Robótica con el Boe-Bot

Guía del Estudiante

VERSION 3.0

PARALLAX INC

GARANTÍA

Parallax garantiza sus productos contra defectos en sus materiales o debidos a la fabricación por un período de 90 días a partir de la recepción de los mismos. Si usted descubre un defecto, Parallax según corresponda, reparará, reemplazará o regresará el valor de la compra. Antes de regresar el producto a Parallax, simplemente pida un número de autorización de regreso de mercancía (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. Parallax le regresará 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 DEL 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 gastos de envío y manejo. Esto no se aplica si el producto ha sido alterado o dañado. Consulte la sección de Garantía arriba acerca de las instrucciones para regresar un producto a Parallax.

DERECHOS DE COPIA Y MARCAS REGISTRADAS

ISBN 9781928982531 (VERSIÓN ORIGINAL EN INGLÉS)

3.0.0-10.11.10-SCP – TRX 1.0-11.07.19

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 beneficios, tiempos muertos, buena fe, daño o reemplazo de equipo o propiedad y cualesquiera costos de recuperación, reprogramación o de reproducción de datos guardados o usados 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, sin importar la naturaleza del riesgo de la misma.

ERRATA

Si bien se realiza un gran esfuerzo para asegurar la precisión de nuestros textos, aún puede haber errores. Ocasionalmente se publica en nuestro sitio www.parallax.com una hoja de fe de errata con una lista de los errores y correcciones conocidos para un texto determinado. Si encuentra un error, por favor envíe un email a editor@parallax.com.

Tabla de Contenido

Prefacio

Los Robots son usados en las industrias automotríz, médica y de manufactura, en todas las formas de vehículos de exploración, y, claro, en muchos filmes de ciencia ficción. La palabra "robot" apareció primero en una obra satírica checoslovaca, los Robots Universales de Rossum, de Karel Capek en 1920. Los robots en esta obra tendían a tener semejanza humana. A partir de este punto, pareció que muchas historias de ciencia ficción involucraron a estos robots tratando de ajustarlo en la sociedad y que tuvieran sentido más allá de las emociones humanas. Esto cambió cuando General Motors instaló los primeros robots en su planta de manufactura en 1961. Estas máquinas automatizadas presentaron una imagen complétamente diferente de los robots “semihumanos” de la ciencia ficción.

Construir y programar un robot es una combinación de mecánica, electrónica y solución de problemas. Lo que está a punto de aprender mientras que realiza las actividades y proyectos en este texto será relevante a las aplicaciones del mundo real que usan control robótico, siendo las únicas diferencias el tamaño y la sofisticación. Los principios mecánicos, los listados de programas ejemplo y los circuitos que usará son muy similares a, y a veces los mismos que, las aplicaciones industriales desarrollados por ingenieros.

La meta de este texto es que los estudiantes se interesen en y se emocionen acerca los campos de la ingeniería, mecatrónica y desarrollo de software conforme diseñan, construyen y programan un robot autónomo. Esta serie de actividades “manos a la obra” y proyectos introducirán a los estudiantes a los conceptos básicos de la robótica usando el robot Boe-Bot® Parallax, llamado "Boe-Bot." Su nombre viene de la tarjeta portadora Board of Education® que se monta en su its chasis sobre ruedas. Un ejemplo de un Boe-Bot con un circuito infrarrojo de detección de obstáculos construído en el área de prototipo libre de soldadura del Board of Education se muestra en la Figura P-1.

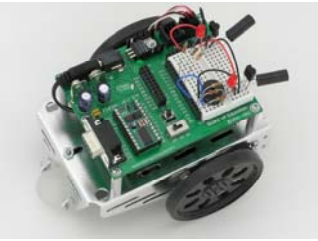


Figura P-1
Robot Boe-Bot® de Parallax Inc.

Las actividades y proyectos en este texto comienzan con una introducción al cerebro de su Boe-Bot, the microcontrolador Parallax BASIC Stamp® 2, y luego prosiguen a la construcción, prueba y calibración del Boe-Bot. Después programará el Boe-Bot para hacer maniobras básicas y luego procederá a agregar sensores y escribir programas que lo hagan reaccionar a sus alrededores y ejecutar tareas autónomas.

ACERCA DE LA VERSION 3.0

Esta es la primera revisión de este título desde 2004. Los cambios mayores incluyen:

- Reemplazo del fotorresistor de sulfato de cadmio con un sensor de luz que cumple conRoHS de un tipo que será mas común en el diseño de productos en el futuro. Esto requirió la reescritura del Capítulo 6.
- Mover la porción "Parametrización y Prueba" del Capítulo 1 y los apéndices de Hardware y Solución de Problemas al archivo de Ayuda. Esto fue hecho para apoyar las conexiones de hardware tanto serial como USB y otras conexiones para programación conforme continúe la expansión de nuestros productos y tecnologías. Esto también permite el mantenimiento dinámico del material de Hardware y Solución de Problemas.
- Remoción de las referencias al CD de Parallax, que ha sido removido de nuestros kits, reduciendo el desperdicio y asegurando que nuestros usuarios bajen la más reciente versión del software BASIC Stamp Editor y drivers USB disponibles para sus sistemas operativos (www.parallax.com/go/Boe-Bot).

Adicionalmente han sido corregidos algunos elementos erróneos identificados en la versión anterior (2.2). El material aún apunta hacia las mismas metas todos los mismos conceptos de programación y comandos son cubiertos, junto con algunos nuevos. Finalmente, la paginación ha sido cambiada para que coincida la numeración PDF y la física, para facilidad de uso.

AUDIENCIA

Este texto está diseñado para ser un punto de entrada a la literatura tecnológica y una curva de aprendizaje fácil tanto de la programación integrada como de la introducción a la robótica. El texto está organizado para que pueda ser usado por la variedad más amplia posible de estudiantes y de estudiantes independientes. Los estudiantes a mitad de sus cursos pueden intentar los ejemplos en este texto a modo de tour guiado simplemente siguiendo las instrucciones marcadas bajo la supervisión de su instructor. Al otro extremo de este espectro, las habilidades de comprensión y solución de problemas de los estudiantes de pre-ingeniería pueden ser puestas a prueba con las preguntas, ejercicios y

proyectos (con soluciones) en cada Resumen de Capítulo. El estudiante independiente puede trabajar a su propio paso y obtener ayuda a través del foro Stamps in Class citado a continuación.

FOROS DE APOYO

Parallax mantiene foros moderados gratuidos para nuestros clients que cubren una variedad de temas:

- Propeller Chip: para todas las discusiones relacionadas con el microcontrolador de multinúcleos Propeller y el desarrollo de herramientas de esta línea de producto.
- BASIC Stamp: ideas de proyectos, apoyo y temas relacionados para todos los modelos del BASIC Stamp Parallax.
- Sensores: Discusiones en relación a la amplia variedad de sensors de Parallax y su interfase con los microcontroladores Parallax.
- Stamps in Class: Estudiantes, maestros y clientes discuten aquí materiales educativos de Parallax y proyectos escolares.
- Robótica: Para todos los robots Parallax y robots personalizados construídos con procesadores y sensores Parallax.
- Wireless: Temas incluyendo XBee, GSM/GPRS, telemetría y comunicación de datos sobre la banda de radio aficionado.
- PropScope: Discusión y apoyo técnico para este osciloscopio USB que contiene un circuito integrado Propeller.
- The Sandbox: Temas relacionados al uso de productos Parallax pero no especificados en los otros foros.
- Proyectos: Publique aquí sus proyectos en proceso y completos, hechos a partir de productos Parallax.

RECURSOS PARA EDUCADORES

Tenemos una variedad de recursos para este texto diseñados para apoyar a los educadores.

"Mini Proyectos" Stamps in Class

Para complementar nuestros textos, proveemos un banco de para el salón de clase. Diseñados para enganchar a los estudiantes, cada "Mini Proyecto" contiene código fuente completo, explicaciones de "Cómo Trabaja", esquemáticos y diagramas de cableado o fotos de un dispositivo que quizá a algún estudiante le guste usar. Muchos proyectos presentan un video introductorio para promover el auto-estudio en aquellos estudiantes más interesados en electrónica y programación. Solo siga la liga a los "Mini Proyectos" de Stamps in Class en www.parallax.com/Education.

Cursos para Educadores

Estos cursos activos e intensivos de 1 o 2 días para instructores son enseñados por ingenieros de Parallax o maestros experimentados que usan los materiales educativos de Parallax en sus salones de clase. Visite www.parallax.com/Education → Educators Courses para más detalles.

Foro de Educadores Parallax

En este foro privado y gratuito los educadores pueden hacer preguntas y compartir sus experiencias al usar los productos Parallax en sus salones de clase. Aquí también se publican materiales educativos complementarios. Para enrolarse envíe un email a education@parallax.com para instrucciones; se requerirán pruebas de su validación com educador.

Materiales Educativos Complementarios

Los textos educativos selectos de Parallax tienen un conjunto de preguntas y soluciones no publicadas en nuestro foro para Educadores Parallax; invitamos a los educadores a copiar y modificar este material a voluntad para una rápida preparación de tareas, cuestinarios y pruebas. También se pueden publicar aquí presentaciones de PowerPoint y materiales de exámenes preparados por otros educadores.

Permisos de Derecho de Autor para Uso Educativo

No se requiere licencia del sitio para descarga, duplicación e instalación de software Parallax para fines educativos con productos Parallax en tantas computadoras de escuela

u hogar como sea necesario. Tanto nuestros textos Stamps in Class como nuestro Manual BASIC Stamp están disponibles como descargas PDF gratuitas y pueden ser duplicados mientras que lo sean para fines educativos exclusivamente con products microcontroladores Parallax y que el estudiante no pague mas que el costo de duplicación. Los archivos PDF no están candadeados, habilitando la selección de texto e imagenes para preparar notas, transparencias, o presentations PowerPoint.

TRADUCCIONES

Los textos educativos Parallax han sido traducidos a otros idiomas; estos textos son descargas gratuitas y son sujetos a los mismos Permisos de Derecho de autor con fines educativos como lo son nuestras versiones originales. Para ver la lista completa, haga click en la liga de Tutoriales y Traducciones en www.parallax.com/Education. Estos fueron preparados en coordinación con el programa de Traductores Voluntarios de Parallax. Si esta interesado en participar en nuestro programa de Taductores Voluntarios, envíe un email a translations@parallax.com.

ACERCA DEL AUTOR

Andy Lindsay se unió a Parallax Inc. en 1999 y desde entonces has escrito once libros y numerosos artículos y documentos de productos para la compañía. Las últimas tres versiones de *Robótica con el Boe-Bot* fueron diseñadas y actualizadas en base a las observaciones y retroalimentaciones de educadores que Andy colectó al viajar por la nación dando cursos y eventos para educadores Parallax. Andy estudió Ingeniería Eléctrica y Electrónica en la Universidad del Estado de California, Sacramento, y es co – autor de varios artículos que tratan el tema de microcontroladores en curricula pre-ingenieril. Cuando no está escribiendo material educativo, Andy hace productos, sus aplicaciones e ingeniería de producto para Parallax.

CONTRIBUCIONES ESPECIALES

El equipo Parallax ensamblado para preparar esta edición incluye: el excelente liderazgo de departmento de Aristides Alvarez, diseño de lecciones y autoría técnica de Andy Lindsay; arte en la portada de Jen Jacobs; ilustraciones gráficas de Rich Allred y Andy Lindsay; nitpicking, edición y arreglo de Stephanie Lindsay. Un agradecimiento especial a Ken Gracey, fundador del programa Stamps in Class y a Tracy Allen y Phil Pilgrim por su consultoría en la selección del sensor de luz usado en esta versión para reemplazar el fotorresistor de sulfato de cadmio. Stephanie esta particularmente agradecida a John Kauffman por su revision de último minuto del Capítulo 6 revisado.

