industrial también de Stamps en Clase, disponible para bajar gratuitamente en www.stampsenclase.com.

Aplicaciones para el Boe-Bot

La detección IR de objetos ahora puede ser usada para determinar la distancia, así como la presencia de un objeto. Agregue esta característica a la de detección de intensidad de la luz y detección táctil de objetos,

aumentando la capacidad del Boe-Bot para percibir el mundo que lo rodea. Este texto también introdujo varias técnicas de navegación para seguir líneas, luces, u otros Boe-Bots. También se trataron técnicas de navegación para lograr que el Boe-Bot se escape de ciertas situaciones, como obstáculos y precipicios. Más importante aún, este manual mostró varias formas de volver a utilizar técnicas de programación de sensores y navegación. La forma en la que estas técnicas fueron usadas e incorporadas en los programas PBASIC dependió de cada situación particular. Lo importante es que usted ahora está listo para intentar mezclar y compatibilizar técnicas de navegación y de uso de sensores, en una competencia de Boe-Bots. Vea el Apéndice H: Reglas de Competencias para Boe-Bot.

Una gran variedad de técnicas de sistemas de control, electrónica y sensores se tratan en otros manuales de Stamps en Clase. Si está interesado en agregar más trucos a su Boe-Bot o planea ingresar en las competencias de Micromouse, Robot Sumo o Robot Bombero, encontrará útiles las técnicas y habilidades introducidas en estos libros, todos disponibles para bajar gratuitamente de www.stampsenclase.com:

- ¿Qué es un Microcontrolador?: Cubre las bases de los proyectos de electrónica microcontrolada con el BASIC Stamp. Se usa la programación en PBASIC para controlar sensores y periféricos elementales con el BASIC Stamp.

- Analógico y Digital Básicos: El BASIC Stamp se está volviendo muy popular como cerebro de muchos diseños electrónicos inteligentes. Este texto presenta técnicas de conversión analógica-a-digital y digital-a-analógica, que se relacionan con el uso del BASIC Stamp. También se presentan algunos de los protocolos de comunicación fundamentales para usar conversores periféricos.

- Mediciones Ambientales: El BASIC Stamp también puede ser usado en muchos dispositivos de almacenamiento de datos y data loggers. Este texto se sumerge en el mundo del uso de sensores electrónicos inteligentes y en la física de mediciones ambientales precisas. Todos estos conceptos están aplicados a ambientes microcontrolados. Los ejemplos incluidos se pueden desarrollar para exploración submarina, hidroponia y gran variedad de desafíos científicos.

- Control Industrial: Este texto usa muchos proyectos con BASIC Stamps como plataforma de lanzamiento a conceptos y principios fundamentales en el control de procesos industriales. Incluye el uso de StampPlot Lite que es un software que se puede bajar gratuitamente. El programa tiene una interfaz similar a la de un osciloscopio ideal para recopilación de datos y es particularmente útil para ilustrar la respuesta de los sistemas al control del BASIC Stamp, cuando se usan las siguientes técnicas: lazo abierto , diferencial, proporcional, integral y derivativo.

Preguntas y Proyectos

Preguntas

1. ¿Cuál es el efecto de los filtros electrónicos del detector IR sobre su capacidad de detectar señales IR reflejadas en los objetos?

2. ¿Cómo posibilita el filtro electrónico del detector IR, reconocer la distancia a un objeto?

3. ¿Qué tipo de datos son necesarios para calibrar el barrido de frecuencia? ¿Cuál es la variable de entrada que es modificada? ¿Cuál es el dato de salida que es examinado? ¿Qué decisiones se toman teniendo en cuenta los datos de salida?

4. ¿Qué declaración de variable y modificaciones en los bucles serían necesarias para lograr un barrido de frecuencia de 16 valores en cada par IR?

5. ¿Cómo se puede usar la medición de distancia para detectar el borde de una mesa?

6. ¿Cómo puede el Boe-Bot usar la medición de distancia para mantener una distancia constante de un objeto estacionario? ¿Hay alguna diferencia en el programa si es necesario que el Boe-Bot mantenga una distancia constante de un objeto en movimiento?

7. Describa el bucle de realimentación del control proporcional. ¿Cómo se aplica este bucle en el caso del ejercicio del Boe-Bot seguidor?

Ejercicios

1. Si se necesitan siete zonas, determine cuántas frecuencias de medición se requerirán.

2. ¿Cuántas frecuencias de medición se necesitarán para 16 zonas?

3. Modifique el Programa 6.3 para que el Boe-Bot pueda controlar 16 zonas por par IR. Ajuste este código para que mantenga una pausa de 20 ms entre pulsos de los servos.

4. Escriba un segmento de código que le permita al Boe-Bot detectar y seguir el borde de una mesa. Pista: el control proporcional será de ayuda en este ejercicio.

Proyectos

1. Programe el Boe-Bot para que siga una línea negra sobre un fondo blanco. Pista: Una línea negra se verá a través de los sensores IR como un objeto muy lejano. Esto involucrará modificar el Programa 6.2 para que sea atraído por objetos lejanos.

2. Agregue algoritmos de aceleración a los Programas 6.2 y 6.3 y luego pruébelos.

3. Programe el Boe-Bot para que siga el borde de una línea negra. Esto es diferente a seguir la línea negra con ambos sensores. El Boe-Bot debería buscar la transición entre blanco y negro. Cuando lo haya logrado, el Boe-Bot debería ser capaz de seguir el borde derecho o el izquierdo de una línea negra.

4. Si tuvo éxito en el Proyecto 3, intente resolver un laberinto que esté delimitado por cintas negras. Pista: la navegación por EEPROM ayudará en ciertas maniobras, como dar vuelta a una esquina. El queso del laberinto es un haz de luz apuntado hacia la superficie blanca. Cuando el Boe-Bot detecta el queso, debería detenerse y hacer titilar un LED 10 veces. Un buen detector de queso podría ser un simple circuito con un fotorresitor.

5. Desafío: Se pueden agregar los bigotes al Boe-Bot, para que detecte obstáculos, para mejorar la navegación por el laberinto que realizó en el Proyecto 4. La información para hacer esto la encontrará en el Capítulo 3, Proyecto 3.

Apéndice A: Lista de Componentes y Suministros

Kit Completo de Robótica

Además de una PC y las herramientas básicas listadas en la Página 7, el Kit Completo de Robótica incluye todos los componentes y documentación que necesitará para completar los experimentos de este texto.

Kit Completo de ¡Robótica! (#28132)

Código Parallax	Descripción	Cantidad
BS2-IC	Módulo BASIC Stamp II	1
27919	BASIC Stamp Manual Version 1.9 (en Inglés)	1
28124	Kit de Componentes de ¡Robótica!	1
28125	Manual ¡Robótica! v1.5 (en Inglés)	1
550-00010	Plaqueta de Educación Rev B	1
800-00003	Cable Serial	1

Kit de Componentes de ¡Robótica!

Si ya tiene una BOE y un BASIC Stamp, el Kit de Componentes de ¡Robótica! Se puede comprar por separado. Como con todos los manuales de Stamps en Clase, los experimentos de ¡Robótica! Fueron diseñados para ser usados con la Plaqueta de Educación y el BASIC Stamp y todos los componentes del kit con el mismo nombre que el texto. El contenido del Kit de Componentes de ¡Robótica! Se muestra abajo. Se pueden ordenar piezas de repuesto en http://www.stampsenclase.com.

Kit de Componentes de Robótica (#28124)

Código Parallax	Descripción	Cantidad
150-01030	resistores 10K ohm	5
150-02210	resistores 220 ohm	3
150-04710	resistores 470 ohm	2
200-01040	capacitor 0.1 uF	4
200-01031	capacitor 0.01 uF	2
201-03080	capacitor 3300 uF	1
350-00006	LEDs rojos	2
350-00009	fotorresistores (EG&G Vactec VT935G grupo B)	2
350-00013	Receptor infrarrojo (Panasonic PNA4602M o eq.)	2
350-00014	LED infrarrojo recubierto con termocontraible (QT QEC113)	2

Kit de Componentes de Robótica (#28124) Continuación

Código Parallax	Descripción	Cantidad
451-00303	Conector de 3 pines	4
700-00010	Bigotes	2
700-00015	Arandelas de nylon	2
700-00049	Separadores macho/hembra de 3/8"	2
800-00016	Cables de interconexión (paquete de10)	2
900-00001	Parlante Piezoeléctrico	1
28133	**Hardware del Boe-Bot**	
700-00002	tornillos 4-40 x 3/8"	8
700-00009	bolilla 1" de polietileno agujereada	1
700-00011	Cubiertas de goma	2
700-00013	Ruedas plásticas	2
700-00016	tornillos cabeza plana 4-40 x 3/8"	2
700-00022	Chasis de aluminio	1
700-00023	chaveta larga 1/16" x 1.5"	1
700-00023	tuercas autoblocantes 4-40	10
700-00025	13/32" gomas pasacables (hueco ½")	1
700-00026	9/32" gomas pasacables (hueco 3/8")	2
700-00027	separadores roscados 1/2"	4
700-00028	tornillos 4-40 x 1/ 4"	8
700-00038	Porta pilas con cable de conexión	1
900-00003	Servos (Futaba s-148 o Parallax)	2

Kits de la Plaqueta de Educación

Los manuales de Stamps en Clase usan el módulo BASIC Stamp y la Plaqueta de Educación como núcleo. Estos componentes se pueden comprar por separado.