1

Capítulo 1: El Cerebro de su Boe-Bot

El robot Boe-Bot® de Parallax Inc es el objeto de las actividades, proyectos, y concursos en este libro. En la Figura 1-1 se muestran el Boe-Bot y un acercamiento de su cerebro microcontrolador programable BASIC Stamp® 2. El módule BASIC Stamp 2 es poderoso y fácil de usar, especialmente con un robot.

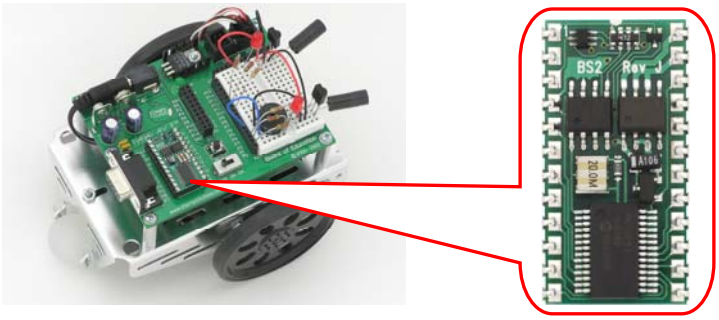


Figura 1-1
Modulo BASIC Stamp en un Robot Boe-Bot

Las actividades en este texto le guiarán a través de la escritura de programas simples que hacen que el BASIC Stamp y su Boe-Bot hagan 4 tareas robóticas esenciales:

1. Monitoree sensores para detectar el mundo a su alrededor
2. Tomar decisiones basado en lo que sensa
3. Controla su movimiento (operando los motores que hacen que giren sus ruedas)
4. Intercambia información con su Robotista (¡ese será usted!)

El lenguaje de programación que usará para completar estas tareas se llama **PBASIC**, que es un acrónimo que significa:

- Parallax–La compañía que inventó y manufactura los microcontroladores BASIC Stamp
- Beginners – Hecho para principiantes para aprender cómo programar computadoras
- All-purpose - Poderoso y útil para resolver muy diferentes tipos de problemas
- Symbolic - Usando símbolos (términos que semejan frases y palabras en Inglés)
- Instruction – Para indicarle a la computadora qué hacer
- Code - en términos que la computadora (y usted) puedan entender

¿Qué es un Microcontrolador? Es un dispositivo programable que está diseñado dentro de su reloj de pulsera digital, teléfono celular, calculadora, radio despertador, etc. En estos dispositivos, el microcontrolador ha sido programado para sensar cuando presiona un botón, haga ruidos electrónicos y controle la pantalla digital del dispositivo. También están construidos dentro de la maquinaria de una fábrica, carros, submarinos y naves espaciales porque ellos pueden ser programados para leer sensores, tomar decisiones y orquestar dispositivos que controlan partes en movimiento.

La guía del estudiante *¿Qué es un micro controlador?* Es el primer texto recomendado para principiantes. Está lleno de ejemplos de cómo usar los microcontroladores y como hacer del BASIC Stamp el cerebro de sus propias invenciones microcontroladas. Esta disponible gratis como descarga de www.paralax.com/go/WAM, y también está incluido en la función de ayuda del editor del BASIC Stamp como un archivo PDF. Está incluído en el Kit de Actividades del BASIC Stamp y el Kit de Descubrimiento del BASIC Stamp, que son distribuídos por muchas casas de electrónica. Estos kits también pueden ser comprados directamente en Parallax, ya sea en línea en www.parallax.com/go/WAM o por teléfono al (888) 512-1024.

HARDWARE Y SOFTWARE

Iniciar con los módulos del microcontrolador BASIC Stamp es similar similar a inicar con una PC o una laptop nueva. Lo primero que mucha gente tiene que hacer es sacarla de la caja, conectarla, instalarla y probar algún software, y quizá escribir algún software propio usando un lenguaje de programación. Si esta es su primera vez usando un modulo BASIC Stamp, estará haciendo estas mismas actividades. Si está un una clase, su hardware puede estar ya configurado para uested. Si este es el caso, su maestro puede tener otras instrucciones. Si no, este Capítulo le llevará a través de los pasos para poner a punto y funcionando su nuevo microcontrolador BASIC Stamp.

ACTIVIDAD #1: OBTENIENDO EL SOFTWARE

El Editor del BASIC Stamp (version 2.5 o posterior) es el software que usará en muchas de las actividades y proyectos en este texto. Usará este software para escribir programas que el modulo BASIC Stamp correrá. También puede usar este software para mostrar mensajes enviados por el BASIC Stamp que le ayudarán a entener lo que está sensando.

Requerimentos del Sistema de Cómputo

Necesitará una computadora personal para corer el software Editor de BASIC Stamp. Su computadora necesitará tener las siguientes características:

- Sistema operative Microsoft Windows 2K/XP/Vista/7 o más reciente

- Un Puerto serie o USB disponible
- Un programa para accesar y navegar por Internet

Bajando el Software desde Internet

Es importante siempre usar la más reciente versión del software del Editor BASIC Stamp si es posible. El primer paso es ir al sitio web de Parallax y bajar el software.

✓ Usando un navegador, vaya a www.parallax.com/basicstampsoftware.

Figura 1-2: Descarga del Editor BASIC Stamp en www.parallax.com/basicstampsoftware

Use el botón "Click Here to Download" para obtener la última versión del software.

- ✓ Haga click en Click Here to Download para bajar la última versión del software Editor BASIC Stamp Windows.

- ✓ Se abrirá una ventana de descarga del archive, preguntándole si quiere corer o guarder este archivo (Figura 1-3). Haga click en Save.

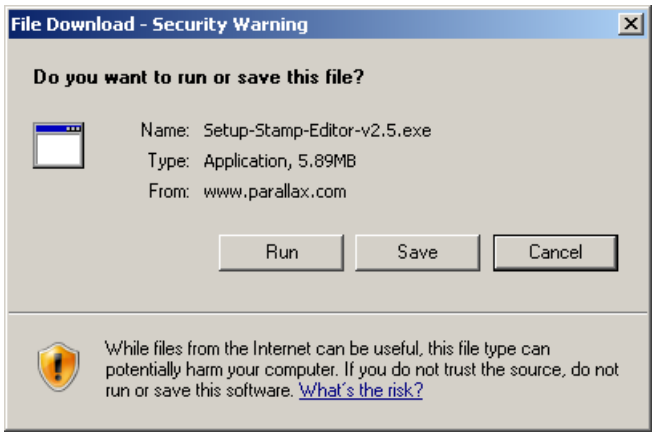


Figura 1-3
Ventana de descarga del archivo

Haga Click en Save, luego salve el archive en su computadora.

- ✓ Siga los avisos que aparezcan. Cuando la descarga esté complete, haga click en Run. Verá mensajes de su sistema operative pidiéndole que verifique que desea continuar con la instalación. Siempre confirme que desea continuar.

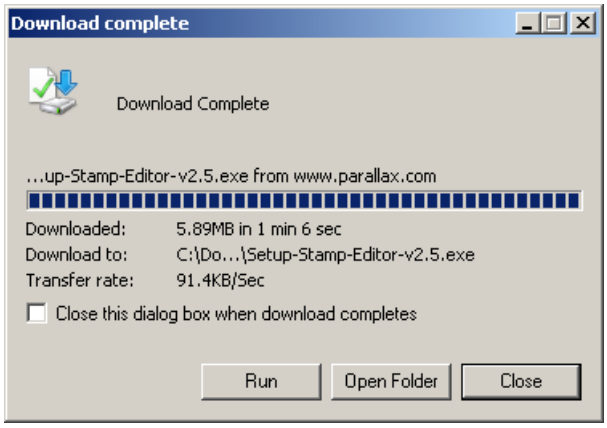


Figura 1-4
Mensaje de descarga completa

Haga Click en Run.

Si se le pide, siempre confirme que desea continuar.

✓ La ventana de instalación del Editor BASIC Stamp se abrirá (Figura 1-5). Haga click en Next y siga los avisos, aceptando los valores predeterminados.

Figura 1-5
Ventana de Instalación del Editor BASIC Stamp

Haga Click en Next.

✓ IMPORTANTE: Cuando aparezca el mensaje "Install USB Driver" (Figura 1-6), no retire la marca en la casilla Automatically install/update driver (recommended) y luego haga click en Next.

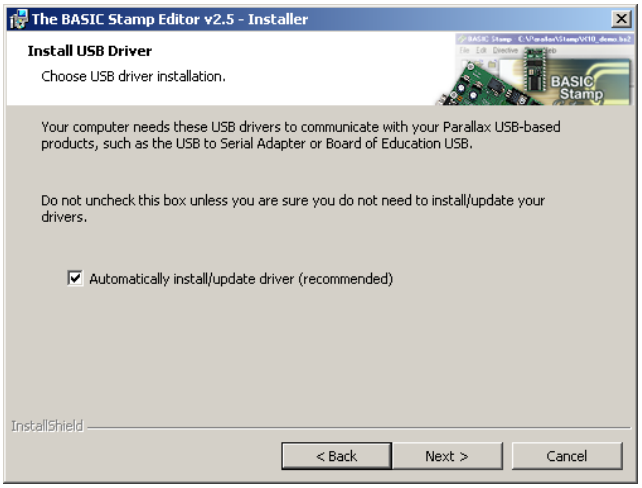


Figura 1-6
Mensaje de Instalación del Driver USB

Deje la marca puesta, y haga click en Next.

✓ Cuando aparezca el mensaje "Ready to Install the Program", haga click en el botón Install. Aparecerá una barra de progreso y esto puede tomar unos minutos.

En este punto, una ventana adicional aparecerá detrás de la ventana activa mientras que los drivers USB se actualizan. Esta ventana eventualmente se cerrará cuando la instalación del driver se complete. Si no ve esta ventana, esto no indica un problema.

Acerca de los drivers USB. Los drivers USB que se instalan por defecto con el instalador del Editor BASIC Stamp Windows son necesarios para usar cualquier hardware Parallax conectado al puerto USB de su computadra. Las siglas VCP quieren decir Virtual COM Port (Puerto COM virtual), y le permitirá al Puerto USB de su computadora ser visto y tratado como un puerto serial RS232 estandar serial port por el hardware Parallax.

Drivers USB para otros sistemas operativos. Los drivers USB VCP incluídos en el software Editor BASIC Stamp Windows son solo para ciertos sistemas operativos Windows. Para más información visite www.parallax.com/usbdrivers.

✓ Cuando la ventana le diga que la instalación se ha completado exitosamente haga click en Finish (Figura 1-7).

Figura 1-7
Instalación del Editor BASIC Stamp Completa

Haga Click en Finish.

ACTIVIDAD #2: USO DE HELP PARA PARAMETRIZAR EL HARDWARE

En esta sección correrá el archivo Help del Editor BASIC Stamp. Dentro de este aprenderá acerca las diversas tarjetas de programación del BASIC Stamp disponibles para el programa Stamps in Class y determine cuál está usando. Luego, siga los pasos en el archivo Help para conectar su hardware a su computadora y probar su sistema de programación BASIC Stamp.

Corriendo el Editor BASIC Stamp por Primera Vez

- ✓ Si ve el ícono del Editor BASIC Stamp en el escritorio de su computadora, haga doble-click sobre el (Figura 1-8).
- ✓ O haga click en el menu Start de su computadora, luego escoja All programs ▸ Parallax Inc ▸ BASIC Stamp Editor 2.5 ▸ BASIC Stamp Editor 2.5.

Figura 1-8
Ícono de escritorio del Editor BASIC Stamp

Haga Doble-click para iniciar el programa.

- ✓ En la barra de herramientas del Editor BASIC Stamp, haga click en Help en la barra (Figura 1-9) y luego seleccione BASIC Stamp Help… del menú desplegable.

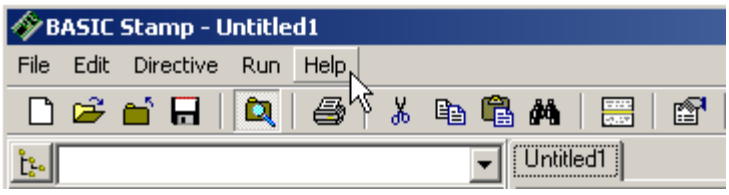
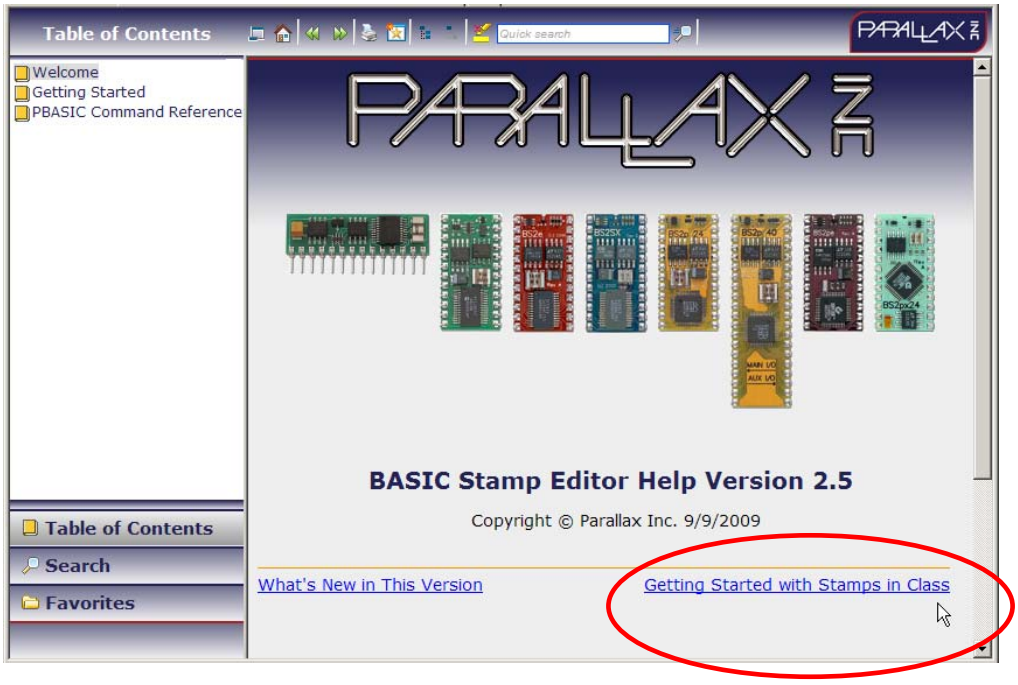


Figura 1-9
Abriendo el Menu Help

Haga Click en Help, luego escoja BASIC Stamp Help del menu desplegable.

Figura 1-10: Ayuda del Editor BASIC Stamp



- ✓ Haga click en la liga Getting Started con Stamps en Class al final de la página de bienvenida, como se muestra en la esquina inferior derecha de la Figura 1-10. El archivo de ayuda está en Inglés.
- ✓ O bien, siga las instrucciones en Español de "Empezando con Stamps in Class" archivo pdf.

Siguiendo las Direcciones en el Archivo de Ayuda

A partir de aquí, siga las direcciones en el archivo de Ayuda para completar estas tareas:

- Identifique cuál tarjeta de desarrollo BASIC Stamp está usando
- Conecte su tarjeta de desarrollo a su computadora
- Pruebe su conexión de programación
- Corrija su conexión de programación, si es necesario
- Escriba su primer programa PBASIC para su BASIC Stamp
- Apague su hardware cuando termine

Cuando haya completado las actividades en el archivo Help (o Empezando con Stamps in Class" archivo pdf), regrese a este libro y continue con el Resumen siguiente antes de continuar con el Capítulo 2.

¿Qué hago si me quedo atorado?

Si tiene problemas al seguir las direcciones en este libro o en el archivo Help, tiene muchas opciones para obtener apoyo técnico:

- **Foros:** inscríbase y envíe un mensaje en nuestro foro moderado y gratuito Stamps in Class en forums.parallax.com.
- **Email:** envíe un email a support@parallax.com.
- **Teléfono:** en los Estados Undos (area Continental) llame gratuitamente al 888-99-STAMP (888-997-8267). De cualquier otro lado llame al (916) 624-8333.
- **Más fuentes:** Visite www.parallax.com/support.

RESUMEN

Este Capítulo le guió a través de los siguientes temas:

- Una introducción al módulo BASIC Stamp
- Dónde conseguir el software del Editor BASIC Stamp gratuito que usará en casi todos los experimentos en este texto
- Cómo instalar el software del Editor BASIC Stamp
- Cómo usar la Ayuda del Editor BASIC Stamp y el Manual del BASIC Stamp
- Una introducción al modulo BASIC Stamp, Board of Education, y HomeWork Board
- Cómo parametrizar su hardware BASIC Stamp
- Cómo probar su software y su hardware
- Cómo escribir y correr un programa PBASIC

- Usar los comandos **DEBUG** y **END**, el control de caracter **CR**, y el formateador **DEC**.
- Una breve introducción al código ASCII
- Cómo desconectar la energía de su Board of Education o su HomeWork Board cuando finalice

Preguntas

1. ¿Qué dispositivo sera el cerebro de su Boe-Bot?
2. Cuando el BASIC Stamp envía un caracter a su PC/laptop, ¿qué tipo de números se usan para enviar el mensaje a través del cable de programación?
3. ¿Cuál es el nombre de la ventana que despliegua mensajes enviados desde el BASIC Stamp a su PC/laptop?
4. ¿Qué tipo de comandos PBASIC aprendió en este Capítulo?

Ejercicios

1. explique lo que hace el asterisco en este comando: **DEBUG DEC 7 * 11**

2. Prediga lo que la terminal de depuración desplegará si corre este comando: **DEBUG DEC 7 + 11**

3. Hay un problema con estos dos comandos. Cuando corra este código, los números que se desplegarán estarán pegados uno al otro como si fueran un número grande en vez de dos números pequeños. Modifique estos dos comandos para que las respuestas aparezcan en diferentes líneas en la terminal de depuración.

   ```
   DEBUG DEC 7 * 11
   DEBUG DEC 7 + 11
   ```

Proyectos

1. Use **DEBUG** para desplegar la solución del problema problema: 1 + 2 + 3 + 4.
2. Salve FirstProgramYourTurn.bs2 con otro nombre. Si tuviera que colocar el comando **DEBUG** mostrado a continuación en la línea justo antes del comando **END** en el programa, ¿que otras líneas podría eliminar y aun hacer que trabaje igual? Modifique la copia del programa para probar su hipótesis (su predicción de lo que pasará).

   ```
   DEBUG "What's 7 X 11?", CR, "The answer is: ", DEC 7 * 11
   ```

Soluciones

Q1. Un módulo microcontrolador BASIC Stamp 2.
Q2. Números binarios, esto es, 0's y 1's.
Q3. La Terminal de Depuración.
Q4. **DEBUG** y **END**
E1. Multiplica los dos operadores 7 y 11, resultando en un producto de 77. El asterisco es el operador multiplicación.
E2. La Terminal de Depuración desplegaría: 18
E3. Para corregir el problema, agregue un retorno de carro usando el caracter de control **CR** y una coma.

```
DEBUG DEC 7 * 11
DEBUG CR, DEC 7 * 11
```

P1. Este es un programa para desplegar una solución al problema: 1+2+3+4.

```
' ¿Qué es un microcontrolador? - Ch01Prj01_Add1234.bs2
'{$STAMP BS2}
'{$PBASIC 2.5}

DEBUG "Cuanto es 1+2+3+4?"
DEBUG CR, "La respuesta es: "
DEBUG DEC 1+2+3+4

END
```

P2. Las últimas tres líneas **DEBUG** pueden ser eliminadas. Se necesita un **CR** adicional después del mensaje "Hola".

```
' ¿Qué és un microcontrolador? - Ch01Prj02_ FirstProgramYourTurn.bs2
' BASIC Stamp envía un mensaje a la Terminal de Depuración.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Hola, soy yo, tu BASIC Stamp!", CR
DEBUG "¿Cuánto es 7 X 11?", CR, "La respuesta es: ", DEC 7 * 11

END
```

La salida desde la Terminal de Depuración es:

```
Hola, soy yo, tu BASIC Stamp!
¿Cuánto es 7 X 11?
La respuesta es: 77
```

Esta salida es la misma que con los códigos anteriores. Este es un ejemplo del uso de comas para emitir mucha información usando sólo un comando **DEBUG** con multiples elementos en el.

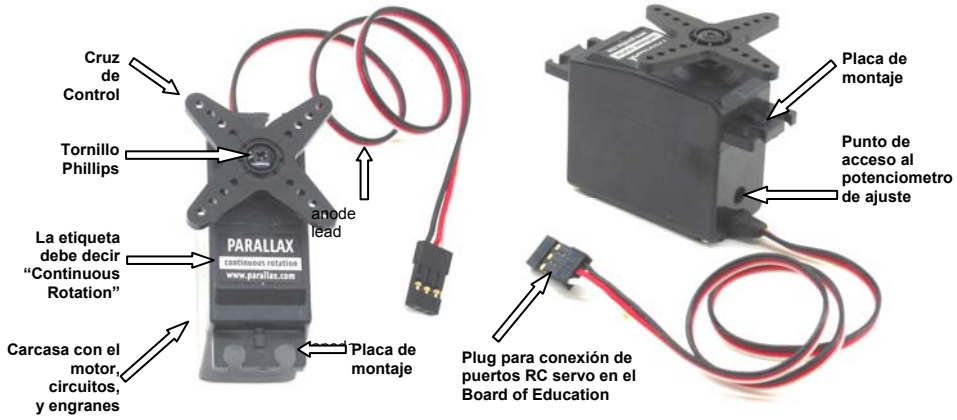
Capítulo 2: Los Servomotores de su Boe-Bot

Este capítulo le guiará a través de la conexión, ajuste y prueba de los motores de su Boe-Bot. Para hacerlo, necesitará entender ciertos comandos PBASIC y técnicas de programación que controlará la dirección, velocidad y duración de los servo movimientos. Así, las actividades 1, 2 y 5 presentarán estas herramientas de programación y luego las actividades tres, 4 y 6 demostrarán cómo aplicarlas en los servos. Puesto que un servocontrol preciso es la clave para el desempeño del robot, ¡completar estas actividades antes de montar los servos en su Boe-Bot es tanto importante como necesario!

PRESENTADO EL SERVO DE ROTACION CONTINUA

Los servos de rotación continua Parallax que se muestran en la Figura 2-1 son los motores que harán girar las ruedas del Boe-Bot. Esta figura muestra las partes externas de los servos. Se hará referencia muchas de estas partes al ir a través de las instrucciones en éste y el siguiente capítulo.

Figura 2-1 Servo de Rotación Continua Parallax



Nota: Puede serle útil hacer una marca en esta página para que pueda regresar aquí más tarde.

Servos Estandar vs. Servos de rotación continua: los servos estándar están diseñados para recibir señales electrónicas que les digan qué posición mantener. Estos servos controlan las posiciones de los flaps en aviones de radio control y el volante de los autos. Los servos de rotación continua reciben las mismas señales electrónicas, pero en vez de mantener cierta posición, giran a ciertas velocidades y direcciones. Los servos de rotación continua son ideales para controlar ruedas y poleas.