Plaqueta de Educación –Kit Completo (#28102)

Código Parallax	Descripción	Cantidad
28150	Plaqueta de Educación	1
800-00016	Cables de interconexión	10
BS2-IC	Modulo BASIC Stamp II	1
750-00008	Fuente de alimentación 300 mA 9 VCC	1
800-00003	Cable Serial	1

Kit Plaqueta de Educación (#28150)

Código Parallax	Descripción	Cantidad
28102	Plaqueta de Educación y cables de interconexión	1
800-00016	Cables de interconexión	6

Bibliografía de Robótica y el BASIC Stamp

La documentación incluida en el Kit completo de ¡Robótica! También se puede conseguir por separado.

Bibliografía de Robótica

Código Parallax	Descripción	¿Disponibilidad en Internet?
27919	Manual BASIC Stamp Version 1.9	http://www.stampsenclase.com
28125	¡Robótica! v1.5 en Inglés	http://www.stampsenclase.com

Distribuidores Parallax

La red de distribución de Parallax funciona en aproximadamente 40 países por todo el mundo. Algunos de esos distribuidores también son distribuidores autorizados de "Stamps en Clase". Los distribuidores de Stamp en Clase normalmente tienen en stock la Plaqueta de Educación (#28102 y #28150) . También se listan algunas compañías de componentes electrónicos para aquellos clientes que deseen armar sus propios Kits de Componentes.

País	Compañía	Notas
Estados Unidos	**Parallax, Inc.** 3805 Atherton Road, Suite 102 Rocklin, CA 95765 USA (916) 624-8333, fax (916) 624-8003 http://www.stampsinclass.com http://www.parallaxinc.com	Suministra Parallax y Stamp en Clase. Fabricante del BASIC Stamp.
Estados Unidos	**Digi-Key Corporation** 701 Brooks Avenue South Thief River Falls, MN 66701 (800) 344-4539, fax (218) 681-3380 http://www.digi-key.com	Fuente de componentes electrónicos. Distribuidor Parallax. Excelente fuente de componentes. Puede tener en stock la Plaqueta de Educación.
Australia	**RTN** 35 Woolart Street Strathmore 3041 Australia phone / fax +613 9338-3306 http://people.enternet.com.au/~nollet	Distribuidor Parallax y Stamp en Clase.
Australia	**Microzed Computers** PO Box 634 Armidale 2350 Australia Phone +612-67-722-777, fax +61-67-728-987 http://www.microzed.com.au	Distribuidor Parallax. Distribuidor de Stamp en Clase. Excelente soporte técnico.

País	Compañía	Notas
Canadá	**HVW Technologies** 300-8120 Beddington Blvd NW, #473 Calgary, AB T3K 2A8 Canada (403) 730-8603, fax (403) 730-8903 http://www.hvwtech.com	Distribuidor Parallax y Stamp en Clase.
Alemania	**Elektronikladen** W. Mellies Str. 88 32758 Detmold Germany 49-5232-8171, fax 49-5232-86197 http://www.elektronikladen.de	Distribuidor Parallax y Stamp en Clase.
Nueva Zelandia	**Trade Tech** Auckland Head Office, P.O. Box 31-041 Milford, Auckland 9 New Zealand +64-9-4782323, fax 64-9-4784811 http://www.tradetech.com	Distribuidor Parallax y Stamp en Clase.
Inglaterra	**Milford Instruments** Milford House 120 High St., S. Milford Leeds YKS LS25 5AQ United Kingdom +44-1-977-683-665 fax +44-1-977-681-465 http://www.milinst.demon.co.uk	Distribuidor Parallax y Stamp en Clase.

Apéndice B: Solución de Problemas de Comunicación entre PC y Stamp

Cuando se selecciona Identify del menú Run del Stamp Editor, la respuesta esperada es:

"Information: Found BS2-IC (firmware v1.0.)."
("Información: Se encontró BS2-IC (firmware v1.0.).")

Cuando en su lugar encuentra un mensaje de error, a veces puede ser desconcertante. Daremos las descripciones de los mensajes de error más comunes, junto con algunas sugerencias para solucionarlos. Si ninguna de estas sugerencias funciona , asegúrese de haber seguido correctamente todas las instrucciones del Capítulo 1. ¿Aún sin suerte? La información de contacto del Soporte Técnico de Parallax está impresa en la BOE.

"Error: Basic Stamp II detected but not responding...Check power supply."
"Error: BASIC Stamp II detectado pero no responde....Controle la alimentación". El software cree haber encontrado un BASIC Stamp en uno de los puertos com, pero cuando intenta comunicarse con el BASIC Stamp, no obtiene respuesta. "Controle la alimentación" significa que controle el estado del porta pilas y sus pilas.

- Asegúrese de usar pilas alcalinas AA de 1.5 V nuevas y que estén correctamente colocadas en el porta pilas.
- Además, controle que la ficha del porta pilas esté bien conectada a la BOE.

¿El BASIC Stamp está conectado al puerto com que el software cree?

- Para controlar y ajustar la configuración del puerto com, haga clic en Edit, seleccionando Preferences.
- Luego, haga clic en la pestaña Editor Operation.
- Puede modificar el Default Com. Port, colocando el valor del puerto com en el que usted cree que está conectado el BASIC Stamp.

También hay una configuración AUTO que hace que el software busque al BASIC Stamp por todos los puertos com conocidos.

- Después de cada modificación en Default com. port, haga clic en OK y luego ejecute Run | Identify nuevamente.

"Error: BASIC Stamp II not responding...Check serial cable connection. Check power supply."
"Error: BASIC Stamp II no responde...Controle la conexión del cable serial. Controle la alimentación."
Este mensaje significa que el software no puede encontrar al BASIC Stamp. El mensaje puede querer decir que no se encontró al BASIC Stamp en un puerto com particular, o que no encuentra al BASIC Stamp en ningún puerto com (dependiendo de las Preferences).

- Revise si el cable serial está correctamente conectado.
- Intente ajustar las configuraciones del puerto com, al igual que en el mensaje de error anterior.

"Error: BASIC Stamp IISX not responding...Check serial cable connection. Check power supply."
"Error: El BASIC Stamp IISX no responde...Controle la conexión del cable serial. Controle la alimentación."
También podría aparecer en el mensaje el BASIC Stamp II E, o P. Esto significa que el software está buscando un BASIC Stamp 2SX (o 2E o 2P) en lugar de un BASIC Stamp 2. Esto es fácil de solucionar.

- Haga clic en Edit seleccionando luego Preferences.
- Haga clic en la pestaña Editor Operation Tab.
- Modifique el campo Default Stamp Mode para que diga BS2, en lugar de BS2SX o BS2E o BS2P.

Apéndice C: Referencia Rápida de PBASIC

El Parallax BASIC Stamp Manual Version 1.9 (en inglés) consiste de aproximadamente 450 páginas de descripciones de comandos PBASIC, notas de aplicación y esquemas. El documento completo está disponible para descargar en http://www.parallaxinc.com y http://www.stampsinclass.com en formato Adobe PDF, pero también puede ser adquirido por estudiantes e instituciones educativas.

Esta Guía de Referencia Rápida de PBASIC es una versión reducida de los comandos del BASIC Stamp II.

BRANCH

BRANCH *indicador*, [*dirección0, dirección1, …direcciónN*]

Se dirige a la dirección especificada por el indicador (si está en el rango).

- *Indicador* es una variable / constante que especifica a cuál de las direcciones listadas dirigirse (0-N).
- *Direcciones* son las etiquetas que especifican a qué lugar dirigirse.

BUTTON

BUTTON *pin, presionado, retardo, velocidad, espaciotrabajo, estado, etiqueta*

Elimina el rebote, realiza auto-repetir y se dirige a la etiqueta si un botón es activado.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *presionado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *retardo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255.
- *velocidad* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255.
- *espaciotrabajo* es una variable tipo byte o word. Se inicia con 0 antes de ser usada.
- *estado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *etiqueta* es la etiqueta a la que debe saltar el programa cuando se presiona el botón.