Cruz de control del Servo – Estrella de 4 puntos vs. rueda: no hay ninguna diferencia. Mientras que este marcado como "continuous rotation" es el servo para su Boe-Bot. Estará reemplazando la Cruz de control con una rueda.

ACTIVIDAD #1: CONSTRUYENDO Y PROBANDO EL CIRCUITO LED

Controlar la velocidad y dirección de un servo involucra un programa que hace que BASIC Stamp envíe un mismo mensaje una y otra vez. El mensaje se tiene que repetir asimismo 50 veces por segundo para que el servo mantenga su velocidad y dirección. Esta actividad tiene algunos programas ejemplo PBASIC que le demostrarán cómo repetir el mismo mensaje una y otra vez y controlar el tiempo del mensaje.

Desplegando mensajes a velocidad humana

Puede usar el comando **PAUSE** para decirle al BASIC Stamp que espere por un tiempo antes de ejecutar el siguiente comando.

PAUSE ***Duración***

El número que pone a la derecha del comando **PAUSE** es llamado argumento ***Duración*** y su valor le dice que tanto debe esperar antes de moverse al siguiente comando. Las unidades del argumento son milésimas de segundo (ms). Entonces, si quiere esperar por un segundo, debe usar un valor de 1000. He aquí cómo se ve el comando:

```
PAUSE 1000
```

Si quiere esperar el doble, intente:

```
PAUSE 2000
```

Un segundo es abreviado "s." En este texto, cuando ve 1 s, significa un segundo.

Un milisegundo es una milésima de segundo, y se abrevia "ms". El comando **`PAUSE 1000`** retrasa al programa por1000 ms, que es 1000/1000 de un segundo, o sea un segundo, o 1 s. ¿Es claro?

Programa Ejemplo: TimedMessages.bs2

Hay muchas formas de usar el comando **PAUSE**. Este programa ejemplo usa **PAUSE** para causar retraso entre la impresión de mensajes que indican el tiempo transcurrido. El programa espera un segundo antes de enviar el mensaje "One second elapsed…" y otros dos segundos antes de desplegar el mensaje "Three seconds elapsed…".

- ✓ Si tiene un Board of Education, mueva el switch de 3 posiciones de la posición 0 a la posición 1.
- ✓ Si tiene un HomeWork Board, reconecte la bacteria de 9 V a su clip.
- ✓ Introduzca el siguiente programa en el Editor BASIC Stamp.
- ✓ Salvelo como TimedMessages.bs2.
- ✓ Corra el programa y observe el retraso entre mensajes.

```
' Robotica con el Boe-Bot - TimedMessages.bs2
' Muestra como el comando PAUSE se usa para mostrar mensajes a velocidad
humana.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Start timer..."

PAUSE 1000
DEBUG CR, "One second elapsed..."

PAUSE 2000
DEBUG CR, "Three seconds elapsed..."

DEBUG CR, "Done."

END
```

De aquí en adelante, las tres instrucciones que vienen después de este programa serán descritas como sigue:

- ✓ Introduzca, salve y corra TimedMessages.bs2.

Su Turno – diferentes oraciones de pausa

Puede cambiar el retraso entre mensajes cambiando el argumento ***Duración*** en el comando **PAUSE**.

- ✓ Intente cambiar el argumento ***Duración*** de **PAUSE** de 1000 y 2000 a 5000 y 10000, por ejemplo:

```
DEBUG "Start timer..."

PAUSE 5000
DEBUG CR, "Five seconds elapsed..."

PAUSE 10000
DEBUG CR, "Fifteen seconds elapsed..."
```

- ✓ Corra el programa modificado.
- ✓ Intente nuevamente con números como 40 y 100 para los argumentos ***Duración***; irán muy de prisa.
- ✓ El argumento más largo posible es 65535. Si tiene un poco de tiempo, intente **PAUSE 60000**.

Una y otra vez

Una de las mejores cosas acerca de las computadoras y los microcontroladores es que nunca se quejan acerca de hacer las mismas tareas aburridas una y otra vez. Puede colocar comandos entre las palabras **DO** y **LOOP** si quiere hacer que se ejecuten una y otra vez. Por ejemplo, digamos que quiere imprimir un mensaje repetidamente una vez cada segundo. Simplemente coloque sus comandos **DEBUG** y **PAUSE** entre las palabras **DO** y **LOOP** como sigue:

```
DO
  DEBUG "Hello!", CR
  PAUSE 1000
LOOP
```

Programa ejemplo: HelloOnceEverySecond.bs2

- ✓ introduzca, salve y corra HelloOnceEverySecond.bs2.
- ✓ Verifique que el mensaje “Hello!” Se imprime una vez cada segundo.

```
' Robotica con el Boe-Bot - HelloOnceEverySecond.bs2
' Despliega un mensaje cada segundo.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO
  DEBUG "Hello!", CR
  PAUSE 1000
LOOP
```

Su Turno – Un mensaje diferente

Usted puede modificar su programa para qué parte se ejecute solo una vez, y otra parte se ejecute una y otra vez.

- ✓ Modifique el programa para que el comando se vea así:

```
DEBUG "Hello!"
DO
  DEBUG "!"
  PAUSE 1000
LOOP
```

- ✓ ¡Córralo y ver qué pasa! ¿Anticipó el resultado?

ACTIVIDAD #2: RASTREANDO TIEMPO Y REPITIENDO ACCIONES CON UN CIRCUITO

En esta Actividad, construirá circuitos que emitan luz que le permitan "ver" qué clase de señales se usan para controlar los servomotores del Boe-Bot.

¿Qué es un microcontrolador? Esta actividad contiene extractos selectos de la guía del estudiante *¿Qué es un microcontrolador*?

- ✓ Aun cuando esté familiarizado con este material de *¿Qué es un microcontrolador*?, no omita esta actividad.

En la segunda mitad de esta Actividad, examinará las señales que controlan sus acervos y los diagramas de tiempo en una perspectiva diferente de la que fueron presentados en *¿qué es un microcontrolador?*

¡Bono! Los componentes en su kit Boe-Bot pueden ser usados para completar muchas de las actividades en *¿qué es un microcontrolador*? Vaya a www.paralllax.com/go/WAM para una lista completa y baje el texto.

Presentando al LED y la resistencia

Una resistencia es un componente que "resiste" el flujo de electricidad. Éste flujo es llamado corriente. Cada resistencia tiene un valor que le dice que tan fuertemente se resiste al flujo de corriente. Éste valor de resistencia es llamado ohm, y su signo es la letra griega omega: Ω. La resistencia con la que estará trabajando en esta actividad es la resistencia de 470 Ω (Figura 2-2). La resistencia tiene 2 cables (llamados puntas), cada una saliendo de cada extremo. Entre ellas hay un encapsulado cerámico y es la parte que resiste al flujo de corriente. La mayoría de los diagramas de circuitos que muestran resistencias usan el símbolo a la izquierda con línea quebrada para decirle a la persona que construye el circuito que debe usar una resistencia de 470 Ω. Éste se llama símbolo esquemático. El dibujo a la derecha es parte de un dibujo usado en algún texto de Stamps in Class para ayudarle a construir circuitos.

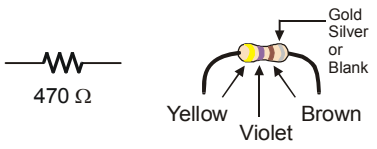


Figura 2-2
dibujo de parte de resistencia de 470 Ω

Símbolo esquemático (izq) y dibujo de parte (derecha)

Las tiras de colores indican los valores de resistencia. Vea el Apéndice B: Códigos de Color de Resistencia y Reglas de tableta en la página 293 para información de cómo determinar el valor de una resistencia a partir de sus tiras de colores en el encapsulado cerámico.

Un diodo es una válvula de corriente de un solo camino, y un diodo emisor de luz (LED) emite luz cuando pasa la corriente a través de el. A diferencia de los códigos de color en una resistencia, el color del LED usualmente sólo le dice de qué color brillará cuando la corriente pase por el. Las marcas importantes en un LED son las contenidas en su forma. Puesto que un LED es una válvula de corriente de un solo camino, tiene que asegurarse de conectarlo en la forma correcta, o no trabajara.

La Figura 2-3 muestra el símbolo esquemático y dibujo de parte. Un LED tiene dos terminales. Una es llamada ánodo y la otra es llamada cátodo. En esta Actividad tendrá que usar él le en un circuito y tendrá que poner atención y asegurarse que el ánodo y el cátodo están conectados al circuito adecuadamente. En el dibujo de parte, la punta de ánodo está etiquetada con el signo más (+). En el símbolo esquemático, el ánodo es la parte ancha del triángulo. En este dibujo de parte, la punta a todo es el pin etiquetado con

un signo menos (-) y en el símbolo esquemático, el cátodo es la línea que parte en la punta del triángulo.

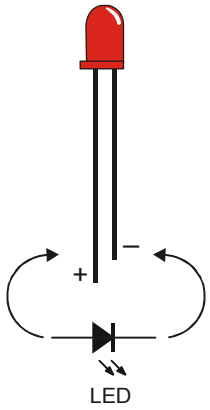


Figura 2-3
dibujo de parte del LED y símbolo esquemático

Dibujo de parte (arriba) y símbolo esquemático (abajo)

En posteriores imágenes, los dibujos de parte del LED tendrán un + cerca del ánodo.

Cuando empiece a construir su circuito, asegúrese de revisarlo contra el símbolo esquemático y dibujo de parte. Sin mirar de cerca el encapsulado plástico del LED en el dibujo de parte, es prácticamente redondo, pero hay un punto pequeño plano cerca de una de las puntas que le dice que es el cátodo. También note que las puntas del LED son de diferentes longitudes. En este texto, el ánodo se mostrará con un signo (+)y el cátodo se mostrará con un signo (–).

Siempre revise el encapsulado plástico del LED. Usualmente, la punta más larga está conectada al ánodo del LED y la punta más corta está conectada a su cargo. Pero algunas veces las puntas han sido recortadas a la misma longitud, o un fabricante no sigue esta convención. Entonces, es mejor siempre buscar el punto plano en el encapsulado. Si conecta un LED al revés, no lo dañará, pero no se encenderá.

Partes para circuito de prueba del LED

(2) LEDs – Rojos
(2) Resistencias, 470 Ω (amarillo-violeta-café)

¡Siempre desconecte la energía de su tableta antes de empezar a modificar circuitos! Para el Board of Education, coloque el switch de 3 posiciones en la posición 0. Para él BASIC Stamp HomeWork Board, desconecte la batería de 9 V de su clip. Siempre busqué posibles errores en su circuito antes de reconectar la energía.

Circuito de prueba del LED

Si completó el texto *¿Qué es un microcontrolador?* estará familiarizado con el circuito mostrado en la Figura 2-4. El lado izquierdo de esta figura muestra el esquemático del circuito, y el lado derecho muestra un ejemplo de diagrama de conexión del circuito construido en el área de prototipo de su tarjeta.

- ✓ Construya el circuito mostrado en la Figura 2-4.
- ✓ Asegúrese de que los pines más cortos en cada LED (los cátodos) están conectados en los sockets marcados Vss.
- ✓ Asegúrese de que los pines más largos (ánodos, marcados ⊕ en el diagrama de conexiones) están conectados a la tableta blanca tal y como se muestra.