COUNT

COUNT *pin, período, resultado*

Cuenta el número de ciclos en un pin, por un *período* de milisegundos y guarda ese número en resultado.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *período* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535.
- *resultado* es una variable tipo bit, nibble, byte o word.

DATA

{*puntero*} DATA {@*ubicación*,} {*WORD*} {*datos*}{(*tamaño*)} {, { *WORD*} {*datos*}{(*tamaño*)}…}

Almacena datos en la EEPROM antes de descargar el programa PBASIC.

- *puntero* es un nombre opcional de una constante no definida o variable tipo bit, nibble, byte o word al que se le asigna el valor del primer lugar de memoria en el que los datos son almacenados.
- *ubicación* es una constante, expresión o variable tipo bit, nibble, byte o word opcional que designa el primer lugar de memoria en el que los datos son escritos.
- *word* es una llave opcional que hace que los datos sean almacenados en la memoria como dos bytes separados.
- *datos* es una constante o expresión opcional a ser escrita en la memoria.
- *tamaño* es una constante o expresión opcional que designa el número de bytes de datos definidos o no, para escribir o reservar en la memoria. Si datos no es especificado, entonces se reserva una cantidad de espacio indefinido y si datos es especificado, se reserva la cantidad especificada por tamaño.

DEBUG
DEBUG *datosalida*{,*datosalida*...}
Envía variables a ser vistas en la PC.

- *datosalida* es una cadena de texto, constante o variable tipo bit, nibble, byte o word. Si no se especifican modificadores DEBUG muestra caracteres en ascii sin espacios ni separación de oraciones a continuación del valor.

DTMFOUT
DTMFOUT *pin*, {*tiempoencendido*, *tiempoapagado*,}[*tono*{,*tono*...}]
Genera pulsos telefónicos DTMF.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *tiempoencendido* y *tiempoapagado* son constantes, expresiones o variables tipo bit, nibble, byte o word en el rango de 0..65535.
- *tono* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

END
END

- Duerme hasta que se interrumpa y reinicie la alimentación o se conecte a la PC. El consumo de energía es reducido a 50 uA.

FOR...NEXT
FOR *variable* = *inicial* to *final* {*paso*} ...NEXT
Crea un bucle repetitivo que ejecuta las líneas de programa entre FOR y NEXT, incrementando o disminuyendo el valor de la variable de acuerdo al *paso*, hasta que el valor de la variable iguala al valor *final*.

- *Variable* es una variable tipo nib, byte, o word usada como contador.
- *Inicial* es una variable o una constante que especifica el valor inicial de la *variable*.

- *Final* es una variable o una constante que especifica el valor final de la *variable*. Cuando el valor de la *variable* supera el valor *final*, end, el bucle FOR . . . NEXT detiene su ejecución y el programa continúa en la instrucción siguiente a NEXT.
- *Paso* es una variable o constante opcional que fija el valor en que aumenta o disminuye la *variable* en cada bucle de FOR / NEXT. Si *inicial* es mayor que *final*, PBASIC2 interpreta que *paso* es negativo, aunque no se use un signo menos.

FREQOUT
FREQOUT *pin*, *milisegundos*, *freq1* {,*freq2*}
Genera una o dos ondas sinusoidales de las frecuencias especificadas (0 – 32.767 hz.).
- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *milisegundos* es una constante, expresión o variable tipo bit, nibble, byte o word.
- *freq1* y *freq2* son constantes, expresiones o variables tipo bit, nibble, byte o word en el rango de 0..32767 que representan las frecuencias correspondientes.

GOSUB
GOSUB *direcciónEtiqueta*
Guarda la dirección de la instrucción siguiente a GOSUB, luego se dirige al punto del programa especificado por *direcciónEtiqueta*.
- *DirecciónEtiqueta* es una etiqueta que especifica a qué lugar dirigirse.

GOTO
GOTO *direcciónEtiqueta*
Se dirige al punto del programa especificado por direcciónEtiqueta.
- *DirecciónEtiqueta* es una etiqueta que especifica a qué lugar dirigirse.

HIGH
HIGH *pin*
Convierte al pin en salida y la pone en estado alto (escribe un 1 en el bit correspondiente de DIRS y OUTS).
- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

IF...THEN
IF *condición* THEN *direcciónEtiqueta*
Evalúa la condición y, si es verdadera, se dirige al punto del programa marcado por *direcciónEtiqueta*.
- *Condición* es una expresión, tal como "x=7", que puede ser evaluada como verdadera o falsa.
- *DirecciónEtiqueta* es una etiqueta que especifica dónde ir en el caso que la condición sea verdadera.

INPUT
INPUT *pin*
Convierte al pin especificado en entrada (escribe un 0 en el bit correspondiente de DIRS).
- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

LOOKDOWN
LOOKDOWN *valor*, {*comparador*,} [*valor0, valor1,... valorN*], *variable*
Compara un valor con los de la lista en función del *comparador* y guarda la ubicación (índice), en la variable.
- *Valor* es una constante, expresión o variable tipo bit, nibble, byte o word.
- *Comparador* es =, <>, >, <, <=, =>. (= es la opción por defecto).
- *Valor0, valor1*, etc. son constantes, expresiones o variables tipo bit, nibble, byte o word.
- *Variable* es una variable tipo bit, nibble, byte o word.

LOOKUP
LOOKUP *índice*, [*valor0, valor1,... valorN*], *variable*
Busca el valor especificado por el índice y lo guarda en la variable. Si el índice excede el máximo valor de índice de la lista, la variable no es afectada. Un máximo de 256 valores puede ser incluido en la lista.
- *Índice* es una constante, expresión o variable tipo bit, nibble, byte o word.
- *Valor0, valor1*, etc. son constantes, expresiones o variables tipo bit, nibble, byte o word.
- *Variable* es una variable tipo bit, nibble, byte o word.

LOW
LOW *pin*
Convierte al pin en salida y la pone en estado bajo (escribe un 1 en el bit correspondiente de DIRS y un 0 en el bit correspondiente de OUTS).
- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

NAP
NAP período
Descansa por un corto período. El consumo de energía es reducido a 50 uA (sin cargas conectadas).
- *período* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..7 que representa intervalos de 18ms.

OUTPUT
OUTPUT *pin*
Convierte al pin especificado en salida (escribe un 1 en el bit correspondiente de DIRS).
- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

PAUSE
PAUSE *milisegundos*
Hace una pausa en la ejecución por 0–65535 milisegundos.

- *milisegundos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535.

PULSIN
PULSIN *pin, estado, variable*
Mide un pulso de entrada (resolución de 2 µs).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *estado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *variable* es una variable tipo bit, nibble, byte o word.

Las mediciones se toman en intervalos de 2uS y el tiempo máximo es de 0.13107 segundos.

PULSOUT
PULSOUT *pin, período*
Genera un pulso de salida (resolución de 2 µs).

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *período* es una constante, expresión o variable tipo bit, nibble, byte o word, en el rango de 0..65535 que especifica la duración del pulso en unidades de 2uS.

PWM
PWM pin, duty, ciclos
Puede ser usado para generar voltajes de salida analógicos (0-5V) usando un capacitor y un resistor. Genera una salida por modulación de ancho de pulso y luego el pin vuelve a ser entrada.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *duty* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255.
- *ciclos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..255 que representa la cantidad de ciclos de 1ms a enviar a la salida.

RANDOM
RANDOM *variable*
Genera un número pseudo-aleatorio.

- *Variable* es una variable tipo byte o word en el rango de 0..65535.

RCTIME
RCTIME *pin, estado, variable*
Mide un tiempo de carga / descarga RC. Puede ser usado para medir potenciómetros.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *estado* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1.
- *Variable* es una variable tipo bit, nibble, byte o word.

READ
READ ubicación, variable
Lee un byte de la EEPROM y lo almacena en variable.
- *ubicación* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..2047.
- *Variable* es una variable tipo bit, nibble, byte o word.

RETURN
Regresa de una subrutina. Regresa el programa a la dirección (instrucción) inmediatamente siguiente al GOSUB más reciente.

REVERSE
REVERSE *pin*
Si el pin es de salida, lo hace de entrada. Si el pin es de entrada, lo hace de salida.
- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

SERIN
SERIN *rpin*{*fpin*}, *baudmodo*, {*petiqueta*,} {*tiempoespera*, *tetiqueta*,} [*datosentrada*]
Recibe datos asincrónicamente (como en RS-232).
- *rpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..16.
- *fpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *baudmodo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535.
- *petiqueta* es una etiqueta a la cual saltar en caso de error de paridad.
- *tiempoespera* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 que representa el número de milisegundos a esperar por un mensaje entrante.
- *tetiqueta* es una etiqueta a la cual saltar en caso de demora.
- *Datosentrada* es un conjunto de nombres de variables, expresiones o constantes, separados por comas que pueden tener los formatos disponibles para el comando DEBUG, excepto los formatos ASC y REP. Además, los siguientes formatos están disponibles:

1. STR vector de bytes\L{\E}
2. SKIP L
3. WAITSTR vector de bytes{\L}.
4. WAIT (valor {,valor...}) Esperar por una secuencia de hasta seis bytes.