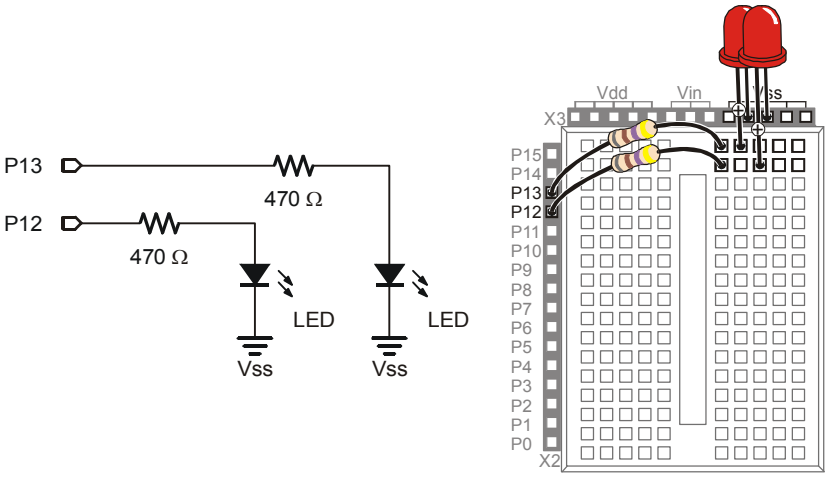


Figura 2-4
Dos LEDs conectados a los pines I/O P13 y P12 del BASIC Stamp

Esquemático (izquierda) y diagrama de conexiones (derecha)

¿Qué es un pin I/O? I/O indica input/output. El BASIC Stamp 2 tiene 24 pines, 16 de los cuales son pines I/O. En este texto, programará el BASIC Stamp para usar pines I/O como salidas para hacer que los LED enciendan y apaguen, controlan la velocidad y dirección de giro de los servos de rotación continua Parallax, hacer tonos con un parlante, y preparar sensores para detectar luz y objetos. También programará el BASIC Stamp para usar los pines I/O para monitorear sensores que indiquen un contacto mecánico, nivel de luz, objetos en la ruta del Boe-Bot, e incluso su distancia.

¿Nuevo construyendo circuitos? Vea el Apéndice B: Códigos de Color de Resistencia y Reglas de tableta en la página 293.

La Figura 2-5 muestra lo que le programará al BASIC Stamp hacer al circuito LED. Imagine que tiene una batería de 5 volt (5 V). Aun cuando una batería de 5 V no es común, el Board of Education tiene un dispositivo llamado regulador de voltaje que suple al BASIC Stamp con el equivalente de una batería de 5 V. Cuando conecta un circuito a Vss, es como si conectara el circuito a la terminal negativa de la batería de 5 V. cuando conecta la otra terminal del circuito a Vdd, es como si conectara la terminal positiva de la batería de 5 V.

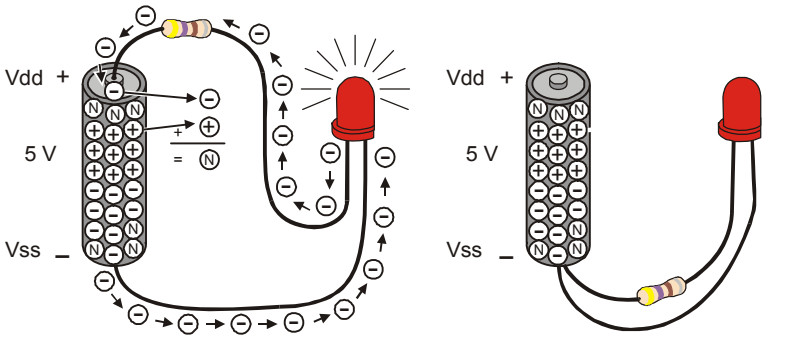


Figura 2-5
corriente y flujo de electrones

***Volts* se abrevia V.** Esto decir que 5 volts se abrevia 5 V. cuando aplica voltaje a un circuito, es como aplicar presión eléctrica.

***Corriente* se refiere al grado en el que los electrones pasan a través de un circuito.** Frecuentemente verá mediciones de corrientes expresadas en amps, que se abrevia A. la cantidad de corriente que un motor maneja es frecuentemente medida en amps, por ejemplo 2 A, 5 A, etc. sin embargo, las corrientes que usará en el Board of Education establecidas en milésimas de un amp, o milliamps. Por ejemplo, 10.3 mA pasa por el circuito en la Figura 2-5.

Cuando se hacen estas conexiones, 5 V de presión eléctrica se aplican al circuito causando que los electrones fluyan a través del LED para emitir luz. En el momento en que desconecte la punta del rey de la resistencia de la terminal positiva de la batería, la corriente deja de fluir y el LED dejar de emitir luz. Puede tomar un siguiente paso conectando la punta de la resistencia a Vss, con el mismo resultado. Esta es la acción que le programará al BASIC Stamp que haga para hacer que el LED se encienda (emita luz) y se apague (no emita luz).

Programas que Controlan los circuitos de prueba del LED

Los comandos **`HIGH`** y **`LOW`** pueden ser usados para hacer que el BASIC Stamp conecte un LED alternativamente a Vdd y Vss. El argumento de ***Pin*** es un número entre 0 y 15 que le dice al BASIC Stamp qué pin I/O pin conectar a Vdd o Vss.

HIGH *Pin*
LOW *Pin*

Por ejemplo, si usa el comando:

```
HIGH 13
```

...le dice al BASIC Stamp conectar el pin I/O P13 a Vdd, lo que enciende al LED.

De igual forma, si usa el comando:

```
LOW 13
```

... le dice al BASIC Stamp conectar el pin I/O P13 a Vss, lo que apaga al LED. Intentémoslo.

Programa ejemplo: HighLowLed.bs2

- ✓ Reconecte la energía a su tarjeta.
- ✓ Introduzca, salve y corra HighLowLed.bs2.
- ✓ Verifique que el circuito circuito LED conectado a P13 se está encendiendo y apagando, una vez por segundo.

```
' Robotica con el Boe-Bot – HighLowLed.bs2
' Enciende/apaga el LED conectado a P13 una vez por segundo.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "The LED connected to Pin 13 is blinking!"

DO

  HIGH 13
  PAUSE 500
  LOW 13
  PAUSE 500

LOOP
```

Cómo trabaja HighLowLed.bs2

La Figura 2-6 muestra como él BASIC Stamp puede conectar un circuito LED alternativamente a Vdd y Vss. Cuando está conectado a Vdd, el LED emite luz. Cuando está conectado a Vss, el LED no emite luz. El comando **`HIGH 13`** instruye al BASIC Stamp a conectar P13 a Vdd. El comando **`PAUSE 500`** instruye al BASIC Stamp dejar el circuito en ese estado por 500 ms. comando **`LOW 13`** instruye al BASIC Stamp to connect the LED a Vss. Nuevamente, el comando **`PAUSE 500`** instruye al BASIC Stamp para dejarlo en ese estado por otros 500 ms. Puesto que estos comandos se colocan entre **`DO`** y **`LOOP`**, se ejecutan una y otra vez.

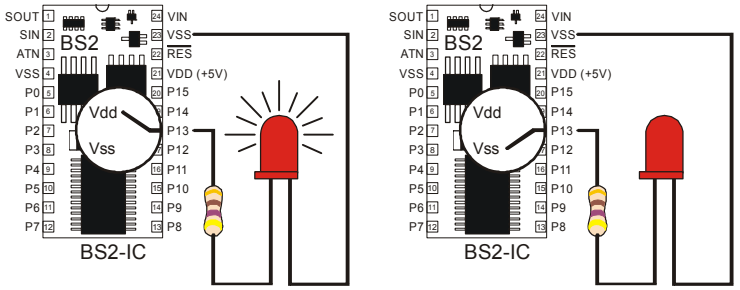


Figura 2-6
Switcheo del BASIC Stamp

El BASIC Stamp puede ser programado para internamente conectar la entrada del circuito LED a Vdd or Vss.

Una prueba de diagnóstico para su computadora

Unas cuantas computadoras, como algunas laptops, detendrán el programa PBASIC antes de su primera pasada a través de la instrucción **`DO...LOOP`**. Estas computadoras tienen un diseño de puerto serie no estándar. Colocando un comando **`DEBUG`** al programa LedOnOff.bs2, la Terminal de Depuración previene que esto ocurra. Ahora volverá a correr este programa sin el comando **`DEBUG`** para ver si su computadora tiene este problema de puerto serie no estándar. No es común, pero es importante para usted saberlo.

- ✓ Abra HighLowLed.bs2.
- ✓ Borre la instrucción completa **`DEBUG`**.
- ✓ Correr programas modificados mientras que observa su LED.

Si el LED parpadear continuamente, tal como lo hizo cuando corrió el programa original con el comando **`DEBUG`**, su computadora no tendrá este problema.

Si el LED parpadea una sola vez y se detiene, tiene una computadora con un diseño de puerto serial no estandar. Si desconecta el cable de programación de su tarjeta y presiona el botón Reset, BASIC Stamp correrá el programa adecuadamente sin congelarse. En sus programas, deberá agregar un solo comando:

```
DEBUG "Program Running!"
```

... justo después de las directivas de compilación. Esto abrirá la terminal de depuración y mantendrá el puerto COM abierto. Esto prevendrá que sus programas se congelen después de la primera pasada por el ciclo **DO…LOOP**, o cualquiera otro de los comandos de cicleo que estará aprendiendo en capítulos posteriores. Verá este comando en algunos de los programas ejemplo que de otra forma no necesitarían la instrucción **DEBUG**. Entonces, debe ser capaz de correr todos los demás programas en este libro aún cuando su computadora haya fallado la prueba diagnóstico.

Presentando el diagrama de tiempo

Un diagrama de tiempo es una gráfica que relaciona señales altas (Vdd) y bajas (Vss) con el tiempo. En la Figura 2-7, el tipo se incrementa de izquierda a derecha, y las señales altas y bajas se alinean ya sea con Vdd (5 V) o Vss (0 V). Este diagrama de tiempo le muestra le muestra una parte de 1000 ms de la señal alta/baja que acaba de experimentar. La línea de puntos a la derecha de la señal (. . .) es una forma de indicar que la señal se repite a sí misma.

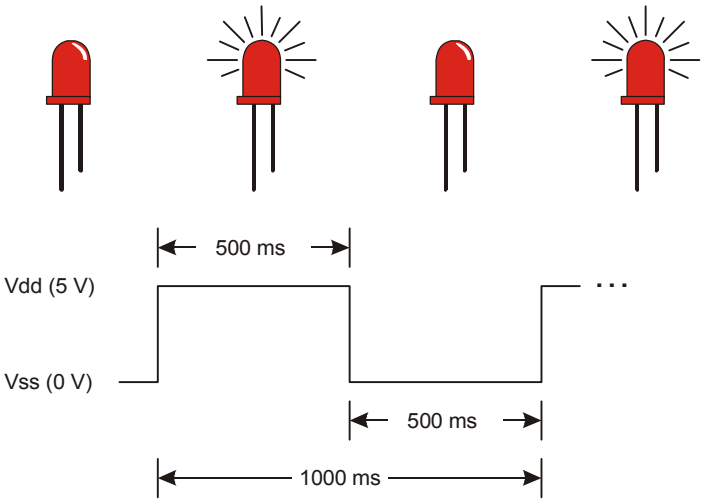


Figura 2-7
diagrama de tiempo para HighLowLed.bs2

Los estados de encendido/apartado del LED se muestran arriba del diagrama de tiempo.

Su turno – Parpadee el otro LED

Hacer parpadear el otro LED (conectado a P12) es un simple asunto de cambiar el argumento ***Pin*** en los comandos **`HIGH`** y **`LOW`** y volver a correr el programa.

- ✓ Modifiquen programas para que el comando se vea como sigue:

```
DO
  HIGH 12
  PAUSE 500
  LOW 12
  PAUSE 500
LOOP
```

- ✓ Corra el programa modificado y verifique que se enciende/apaga el otro LED.

También puede ser que ambos LEDs parpadeen al mismo tiempo.

- ✓ Modifiquen programas para que el comando se vea como sigue:

```
DO
  HIGH 12
  HIGH 13
  PAUSE 500
  LOW 12
  LOW 13
  PAUSE 500
LOOP
```

- ✓ Correr programa modificado y verifique que ambos LEDs parpadean aproximadamente al mismo tiempo.

Puede modificar el programa nuevamente y hacer que un LED parpadee alternativamente, y puede cambiar el tiempo de parpadeo del LED ajustando el argumento ***Duración*** en el comando **`PAUSE`** hacia arriba o hacia abajo.

- ✓ ¡Inténtelo!

Viendo la señal de control del servo con un LED

Las señales altas y bajas que programará en el BASIC Stamp para ser enviada a los servomotor es debe durar cantidades de tiempo muy precisas. Esto es por que los servomotores miden la cantidad de tiempo que la señal está en alto y lo usa como una instrucción de hacia dónde girar. Para un control preciso del servomotor, el tiempo que estas señales están en alto debe ser mucho más preciso de lo que puedo tener con un comando **HIGH** y **PAUSE**. Sólo puede cambiar el argumento ***Duration*** del comando **PAUSE** en 1 ms (recuerde, 1/1000 de segundo) a la vez. Hay un comando diferente llamado **PULSOUT** que puede entregar señales en alto en cantidades de tiempo precisas. Estas cantidades de tiempo son valores que usa en el argumento ***Duration***, ¡y están medidos en unidades de 2 millonésimas de segundo!

PULSOUT ***Pin*****,** ***Duration***

1 microsegundo es una millonésima de segundo. Se abrevia µs. Sea cuidadoso cuando escriba este valor, no es la letra 'u' de nuestro alfabeto; es la letra griega mu 'µ'. Por ejemplo, 8 microsegundos se abrevia 8 µs.

Puede enviar una señal **HIGH** que encienda LED en P13 por 2 µs (2 millonésimas de Segundo) usando este comando:

```
PULSOUT 13, 1
```

Este comando encenderá el LED por 4 µs:

```
PULSOUT 13, 2
```

Este comando envía una señal alta que puede ver:

```
PULSOUT 13, 65000
```

¿Que tanto tiempo el circuito LED conectado a P13 se mantiene encendido cuando envía este pulso? Averigüémoslo. El tiempo que está encendido es 65000 por 2 µs:

$$\begin{aligned} Duration &= 65000 \times 2\,\mu s \\ &= 65000 \times 0.000002\,s \\ &= 0.13\,s \end{aligned}$$

... que aún es muy rápido, 13 centésimas de segundo.

El mayor valor que puede usar en el argumento *Duration* en PULSOUT es 65535.

Programa ejemplo: PulseP13Led.bs2

El diagrama de tiempo en la Figura 2-8 muestra el tren de pulsos que está a punto de enviar al LED con este nuevo programa. Esta vez, las señales alto duran 0.13 segundos y las señales bajo duran 2 segundos. Esto es 100 veces más lento que la señal que el servo necesitará para controlar su movimiento.

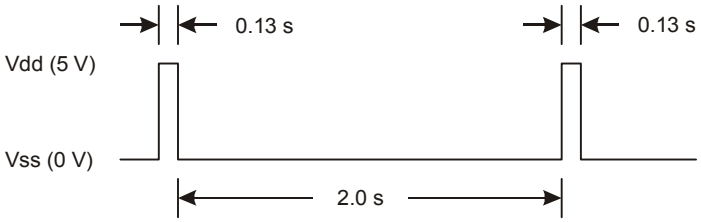


Figura 2-8
diagrama de tiempo para PulseP13Led.bs2

- ✓ Introduzca, salve y corra PulseP13Led.bs2.
- ✓ Verifique que el circuito LED conectado a P13 pulsa por alrededor de 13 centésimas de segundo, una vez cada 2 segundos.

```
' Robotica con el Boe-Bot – PulseP13Led.bs2
' Envía un pulso de 0.13 segundos al circuito LED conectado a P13 cada 2 s.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO

  PULSOUT 13, 65000
  PAUSE 2000

LOOP
```

Programa Ejemplo: PulseBothLeds.bs2

Este programa ejemplo el día un pulso al LED conectado a P13 y luego manda un pulso al LED conectado a P12 como se muestra en la Figura 2-9. Luego, pasa por dos segundos.

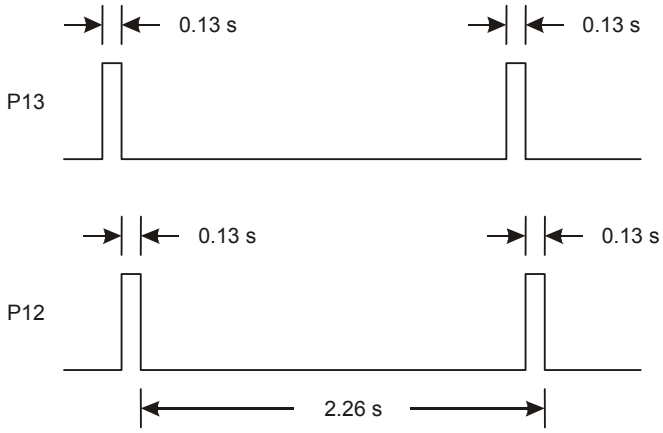


Figura 2-9
Diagrama de tiempo para PulseBothLeds.bs2

Los LEDs emiten luz por 0.13 segundos mientras que la señal está en alto.

Los voltajes (Vdd y Vss) en este diagrama de tiempo no están etiquetados. Con el BASIC Stamp, se entiende que la señal en alto es 5 V (Vdd) y la señal en bajo es 0 V (Vss).

Es una práctica común en documentos que explican el tiempo de señales alto y bajo. Hay uno o más de estos documentos para cada componente dentro del circuito que un ingeniero está diseñando. Los ingenieros que crearon el BASIC Stamp tuvieron que revisar muchos de estos documentos buscando información necesaria para tomar decisiones mientras que se diseñó el producto.

A veces los tiempos también se dejan fuera, o sólo se muestran con una etiqueta como t_{high} y t_{low}. Entonces, los valores de tiempo deseados para t_{high} y t_{low} se enlistan en una tabla en algún lugar después del diagrama de tiempo. Este concepto se discutirá con más detalle en *Basic Analog y Digital,* otra guía del estudiante Parallax Stamps in Class.

- ✓ Introduzca, salve y corra PulseBothLeds.bs2.
- ✓ Verifique que ambos circuitos LED pulsan simultáneamente por alrededor de 13 centésimas de segundo, una vez cada dos segundos.

```
' Robotica con el Boe-Bot – PulseBothLeds.bs2
' Envía un pulso de 0.13 segundos a P13 y P12 cada 2 segundos.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 13, 65000
  PULSOUT 12, 65000
  PAUSE 2000
LOOP
```

Su Turno – Viendo la servo señal a velocidad completa

Recuerde que la servo señal es 100 veces tan rápida como el programa que acaba de correr. Primero, tratemos de correr el programa 10 veces más rápido. Esto quiere decir dividir todos los argumentos ***Duration*** (**PULSOUT** y **PAUSE**) entre 10.

- ✓ Modifique el programa para que el comando se vea como sigue:

```
DO
  PULSOUT 13, 6500
  PULSOUT 12, 6500
  PAUSE 200
LOOP
```

- ✓ Corredor y verifique que hace a parpadear a los LEDs 10 veces más rápido.

Ahora, intentemos correr 100 veces más rápido (una milésima de duración). En vez de aparentar parpadeo, él LED aparecerá no ser tan brillante de como lo era cuando envió una simple señal en alto. Esto es porque él LED está parpadeando tan rápido y por períodos de tiempo tan cortos que el ojo humano no puede detectar el parpadeo actual , sólo ve un cambio en el brillo.

- ✓ Modifique el programa para que el comando se vea como sigue:

```
DO
  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

- ✓ Corra el programa modificado y verifique que haga que ambos LEDs se vean aproximadamente del mismo brillo.
- ✓ Intente sustituir 850 en el argumento ***Duration*** del comando **PULSOUT** de P13.

```
DO
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

- ✓ Corra el programa modificado y verifique que él LED P13 ahora aparece ligeramente más brillante que él LED en P12. Quizá tenga que rodear con sus manos el LED y mirar dentro para ver la diferencia. Difieren porque la cantidad el tiempo que el LED P13está encendido es mayor que la cantidad de tiempo que el P12 permanece encendido.

- ✓ Sustituya 750 en el argumento ***Duration*** en los dos comandos **PULSOUT**.

```
DO
  PULSOUT 13, 750
  PULSOUT 12, 750
  PAUSE 20    a
LOOP
```

- ✓ Corra al programa modificado y verifique que el brillo de ambos LEDs es la misma nuevamente. Pudiera no ser obvio, pero el nivel de brillo está entre los argumentos de ***Duration*** de 650 y 850.

ACTIVIDAD #3: CONECTANDO LOS SERVOMOTORES

En esta actividad, construirá un circuito que conecte el servo a una fuente de energía y a un pin I/O del BASIC Stamp. Los circuitos LED que desarrolló en la última actividad serán usados más tarde para monitorear las señales que el BASIC Stamp envía a los servos para controlar su movimiento.

Partes para conectar los servos

(2) servos de rotación contínua Parallax
(2) circuitos LED construidos y probados de la actividad previa

Encontrando las instrucciones de conexión para su tarjeta

Hay diferentes revisiones del Board of Education y del BASIC Stamp HomeWork Board. Más aún, hay variaciones del Board of Education, basado en la interfase de programación. En el capítulo uno, usó el archivo de ayuda del editor BASIC Stamp para determinar el tipo y revisión de su tarjeta e instrucciones especiales para tarjetas anteriores.

Las instrucciones en este libro fueron escritas para apoyar las tarjetas actualizadas en el momento de escribir este texto y revisiones compatibles anteriores:

- Board of Education Serial - Rev C o posterior
- Board of Education USB - Rev A o posterior
- BASIC Stamp HomeWork Board Serial - Rev C o posterior

✓ Examine el marcado en su tarjeta y note el tipo y revisión.
✓ Para tarjetas anteriores, revise el archivo de ayuda del editor BASIC Stamp para notas específicas de su tarjeta.

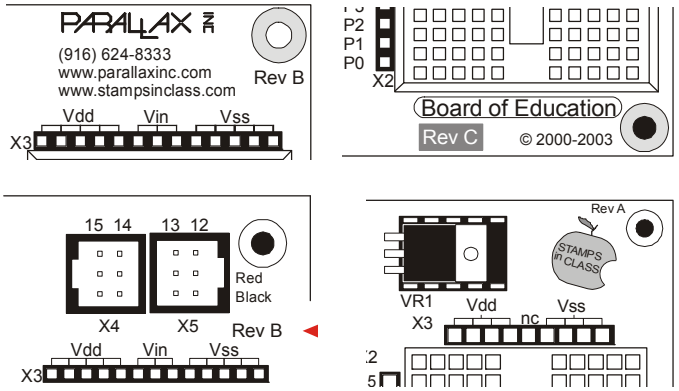


Figura 2-10
Switcheo del BASIC Stamp

El BASIC Stamp puede ser programado para internamente conectar las entradas de los circuitos LED a Vdd or Vss.

✓ Si su tarjeta es una del tipo y revisiones listadas anteriormente, vaya a una de las páginas que siguen:

- Board of Education: vaya a la página 42.
- HomeWork Board: vaya a la página 45.

Conectando los Servos al Board of Education

- ✓ Desenergice colocando el switch de tres posiciones de su Board of Education a la posición 0 (Figura 2-11).