SEROUT

SEROUT *tpin*{*fpin*}, *baudmodo*, {*pausa*,} {*tiempoespera*, *tetiqueta*,} [*datossalida*]
Envía datos asincrónicamente (como en RS-232).

- *tpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..16.
- *fpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.
- *baudmodo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..60657.
- *pausa* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 que especifica un tiempo (en milisegundos) de retardo entre los bytes transmitidos. Este valor sólo puede ser especificado si no se da un valor a fpin.
- *tiempoespera* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 representa el número de milisegundos a esperar para la señal de transmisión del mensaje. Este valor sólo puede ser especificado si se da un valor a fpin.
- *tetiqueta* es una etiqueta a la cual saltar si se sobrepasa el tiempoespera. Se especifica con fpin.
- *datossalida* es un conjunto de constantes, expresiones y nombres de variables separados por comas y opcionalmente precedido por modificadores disponibles en el comando DEBUG.

SHIFTIN
SHIFTIN *dpin, cpin, modo,* [resultado{*bits*} { ,*resultado*{*bits*}... }]
Convierte los bits recibidos de serie a paralelo y los almacena.

- *dpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de datos.
- *cpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin del reloj (clock).
- *modo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..4 que especifica el modo de transmisión. 0 o MSBPRE = msb primero, pre-reloj, 1 o LSBPRE = lsb primero, pre-reloj, 2 o MSBPOST = msb primero, post-reloj, 3 o LSBPOST = lsb primero, post-reloj.
- *resultado* es una variable tipo bit, nibble, byte o word donde el dato recibido es guardado.
- *bits* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 1..16 que especifica el número de bits a recibir en resultado. El valor predeterminado es 8.

SHIFTOUT
SHIFTOUT *dpin, cpin, modo*, [*datos*{*bits*} {, *datos*{*bits*}... }]
Envía los datos en forma serial.

- *dpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de datos.
- *cpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin del reloj (clock).

- *Modo* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..1 que especifica el modo de transmisión, 0 o LSBFIRST = lsb primero, 1 o MSBFIRST = msb primero.
- *datos* es una constante, expresión o variable tipo bit, nibble, byte o word que contiene el dato a ser enviado.
- *bits* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 1..16 que especifica el número de bits a enviar. El valor predeterminado es 8.

SLEEP
SLEEP *segundos*
Duerme por 1-65535 segundos. El consumo de energía es reducido a 50 uA.

- *segundos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..65535 que especifica el número de segundos a dormir.

TOGGLE
TOGGLE *pin*
Invierte el estado de un pin.

- *pin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15.

WRITE
WRITE ubicación, datos
Escribe un byte en la EEPROM.

- *ubicación* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..2047.
- *datos* es una constante, expresión o variable tipo bit, nibble, byte o word.

XOUT
XOUT *mpin, zpin*, [*casaclavecomando*{*ciclos*} {, *casa**clavecomando*{*ciclos*}... }]**
Genera códigos de control de línea X-10. Se usa con los módulos de interfase TW523 o TW513.

- *mpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de modulación.
- *zpin* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el pin de cruce por cero.
- *casa* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica el código de casa A..P.
- *clavecomando* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 0..15 que especifica las claves 1..16 o es uno de los comandos mostrados en BASIC Stamp Manual Version 1.9. Estos comandos incluyen encender y apagar luces.
- *ciclos* es una constante, expresión o variable tipo bit, nibble, byte o word en el rango de 2..65535 que especifica el número de veces que se transmite un comando. (Predeterminado es 2).

Apéndice D: Construcción de Puertos para Servos en la Plaqueta de Educación Rev. A

Antes de Julio de 2000, Parallax vendía la Plaqueta de Educación Rev A junto con los Kits del Boe-Bot, en lugar de la Plaqueta de Educación Rev B. La Plaqueta de Educación Rev B tiene cuatro puertos para servos para hacer más simple la conexión de servos. Si tiene una Plaqueta de Educación Rev A, deberá construir sus propios puertos para servos, que es bastante fácil. Todos los proyectos de este texto se diseñaron para que no hubiese conflictos con los puertos que usted construirá, siguiendo las instrucciones de este Apéndice.

Los puertos para servos que construiremos reemplazarán a los Puertos 12 y 13 de la Plaqueta de Educación Rev B. Una vez que arme y pruebe los puertos, podrá dejarlos conectados para todas las actividades desde el Capítulo 1, Actividad 6 hasta el fin del libro (Capítulo 6, Actividad 3). Teniendo esto en cuenta, deje armados los puertos de su BOE Rev A entre las distintas prácticas. Cualquier actividad que muestre un esquema con servos conectados a los pines de E/S P12/P13 funcionará bien con el diseño de puertos de la Figura D.2. Cada vez que una imagen muestre los servos de la BOE Rev B conectados a los Puertos 12 y 13, la Figura D.2 será el sustituto para el conexionado de la BOE Rev A.
Así deberá construir sus puertos para servos en la BOE Rev A:

Lista de Componentes:

(1) Plaqueta de Educación Rev A
(1) Capacitor 3300µF
(2) conectores de 3 pines
(2) Servos
(varios) Cables de interconexión

Los conectores de 3 pines del Kit de ¡Robótica! Rev A se muestran en la Figura D.1 (a). Modifique estos conectores para que el plástico que sujeta los tres pines juntos quede en el centro, como se muestra en la Figura D.1 (b).

Figura D.1 (a) Conector sin modificar y (b) conector modificado.

Luego conecte los puertos a la Plaqueta de Educación, como se muestra en la Figura D.2.

- Conecte el capacitor de 3300µF de Vdd a Vss. Asegúrese que la pata más corta del capacitor se conecte a uno de los dos zócalos rotulados Vss, en el conector de 20 pines (20-socket app-mod header). De la misma forma controle que la pata más larga se conecte al zócalo rotulado Vdd del mismo conector.

- Inserte los conectores de tres pines en la protoboard y use cables de interconexión para conectar Vin, Vss, P12 y P13 a la protoboard EXACTAMENTE COMO SE MUESTRA en la Figura D.2.

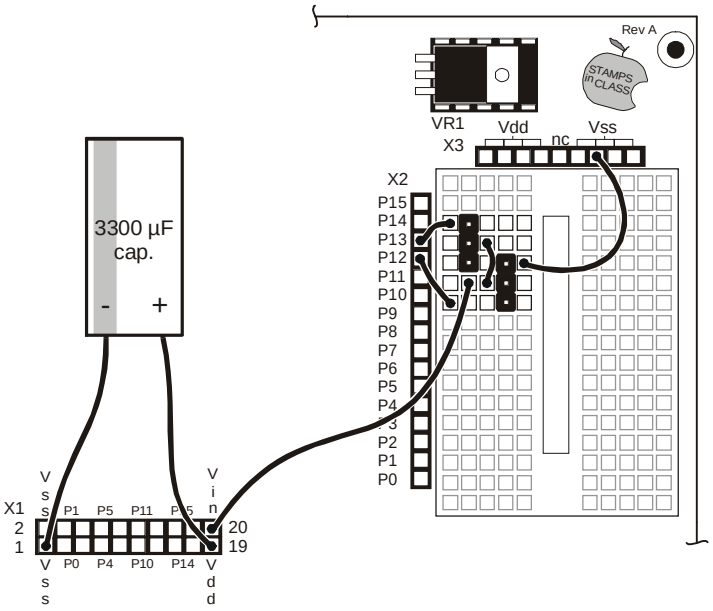


Figura D.2: Conexionado del puerto para servos.

Luego, los servos pueden ser conectados a estos puertos. El conector de tres pines superior es equivalente al Puerto para servos 13 y el inferior es equivalente al Puerto 12.

Asegúrese de seguir la secuencia de colores de la conexión de los servos que se muestra en la Figura D.3. Si los conecta en orden inverso podría dañar los servos, el BASIC Stamp, o las dos cosas juntas.