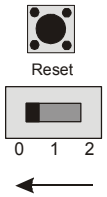


Figura 2-11
Desenrgizando

La Figura 2-12 muestra el cabezal servo header en el Board of Education. Esta tarjeta incluye un conector que puede usar para conectar la fuente de energía de los servos ya sea a Vin o a Vdd. Para moverlo, tendrá que jalar lo hacia arriba y afuera de los pines donde se encuentre, y empujarlo a los pines en donde desea que esté.

- ✓ Si está usando un paquete de baterías de 6 V, asegúrese de que el conector entre los servo puertos en el Board of Education está colocado a Vin como se muestra en la izquierda de la Figura 2-12.

Acerca de las baterías recargables. El Boe-Bot requiere 6 V, fácilmente obtenidos de 4 baterías AA de 1.5 V. Las baterías alcalinas AA son de 1.5 V. sin embargo, muchas baterías recargables AA entregan sólo 1.2 V, dando un total de 4.8 V, que no es suficiente para energizar el BASIC Stamp y el Boe-Bot. Si no puede encontrar baterías recargables de 1.5 V, puede usar el Boe-Boost (#30078) que no es caro para agregar una quinta batería recargable de 1.2 V, regresando del total a 6 V.

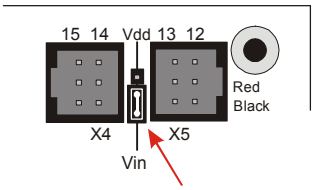
- ✓ Si está usando una fuente de CD a 7.5 V, 1000 mA con centro positivo, coloque el conector a Vdd como se muestra en el lado derecho de la Figura 2-12.

PRECAUCION–el mal uso de fuente CD alimentadas por CA pueden dañar sus servos.

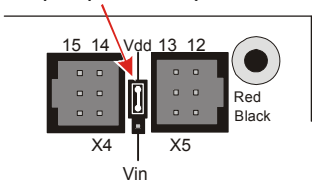
Si no tiene experiencia con fuentes CD, considere apegarse al paquete de baterías de 6 V que vienen con el Boe-Bot.

Sólo use fuentes con voltajes de salida CD entre 6 y 7.5 V y salida de corriente en el rango de 800 mA o más.

Sólo use una fuente de CD QUE esté equipada con la misma clase de plug que el paquete de baterías del Boe-Bot (2.1 mm, centro positivo).



Seleccione Vdd si está usando una fuente CD alimentada por un contacto CA (adaptador CA).

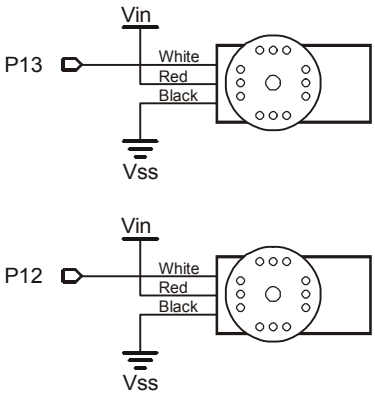


Seleccione Vin si está usando el paquete de batería que viene con los kits Boe-Bot.

Figura 2-12
Seleccione la fuente de energía de sus servo puertos en el Board of Education

Todos los ejemplos e instrucciones en este libro usarán el paquete de baterías. La Figura 2-13 muestra el esquemático del circuito que construirá en el Board of Education. El conector está colocado en Vin.

✓ Conecte sus servos a su Board of Education como se muestra en la Figura 2-13.



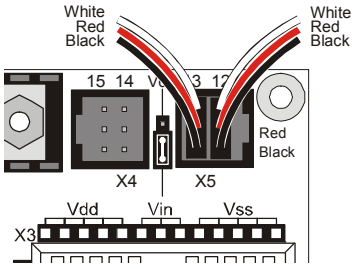


Figura 2-13
Servo Conexiones para el Board of Education

¿Cómo indicó que servo está conectado a P13 y qué servo a P12? Acaba de conectar sus servos en los cabezales con números sobre ellos. Si el número sobre el cabezal donde el servo está conectado es 13, quiere decir que el servo está conectado a P13. Si el número es 12, quiere decir que está conectado a P12.

✓ Cuando termine de ensamblar el sistema, debe semejar a la Figura 2-14 (circuitos LED no mostrados).

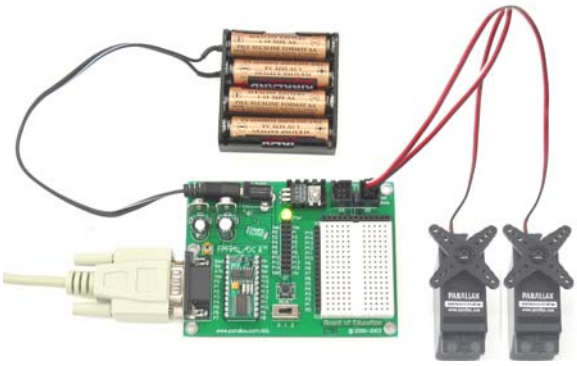


Figura 2-14
Board of Education con Servos y paquetes de batería conectados

✓ Si removió los circuitos LED después de la Actividad #2, reconstrúyalos como se muestra en la Figura 2-15. Serán sus circuitos de monitoreo de la señal servo.

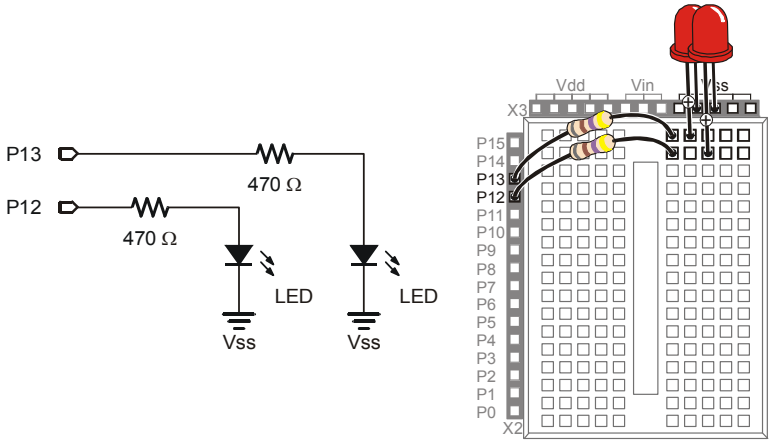


Figura 2-15
Circuito LED de Monitoreo de la Servo señal

Desconectando la energía del Board of Education

nunca dejé la energía conectada a su sistema cuando no está trabajando con él.

✓ Para desconectar la energía de su Board of Education, mueva el switch de 3 posiciones a la posición 0.

✓ Continúe con la Actividad #4: Centrando los Servos en la página 49.

Conectando los Servos al BASIC Stamp HomeWork Board

si está conectando sus servos al BASIC Stamp HomeWork Board, necesitarán las partes listadas abajo y mostradas en la Figura 2-16:

Lista de Partes:

(1) paquete de baterías con puntas estañadas (no incluídas, vea el Apéndice A)
(2) Servos de rotación continua Parallax
(2) conector macho-macho de 3 pines (no incluído, vea el Apéndice A)
(4) cables conectores
(4) baterías alcalinas AA– 1.5 V
(2) circuitos LED construidos y probados de la actividad previa

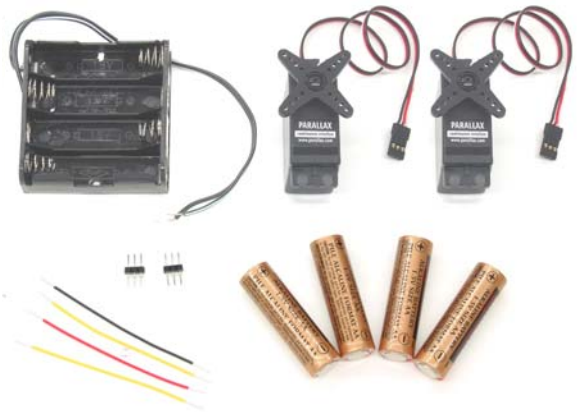


Figura 2-16
partes para centrado del Servo para él HomeWork Board

La Figura 2-17 muestra un esquemático de los circuitos servo en el HomeWork Board. Antes de que empiece a construir este circuito, asegúrese de que la energía está desconectada del BASIC Stamp HomeWork Board.

- ✓ La batería de 9 V debe ser desconectada de su clip y el paquete de baterías no debe tener ninguna batería cargada.

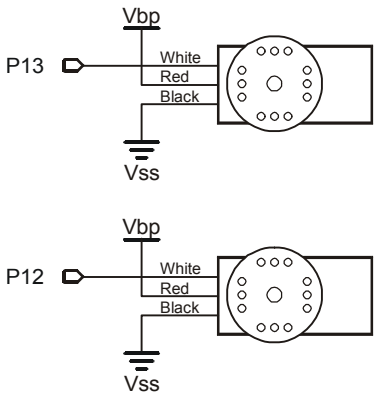


Figura 2-17
esquemático de Conexión Servo para él BASIC Stamp HomeWork Board

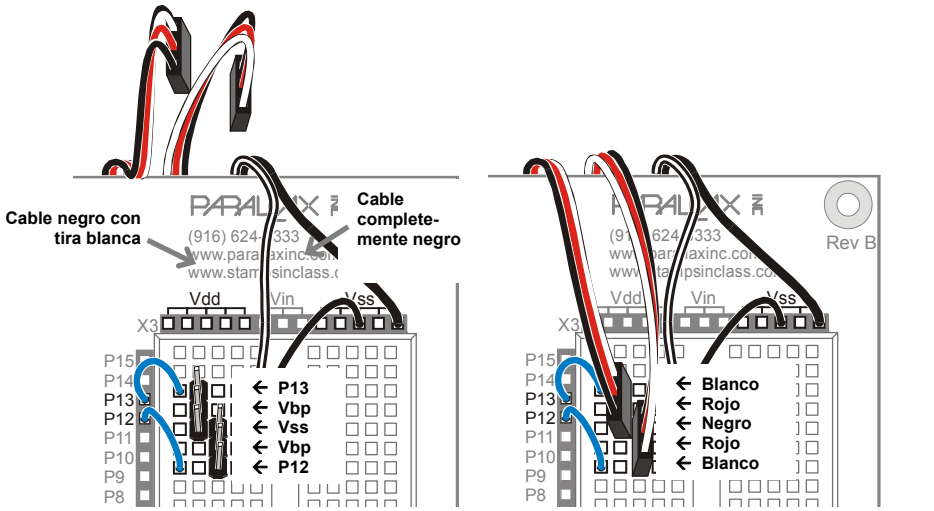
Nota: Vbp indica voltaje del paquete de baterías. Vea el recuadro de información abajo.

- ✓ Remuevan los dos circuitos LED/resistencia, y salve las partes.
- ✓ Construyan los puertos servo mostrados al lado izquierdo de la Figura 2-18.
- ✓ Revise y asegúrese de que el cable negro con tira blanca está conectado a Vbp, y el cable completamente negro esté conectado a Vss.
- ✓ Revise y asegúrese de que todas las conexiones en P13, Vbp, Vss, Vbp (una más), y P12 son exactas como marca el diagrama de conexiones.
- ✓ Conecte los servo plugs a los cabezales macho como se muestra en la Figura 2-18, al lado derecho de la figura.
- ✓ Revise y asegúrese de que los colores en los cables coinciden con la figura.

Vbp indica voltaje battery pack. Se refiere a los 6 VDC suministrados por las cuatro baterías 1.5 V. esto es traído directamente a la tableta para energizar los servos para Boe-Bots con el HomeWork Board. Su BASIC Stamp aún está energizado por la batería de 9 V.