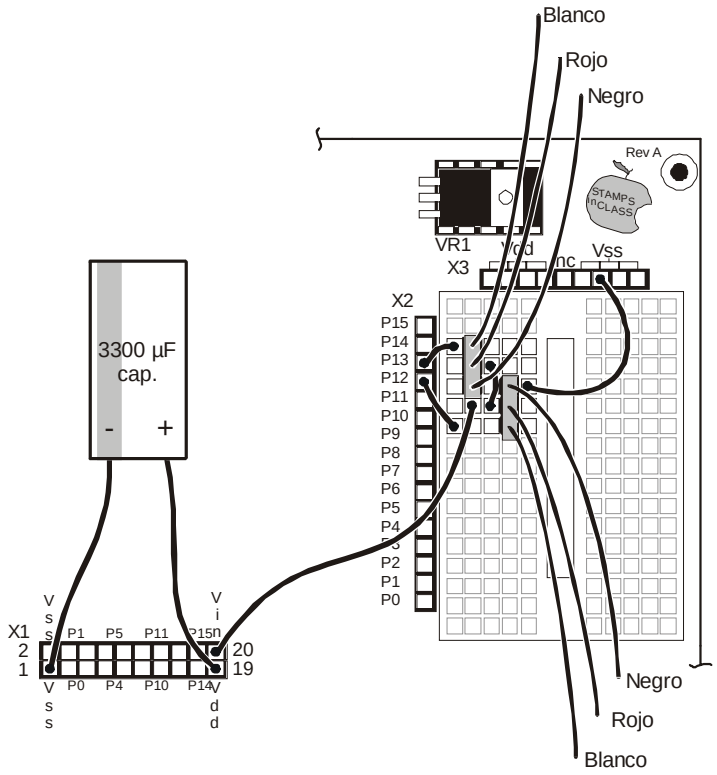


Figura D.3: Código de colores para conectar los servos en sus puertos.

Apéndice E: Cambio del Regulador de Voltaje de la Plaqueta de Educación Rev A

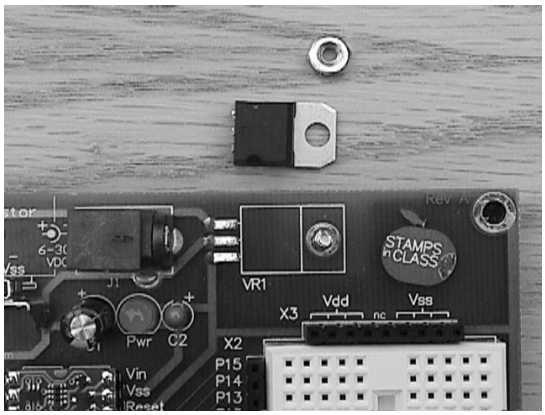
Antes de Junio de 1999, Parallax producía la Plaqueta de Educación con un regulador de voltaje LM7805CV. Este regulador funciona bien para la mayoría de las aplicaciones, pero reiniciará el BASIC Stamp cuando se use con una alimentación de 6 Volt y dispositivos de mucha corriente como servos y motores paso a paso. Si su Plaqueta de Educación tiene el regulador de voltaje LM7805CV, comuníquese con Parallax y pida gratis un Kit de actualización de Regulador de Voltaje de la BOE (código #28151). Este apéndice incluye las instrucciones para quitar el LM7805CV y reemplazarlo con el regulador de baja caída LM2940. Parallax se lo enviará gratuitamente por U.S. Mail.

El regulador de voltaje es el único componente que está atornillado a la plaqueta, al lado del logo "Stamps in Class". Para poner el regulador LM2940, todo lo que necesita es un soldador pequeño, alicates, destornillador y un poco de estaño. Reemplazar el LM7805CV con un LM2940 solo toma unos minutos y es tiempo bien empleado. Así es cómo se hace.

Paso 1: Quite el Regulador de Voltaje 7805

Usando un alicate pequeño, corte las patas del regulador 7805 como se muestra en la Figura C.1. Después de hacer esto las patas del regulador estarán aun soldadas a la plaqueta. Usted no tendrá que desoldar nada.

Figura C.1: Quite el regulador 7805 cortando los terminales al lado del encapsulado y desatornillando la tuerca pequeña.



Paso 2: Instale el regulador LM2940

Coloque el regulador LM2940 en el lugar del anterior, corte el extremo sobrante de las patas del regulador y colóquelo en su lugar ajustándolo con el tornillo y tuerca. Aliñe las patas del regulador para que se ajusten por encima de las patas del regulador anterior 7805. La Figura C.2 muestra cómo se verá la plaqueta en este punto.

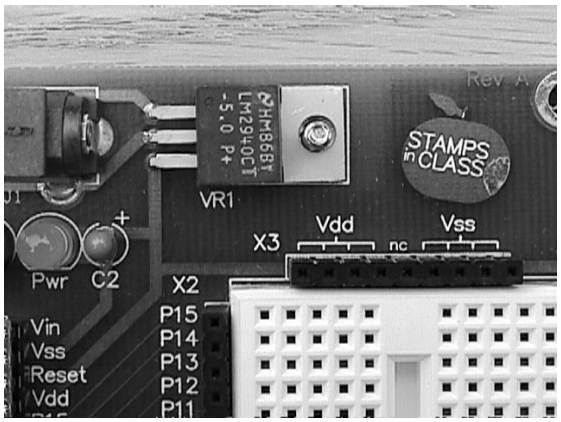


Figura C.2: Ponga el LM2940 en el lugar del 7805. Las patas se superpondrán a las del 7805.

Paso 3: Suelde el Regulador a la Plaqueta de Educación

Caliente el soldador y prepárese a soldar las patas del LM 2940 a las patas del regulador anterior. Antes de soldar, ajuste las patas del LM 2940-5 para que no sobrepasen las existentes. Suelde las patas y listo. Asegúrese de haber apretado la tuerca y el tornillo que fijan el regulador a la Plaqueta de Educación.

Apéndice F: Reglas de Armado en Protoboard

La Figura F.1 muestra los símbolos de circuito usados en los experimentos junto con la forma de encontrarlos en la Plaqueta de Educación Rev A. El símbolo para Vdd es la fuente de alimentación de 5-volt para el BASIC Stamp y la Plaqueta de Educación. Hay cuatro puntos de conexión en el extremo superior izquierdo para hacer conexiones a Vdd.

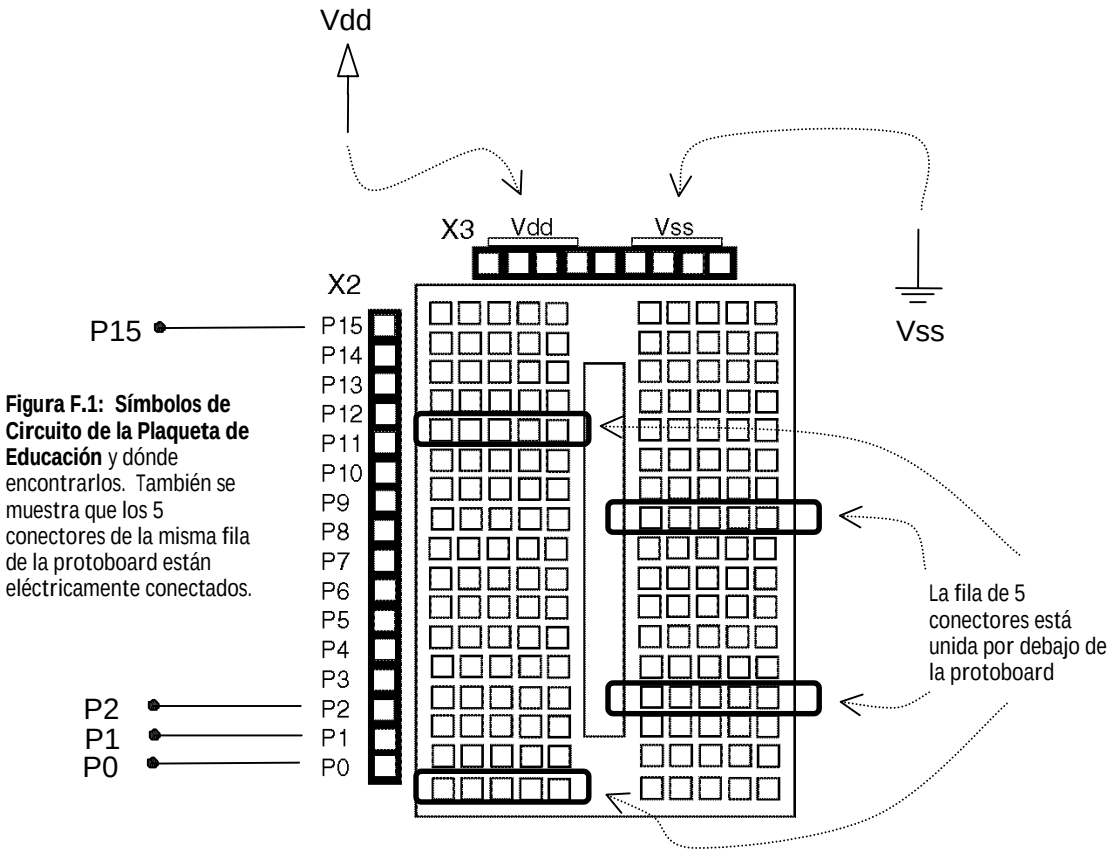


Figura F.1: Símbolos de Circuito de la Plaqueta de Educación y dónde encontrarlos. También se muestra que los 5 conectores de la misma fila de la protoboard están eléctricamente conectados.

Luego podemos observar el símbolo de masa, Vss. Este es el terminal de referencia para efectuar mediciones, considerando que tiene 0 volts en comparación con todas las tensiones de la Plaqueta de Educación. Los cuatro conectores de Vss están en el extremo superior derecho de la protoboard.

Hay una tira vertical de 15 conectores al costado izquierdo de la protoboard, para conectar los pines de E/S del BASIC Stamp. Cada pin de E/S tiene un rótulo. El pin de E/S P0 está unido al conector inferior. El pin P1 está conectado al inmediatamente superior, hasta llegar al pin P15 que está en el extremo superior izquierdo.

La Figura F.1 también muestra que las filas de 5 conectores se encuentran unidas eléctricamente, por debajo de la protoboard. Hay 34 de estas filas distribuidas en dos columnas. Si quiere conectar eléctricamente dos cables de interconexión, puede enchufarlos en la misma fila de 5 conectores. Del mismo modo, si quiere conectar uno o más cables al terminal de un componente, simplemente coloque a todos en la misma fila y ya estarán conectados. Se pueden agregar cuatro conectores más a Vdd, Vin, o Vss, simplemente colocando un puente entre por ejemplo, Vdd y una fila vacía de la protoboard.

La Figura F.2 muestra la protoboard de la BOE Rev B. Junto con Vdd y Vss, se agregaron tres conectores para Vin, la tensión de fuente de alimentación de la BOE sin regular. Cualquiera sea la tensión aplicada a la BOE, aparecerá en los conectores de Vin. Si la alimentación se obtiene de cuatro pilas de 1.5 V AA en el porta pilas del Boe-Bot, Vin tendrá 6 V. Si la fuente de alimentación es una batería de 9 V conectada a los terminales de la BOE, Vin tendrá 9 V. Si se conecta una fuente de alimentación de 7.5 V a la ficha de alimentación de la BOE, Vin debería tener 7.5 V. Sin embargo, tenga cuidado con las fuentes de alimentación. Su tensión de salida varía según la corriente que utilizamos.

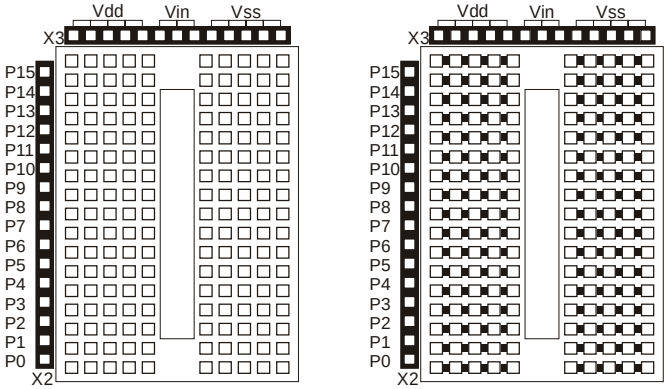
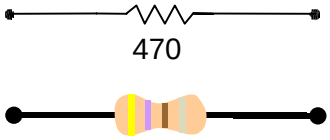


Figura F.2: Otra vista de las conexiones de la protoboard, esta vez en la BOE Rev B.

Apéndice G: Código de Color de Resistores

La Figura G.2 muestra el dibujo de un resistor debajo de su símbolo eléctrico. Este símbolo normalmente tiene el valor de la **resistencia** escrito por encima o debajo de él. Las líneas de color indican su valor, que se mide en **Ohms**. El símbolo omega(Ω) representa la unidad Ohm.

Figura G.2: Símbolo de circuito de un resistor y dibujo del componente



La mayoría de los tipos comunes de resistores tienen bandas de colores que indican su valor. Los resistores que usaremos en esta serie de experimentos son normalmente "1/4 watt, de carbón, con una tolerancia del 5%". Si se fija en la secuencia de bandas, observará que una de las bandas (en un extremo) es dorada. Ésta es la cuarta banda y el color dorado significa que tiene una tolerancia del 5%.

El código de colores del resistor es un estándar industrial para la identificación de valores de resistores. Cada banda de color representa un número y el orden en que se encuentran tiene un significado. Las dos primeras bandas indican un número. La tercera banda de color indica el multiplicador, o en otras palabras, la cantidad de ceros. La cuarta banda indica la tolerancia del resistor +/- 5, 10 o 20 %.

Color	1er Dígito	2do Dígito	Multiplicador	Tolerancia
negro	0	0	1	
marrón	1	1	10	
rojo	2	2	100	
naranja	3	3	1,000	
amarillo	4	4	10,000	
verde	5	5	100,000	
azul	6	6	1,000,000	
violeta	7	7	10,000,000	
gris	8	8	100,000.000	
blanco	9	9	1,000,000,000	
dorado			1/10	5%
plateado			1/100	10%
Sin color				20%

Apéndice G: Código de Color de Resistores

Un resistor tiene las siguientes bandas de color:

Banda 1. = rojo
Banda 2. = violeta
Banda 3. = amarillo
Banda 4. = dorado

Mirando la lista de arriba vemos que el rojo vale 2.

Así que escribimos: “2”.
Violeta tiene un valor de 7.
Así que escribimos: “27”
Amarillo tiene un valor de 4.
Así que escribimos: “27 y cuatro ceros” o “270.000”.

Este resistor tiene un valor de 270,000 ohms (o 270k) y una tolerancia del 5%.

Apéndice H: Calibración de la Medición de Distancia IR

Encontrando los Mejores Valores de Barrido de Frecuencia

Para ajustar la medición de distancia del Boe-Bot tendremos que determinar qué frecuencia es más conveniente para cada zona en cada par IR.

Nota	Este apéndice utiliza un método para determinar las mejores frecuencias para medir una distancia dada, usando hojas de cálculo. Esta actividad necesita tiempo y paciencia y solamente es recomendada en el caso que su medición de distancia esté muy descalibrada. El proceso involucra la toma de datos del barrido de frecuencia para determinar los valores más convenientes para medir ciertas distancias.

- Apunte los LEDs IR y los detectores en línea recta hacia delante.
- Coloque el Boe-Bot apuntando hacia una hoja blanca pegada a la pared, como blanco del IR.
- Coloque el Boe-Bot de forma que los LEDs IR estén a 1 cm del papel. Los dos pares IR deberían estar apuntados directamente hacia el papel.

Programa de Calibración de IR

El Programa H.2 realiza una barrido de frecuencia sobre el detector IR y muestra los resultados. Aunque las técnicas usadas son similares a las de otros programas, este programa tiene una característica única. El Basic Stamp está programado para esperar que usted presione la tecla Enter.

- Ingrese y ejecute el Programa H.2, pero no desconecte el cable serial del Boe-Bot.

```
' ¡Robótica! v1.5, Programa H.2: Barrido de Frecuencia IR

'----- Declaración --------------

  valores_izq       var    bit          ' Almacena valores izquierda.
  valores_der       var    bit          ' Almacena valores derecha.
  IR_freq           var    word         ' Almacena frecuencia de la tabla lookup.

'----- Inicialización ------------

  output 2                              ' Configura todas las E/S que enviarán
  output 7                              ' freqout como salidas.
  output 1
  freqout 2, 2000, 3000                 ' Indicador de reset.
```

```
  low 12                                ' P12 y 13 salidas, en estado bajo.
  low 13

'----- Rutina Principal --------------

principal:

  ' Muestra mensaje y espera recibir un Enter.

  debug cr,"Presione Enter cuando esté listo.  ", cr
  serin 16,16468,[wait(cr)]

  ' Después de que presione Enter, muestra el encabezado de tabla.

  debug "IR         Sensor  Sensor ", cr
  debug "Frec       Izq.    Derecho  ", cr
  debug "-------------------------", cr

  ' Toma múltiples mediciones de frecuencia.

  for ir_freq = 30500 to 46500 step 1000

    control_sensores:                   ' Controla los sensores

    valores_izq = 0
    valores_der = 0

    control_sensor_izq:
      freqout 7,1,IR_freq
      valores_izq = ~in8
       pause 10

    control_sensor_der:
      freqout 1,1,IR_freq
      valores_der = ~in0
      pause 10

   ' Muestra cada medición de frecuencia antes de pasar al siguiente bucle.

    debug dec5 IR_freq, "       ",bin1 valores_izq, "       ", bin1 valores_der, cr

  next

goto principal                          ' Salta a principal y repite el proceso.
```

- Haga clic en el recuadro superior que se muestra en la Figura H.3.

- Presione la tecla Enter. Los datos de respuesta en frecuencia aparecerán como se muestra en la figura.

El BASIC Stamp ha sido programado para hacer que la Debug Terminal muestre un "1" si se detecta un objeto y un "0" si no es así. La Figura H.3 muestra que la región del sensor izquierdo de buena respuesta a la señal, es entre 36500 y 40500. Para el sensor derecho, la respuesta en frecuencia es más sólida entre 37500 y 41500.

- Modifique el bucle **`for...next`** del Programa H.2 para que de pasos en incrementos de 250 e incluya los límites superior e inferior de ambos detectores. En el ejemplo mostrado en la Figura H.3, los valores de ***`inicio`***, ***`final`*** y ***`paso`*** del bucle **`for...next`** podrían ser modificados de la siguiente forma:

```
for ir_freq = 36500 to 41500 step 250
```

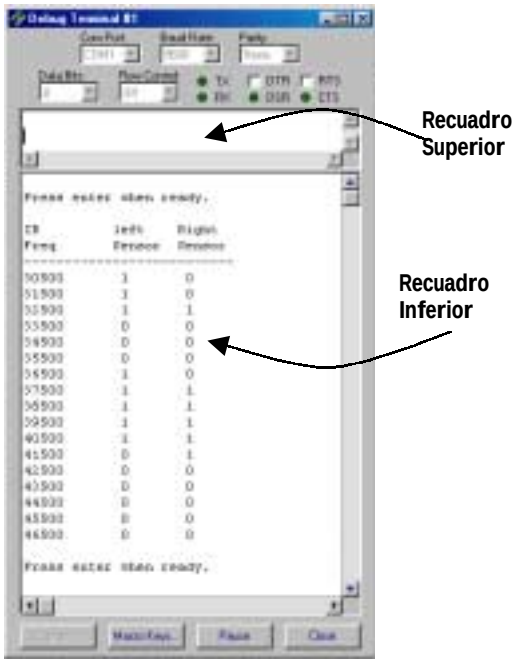


Figura H.3 Debug de datos de frecuencia.

- Vuelva a ejecutar su Programa H.2 modificado y presione enter nuevamente.

- Almacene los datos de los lados izquierdo y derecho en diferentes hojas de cálculo.

- Presione la tecla Enter nuevamente y registre el siguiente juego de datos.

- Repita este proceso tres veces más. Cuando termine, tendrá cinco juegos de datos para cada sensor en diferentes hojas de datos para cada frecuencia.

- Aleje el Boe-Bot a 2,5 cm Ahora los detectores IR de su Boe-Bot estarán a 5 cm del papel.

- Registre cinco juegos de datos a esta distancia.

- Siga alejando al Boe-Bot de a 2,5 cm por vez y almacenando los cinco juegos de datos del barrido de frecuencia para cada ajuste de distancia.

- Cuando el Boe-Bot se ha alejado 20 cm, el barrido de frecuencia mostrará principalmente ceros. Cuando el barrido de frecuencia entrega todos ceros, significa que no se detectó ningún objeto a ninguna de las frecuencias del barrido.

Revisando cuidadosamente las hojas de cálculo y realizando un proceso de eliminación, puede determinar la frecuencia óptima para cada par IR en cada zona. Se pueden detectar hasta 8 zonas sin ninguna reestructuración de las rutinas de navegación del Boe-Bot. Si quisiera observar 15 zonas, se usarían 30 comandos freqout de un milisegundo. Esta pausa sería demasiado larga para ubicarse entre pulsos de los servos. Una solución podría ser tomar 15 mediciones por pulso.

Veamos como determinar las mejores frecuencias para el sensor izquierdo. **Tenga en cuenta que tendrá que repetir este proceso para el sensor derecho.** Este ejemplo asume que esta buscando seis zonas (0 a 5).

- Comience examinando los puntos de datos tomados cuando el Boe-Bot estaba más alejado del papel. Tal vez no haya ningún juego de valores de la misma frecuencia que sean todos unos. Controle los datos de los siguientes 2,5 cm hacia el papel. Probablemente, verá un juego de cuatro o cinco unos a una frecuencia. Anote esta frecuencia como probable línea divisoria entre las Zonas 0 y 1.

- A cada una de las restantes cinco distancias, encuentre la frecuencia a la que los valores de salida comienzan a ser estables.

Por ejemplo, a 15 cm., tres frecuencias diferentes podrían mostrar cinco unos. Si revisa nuevamente los datos para 17,5 cm, dos de esas frecuencias eran estables, pero no la restante. Marque la frecuencia que no era estable a 17,5 cm pero sí a 15 cm, como la conveniente. Ahora, este ejemplo ha determinado las frecuencias que pueden ser usadas para separar las Zonas 5 y 4 y las Zonas 4 y 3. Repita este procedimiento para las restantes Zonas.

Cómo Funciona el Programa Barrido de Frecuencia

Dos variables tipo bit, **`valores_izq`** y **`valores_der`**, son declaradas para almacenar temporariamente cada salida del detector IR. Una variable llamada **`IR_freq`** se declara como índice del bucle **`for…next`**.

```
valores_izq     var         bit
valores_der     var         bit
IR_freq         var         word
```

¿Qué es una…

Comunicación Serial

La comunicación serial es uno de los métodos más comunes para que los dispositivos electrónicos intercambien datos. Hay dos tipos de comunicación serial, sincrónica y asincrónica.

La forma más común de comunicación serial sincrónica usa una línea para la transmisión de datos y otra para la señal de reloj. Cuando el dispositivo que envía los datos coloca un nuevo bit de información en la línea de datos, envía un pulso de reloj por la otra línea, indicando que el dato debe ser leído.

La comunicación serial asincrónica puede realizarse con una sola línea para la transmisión de datos. El dispositivo que envía la información y el que la recibe, saben a qué velocidad se transmitirán los datos. Conocida como baud rate, esta velocidad se mide en bits por segundo o bps.

Los mensajes seriales asincrónicos también tienen un bit de inicio, un número fijo de bits de datos, un bit de parada y algunas veces bit de paridad. El bit de parada es un valor binario que marca la separación entre los paquetes de datos. Se indica el bit de inicio cuando el valor binario de la línea de datos cambia de estado. Dado que los dos dispositivos conocen la velocidad de la comunicación, el transmisor sabe durante cuanto tiempo debe enviar el bit de inicio antes de enviar el primer bit de datos. De la misma forma, el receptor sabe cuanto esperar luego de recibir el bit de inicio antes de comenzar a registrar los datos.
El BASIC Stamp Manual explica este tema con mucho detalle. Para obtener más información sobre comunicación serial sincrónica, consulte los comandos **`shiftin`** y **`shiftout`**. Sobre la asincrónica consulte los comandos **`serin`** y **`serout`**.

El primer comando debug de la rutina principal no es nada nuevo. Muestra un mensaje indicando al usuario que presione la tecla enter cuando esté listo (para ejecutar un barrido de frecuencia). Las tres instrucciones debug que siguen, también resultarán conocidas. Sin embargo, el comando, **`serin 16, 16468, [wait(cr)]`** es nuevo. **`Serin`** es un comando multipropósito diseñado para recibir mensajes seriales asincrónicos enviados por otros dispositivos electrónicos.

```
debug cr,"Presione enter si está listo", cr
serin 16,16468,[wait(cr)]

debug "IR         Sensor  Sensor ", cr
debug "Freq       izq.    derecho", cr
debug "--------------------------", cr
```

El BASIC Stamp usa este comando **`serin`** para escuchar un mensaje serial enviado por la PC al pin de E/S P16, que es la línea Sin (Serial input) conectada a la línea de transmisión de datos de la computadora (DTR) mediante el cable serial. El argumento ***`baudmode`*** es 16468, que se elige de una tabla en la sección del comando **`Serin`** del BASIC Stamp Manual. El valor 16468 le dice al BASIC Stamp que se comunique con la Debug Terminal. Las especificaciones para la Debug Terminal son 9600 bits por segundo (bps), 8-bits, sin paridad, un bit de parada. Estos argumentos se explican en detalle en el BASIC Stamp Manual.

Normalmente **`serin`** tiene una variable entre los corchetes. De esta forma, los datos enviados desde la PC al BASIC Stamp pueden ser guardados y usados posteriormente. En este caso, el argumento **`wait(cr)`** no almacena ningún dato. Todo lo que hace es decirle al comando **`serin`** que espere una indicación de salto de línea (enter). El comando **`serin`** no hace nada más que controlar la entrada hasta recibir un "enter". Cuando el comando **`serin`** recibe el salto de línea, el programa se mueve al siguiente comando. En otras palabras, **`serin`** le permite al programa continuar ejecutándose, solamente después de recibir el caracter de salto de línea. El resultado de este comando es que puede tomarse su tiempo acomodando al Boe-Bot y cuando esté listo puede presionar la tecla "enter" para tomar los datos nuevos.

Antes de ser modificado, el bucle **`for...next`** barría los valores de **`IR_freq`** desde 30500 hasta 46500 en pasos de 1000. El comando **`freqout`** anidado dentro del bucle **`for...next`** usa el valor de **`IR_freq`** como su argumento de ***`frecuencia`***.

Como en los ejemplos de programas anteriores, el valor de la salida de los detectores IR es invertido usando el operador NOT (~), luego se almacena en una variable de un bit, como **`valores_izq`** para el par IR izquierdo.

```
for ir_freq = 30500 to 46500 step 1000

    control_sensor_izq:
      freqout 7,1,IR_freq
      valores_izq = ~in8
      pause 10


    control_sensor_der:
      freqout 1,1,IR_freq
      valores_der = ~in0
      pause 10
```

Luego, los valores de salida de **`debug`** se imprimen en formato de tabla en la Debug Terminal. Para que se vea mejor, se insertaron espacios en blanco entre los valores usando el mensaje **`"  "`**. El ancho de cada número también está controlado por modificadores como **`bin5`** y **`dec1`**.

```
debug dec5 IR_freq, "     ",bin1 valores_izq, "     ",bin1 valores_der, cr

  next
```

Su Turno

- Si tuvo éxito en sus calibraciones para las cinco mediciones y el tiempo lo permite, pruebe de incrementar la resolución, aumentando a ocho mediciones. Almacene los datos para ambos métodos.

Apéndice I: Reglas de Competición del Boe-Bot

Si usted está planeando una competencia de robots autónomos, estas reglas son provistas por cortesía de Seattle Robotics Society (SRS).
http://www.seattlerobotics.org/

Desafío 1: Ejercicios de Piso del Robot

Propósito
La competencia de ejercicios de piso intenta darle a los inventores de robots una oportunidad de mostrar sus robots u otros artefactos técnicos.

Reglas
Las reglas para esta competencia son bastante simples. Se marca un área plana de 3 metros por 3 metros, preferiblemente con algún borde sólido. Cada contendiente tendrá un máximo de 5 minutos en esta área para mostrar lo que puede hacer. El controlador del robot puede comentar las capacidades y características del robot. Como siempre, cualquier robot que pueda dañar el área o suponga un riesgo para el público, no será aceptado. Los Robots no necesariamente deberán ser autónomos, pero es recomendado. El ganador será determinado por el público, tanto por los aplausos (el nivel de sonido determinará el ganador), o mediante algún otro mecanismo de votación.

Desafío 2: Reglas de Seguimiento de Línea

Objetivo
Construir un robot autónomo que comience en el Área "A" (en la posición "S"), viaje al Área "B" (completamente sobre la línea), luego viaje al Área "C" (completamente sobre la línea), luego regrese al Área "A" (en la posición "F"). El robot que realice esta tarea en la menor cantidad de tiempo, gana. El robot debe entrar a las áreas "B" y "C" para calificar. La disposición exacta de la pista no será conocida hasta el día del desafío, pero tendrá las tres áreas previamente descriptas.

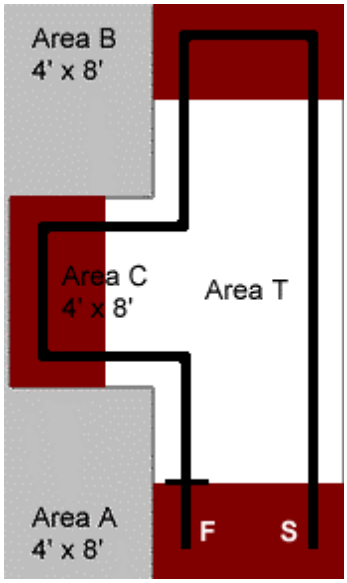
Habilidades evaluadas

La habilidad de reconocer una ayuda de navegación (la línea) y usarla para lograr el objetivo.

Tiempo Máximo para Completar la Prueba

Cuatro minutos.

Ejemplo de Pista



Todas las medidas del ejemplo son aproximadas. Hay una línea sólida que separa el Área "A" del Área "T" en la posición "F". Esto indica dónde termina la pista. La línea es negra, de aproximadamente 2 centímetros de espesor y separada aproximadamente medio metro de las paredes. Todas las curvas tienen un radio mínimo de 30 centímetros y un radio máximo de 1 metro. Las paredes tienen una altura de 10 centímetros y rodean toda la pista. El piso es blanco y esta hecho de papel o Tyvec. Tyvec es un plástico fuerte usado en sobres de correo y en la construcción de casas.

Las posiciones "S" y "F" son simples ejemplos y no son ubicaciones específicas. Un competidor puede poner el robot en cualquier parte del Área "A", apuntando en cualquier dirección cuando arranca. El robot debe estar completamente dentro del Área "A". Áreas "A", "B" y "C" no están (en color diferente) en esta pista.

Puntaje

El puntaje de cada contendiente es calculado tomando el tiempo necesitado para completar la pista (en segundos), menos 10% por cada "tarea cumplida". El contendiente con menor puntaje gana.

Tarea cumplida	Reducción
Se detiene en A antes de alcanzar B y C	10%
No toca ninguna pared	10%
Inicia con una orden	10%

("Inicia con una orden" significa que el robot comienza con una orden externa, sin ser tocado. Podría ser, por ejemplo, un sonido o una luz.)

Desafío 3: Laberinto

Propósito

El gran laberinto intenta presentar una prueba de astucia navegacional del robot autónomo. El puntaje está a favor de los robots que son brutalmente rápidos o de aquellos que tienen la capacidad de memorizar el laberinto. El objetivo de la prueba es poner un robot en la entrada del laberinto y sin ayuda externa, encuentre la salida en la menor cantidad de tiempo.

Características físicas

El laberinto se construye en madera terciada de 18 milímetros. Las paredes tienen una altura aproximada de medio metro y son pintadas en colores primarios con pintura brillante. Las paredes son fijadas con un espaciamiento aproximado de medio metro. Debido al espesor del terciado y limitaciones en la precisión, los caminos pueden tener un tamaño un poco menor. El laberinto puede tener hasta 2 metros cuadrados, pero puede ser más pequeño en función del espacio disponible para el evento.

El laberinto se construirá sobre una alfombra industrial o piso rígido (dependiendo del lugar donde se desarrolla el evento). El laberinto se construirá bajo cubierta, así que su robot no tiene por que ser resistente a la lluvia; sin embargo, puede ser expuesto a temperaturas variadas, viento y condiciones de luz variable. El laberinto es de tipo clásico de dos dimensiones: hay un solo camino de la largada a la llegada y no hay islas en el laberinto. La salida y la llegada se encuentran en paredes externas. Algunos laberintos se construyen de forma que si usted sigue la pared izquierda o la pared derecha llega a la salida. En el caso de este laberinto se tendrá especial cuidado de evitar esto, de forma que no pueda obtener ventajas siguiendo una u otra pared.

Apéndice I: Reglas de Competición del Boe-Bot

Limitaciones de los Robots

El límite principal de los robots es que deben ser autónomos: una vez encendido por el propietario, no se permite ninguna interacción hasta que el robot aparezca en la salida, o desgraciadamente quede trabado contra alguna pared. Obviamente el robot debe ser suficientemente pequeño para pasar entre las paredes del laberinto. Puede tocar las paredes, pero no puede moverlas en su beneficio (no se permiten topadoras ni tanques de guerra). Los jueces pueden descalificar a cualquier robot que mueva las paredes excesivamente. El robot no debe dañar las paredes del laberinto, ni el piso. Cualquier forma de energía es permitida, siempre y cuando las leyes locales no requieran protección auditiva en su presencia, o coloquen algún otro tipo de limitación.

Puntaje

Cada robot será puesto en el laberinto tres veces. El robot con el menor tiempo en alguna de las pasadas es el ganador. El tiempo máximo permitido por pasada es de 10 minutos. Si un robot no puede encontrar la salida en ese tiempo, es detenido figurando en la planilla un tiempo de 10 minutos y la distancia a la salida. Si ningún robot encuentra la salida, aquél que llegue más cerca de la salida será el ganador, siendo determinado por los jueces del evento.

Logística

Cada robot hará una pasada, hasta que todos hallan probado el laberinto. Cada robot entonces realiza una segunda pasada y luego realizan la tercera. Queda a criterio de los jueces permitirle a un contendiente demorar su pasada por inconvenientes técnicos. Un robot puede recordar lo que encontró en una pasada previa, para tratar de mejorar su tiempo (mapear el laberinto en la primera pasada) y puede usar esta información en las pasadas subsecuentes, siempre y cuando el robot lo haga por sí mismo. No está permitido "configurar" manualmente el robot mediante hardware o software con la distribución del laberinto.