Acerca de las baterías recargables. El Boe-Bot requiere 6 V, fácilmente obtenidos de 4 baterías AA de 1.5 V. Las baterías alcalinas AA son de 1.5 V. sin embargo, muchas baterías recargables AA entregan sólo 1.2 V, dando un total de 4.8 V, que no es suficiente para energizar el BASIC Stamp y el Boe-Bot. Si no puede encontrar baterías recargables de 1.5 V, puede usar el Boe-Boost (#30078) que no es caro para agregar una quinta batería recargable de 1.2 V, regresando del total a 6 V.

Figura 2-18: Diagrama de conexiones de los servos (BASIC Stamp HomeWork Board)



Conexiones del puerto *Servo conexiones por color de cables*

Su armado se debe parecer a la Figura 2-19.

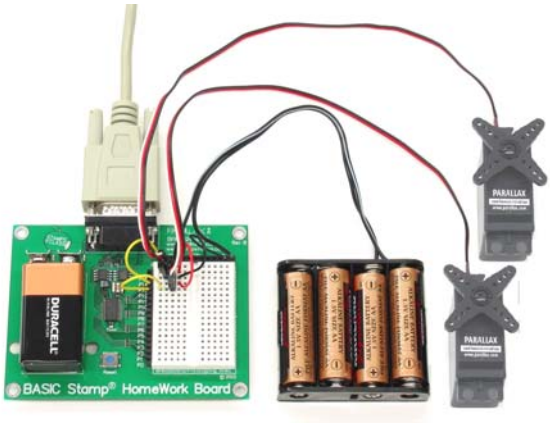


Figura 2-19
Alimentación doble y Servos conectados

✓ Reconstruya el circuito LED como se muestra en la Figura 2-20.

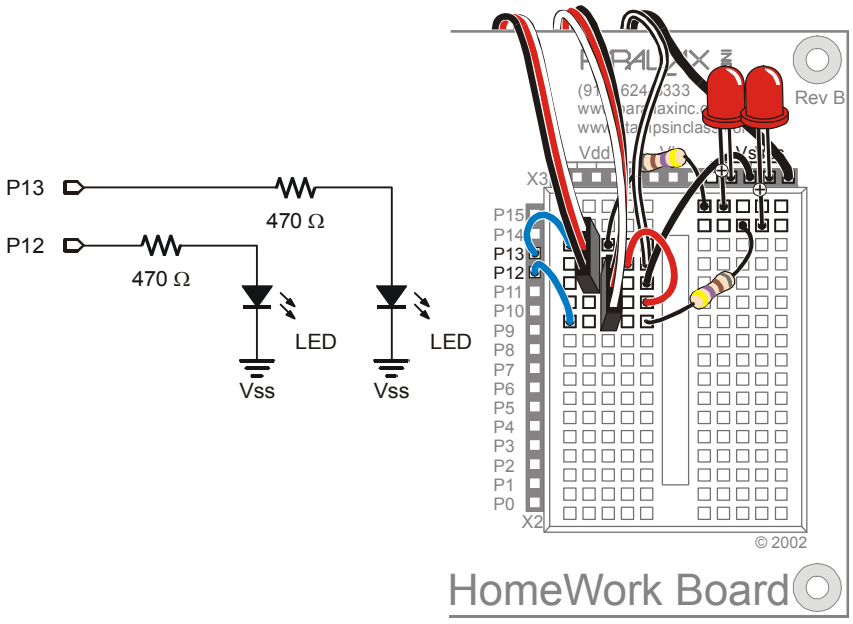


Figura 2-20
Circuito LED de Monitoreo de la señal Servo

✓ Cuando todas sus conexiones estén hechas y revisadas, para el paquete de baterías con baterías y vuelva conectar la batería de 9 V a su clip en el HomeWork Board.

Desconectando la energía en el HomeWork Board

Nunca dejé conectada la energía su sistema cuando no esté trabajando en él. A partir de aquí, la desconexión de la energía tomará dos pasos:

- ✓ Desconecte la batería de 9 V de su clip para desconectar la energía del HomeWork Board. Esto desconecta la energía del BASIC Stamp, y los sockets de energía en la parte superior de la tableta (Vdd, Vin, y Vss).
- ✓ Remuevan una batería del paquete de baterías. Esto desconecta la energía de los servos.

✓ Continúe con la Actividad #4: Centrando los Servos.

ACTIVIDAD #4: CENTRANDO LOS SERVOS

En esta actividad, correrá un programa que envíe una señal a los servos, instruyendolos a que se queden quietos. Debido a que los servos no están ajustados en la fábrica, empezarán a girar. Entonces usará un desarmador para ajustarlos para que se queden quietos. Esto es llamado centrado de los servos. Después del ajuste, probará los servos para asegurarse de que funcionan adecuadamente. Los programas de prueba enviarán señales que hagan que los servos giren a la derecha y a la izquierda a varias velocidades.

Partes y herramientas de los Servos

El desarmador Parallax mostrado en la Figura 2-21 es la única herramienta extra que necesitará para esta actividad. Si es necesario, cualquier desarmador Phillips #1 y cuerpo de 1/8″ (3.18 mm) hará el trabajo.

Figura 2-21
Desarmador Parallax

Enviando la señal de centrado

La Figura 2-22 muestre la señal que debe ser enviada al servo conectado a P12 para calibrar. Esta es llamada la señal de centrado, y después de que el servo ha sido adecuadamente ajustado, esta señal le instruye para quedarse quieto. La instrucción consiste en una serie de pulsos de 1.5 ms con pausas de 20 ms entre cada pulso.

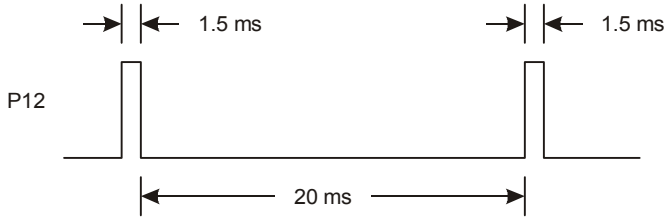


Figura 2-22
diagrama de tiempo para CenterServoP12.bs2

Los pulsos de 1.5 ms instruyen al servo a quedarse quieto.

El programa para esta señal será un comando **PULSOUT** y un comando **PAUSE** DENTRO de un ciclo **DO…LOOP**. Imaginar el comando **PAUSE** a partir del diagrama de tiempo es fácil; será **PAUSE 20** para las 20 ms entre pulsos.

Imaginar el argumento ***Pin*** en el comando **PULSOUT** tampoco es difícil; será 12, para el pin I/O P12. Después, pensemos que el argumento ***Duration*** del comando **PULSOUT**

tendría que ser para pulsos de 1.5 ms. 1.5 ms es 1.5 milésimas de segundo, o 0.0015 s. Recuerde que cualquier número que éste en el argumento ***Duration*** del comando **PULSOUT** multiplica ese número por 2 µs (2 millonésimas de un segundo = 0.000002 s), y sabrá que tanto durará el pulso. También podrá imaginar cuál debe ser el argumento ***Duration*** del comando **PULSOUT** si sabe cuánto desea que dure el pulso. Sólo divida entre 2 µs el número que desea que dure el pulso:

$$Duration\ argument = \frac{Pulse\ duration}{2\mu s} \frac{0.0015\,s}{0.000002\,s} = 750$$

... ahora sabemos que él comando para un pulso de 1.5 ms pulse en P12 será **PULSOUT 12, 750**.

Es mejor centrar un servo a la vez porque de esa manera podrá escuchar cuando el motor se detenta mientras que lo esté ajustando. Este programa sólo enviará la señal de centrado al servo conectado a P12, y las siguientes instrucciones le guiarán a través del ajuste. Cuando complete el proceso con el servo P12, lo repetirá con el servo conectado a P13.

- ✓ Si usted tiene un Board of Education, asegúrese recolocar el switch de 3 posiciones en la posición 2 como se muestra en la Figura 2-23.

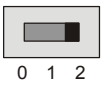


Figura 2-23
Coloque el Switch de 3-Posiciones a la Posicion-2

- ✓ Si está usando el HomeWork Board, revise las conexiones de energía tanto en su BASIC Stamp como en sus servos. La batería de 9 V debe ser conectada al clip y el paquete de batería de 6 V debe tener las cuatro baterías cargadas.

Si los servos empiezan a girar (o a vibrar) en el momento en que con el de la energía:

Probablemente es porque el BASIC Stamp está corriendo un programa que corrió en una actividad previa.

- ✓ Asegúrese de introducir, salvar y correr CenterServoP12.bs2 antes de continuar con las instrucciones de centrado del servo que siguen en el programa ejemplo.

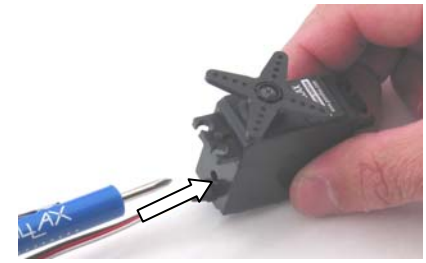
- ✓ Introduzca, salve y corra CenterServoP12.bs2, luego continúe con las instrucciones que siguen al programa.

Programa Ejemplo: CenterServoP12.bs2

```
' Robotica con el Boe-Bot - CenterServoP12.bs2
' Este programa envía pulsos de 1.5 ms al servo conectado a P12 para centrado
' manual.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 12, 750
  PAUSE 20
LOOP
```

Si el servo no ha sido aún centrado, su Cruz empezará a girar y podrá escuchar el motor dentro haciendo ruido.

- ✓ Si el servo aún no está centrado, use un desarmador para ajustar suavemente el potenciómetro en el servo como se muestra en la Figura 2-24. Ajuste lo hasta que encuentre la posición que haga que el servo deje de girar.



1) Inserte la punta de un desarmador Phillips dentro del hoyo de acceso al potenciometro.

2) suavemente gire el desarmador para ajustar el potenciómetro hasta que el servo deje de moverse.

Figura 2-24
ajuste del centro del Servo

Precaucion: ¡no empuje muy fuerte con el desarmador! El potenció metro dentro del servo es bastante delicado, tenga cuidado de no aplicar más presión que la necesaria.

- ✓ Verifique que el circuito LED monitor de la señal conectado a P12 están mostrando actividad. Debería estar emitiendo luz, indicando que los pulsos están siendo transmitidos al servo conectado en P12.

Si el servo ya ha sido centrado, no girará. No es común, pero un servo dañado o defectuoso tampoco girará. La Actividad #6 eliminará esta posibilidad antes de que los servos sean instalados en el chasis de su Boe-Bot.

- ✓ Si el servo no gira, vaya a la sección Su Turno para que pueda probar y centrar el otro servo que está conectado a P13.

¿Qué es un potenciómetro? Un potenciómetro es una especie de resistencia ajustable. La resistencia de un potenciómetro se ajusta con una parte móvil. En algunos potenciómetros, esta parte móvil es un tornillo o una barra deslizante, otros tienen sockets que pueden ser ajustados con desarmadores. La resistencia del potenciómetro dentro del servo de rotación contínua Parallax es ajustado con la punta de un desarmador Phillips #1. Puede aprender más de potenciolos en las guías de estudiante *¿Qué es un Microcontrolador*? y *Basic Analog y Digital*.

Su Turno – centrando el servo conectado a P13

- ✓ repita el proceso para el servo conectado a P13 usando este programa:

Programa Ejemplo: CenterServoP13.bs2

```
' Robotica con el Boe-Bot - CenterServoP13.bs2
' Este programa envía pulsos de 1.5 ms al servo conectado a P13 para centrado
' manual.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 13, 750
  PAUSE 20
LOOP
```

Recuerde completamente desconectar la energía cuando haya terminado.

Si tiene un Board of Education:

- ✓ mueva el switch de 3 posiciones a la posición 0.

Si tiene un BASIC Stamp HomeWork Board:

- ✓ desconecte la batería de 9 V de su clip para desconectar la energía del HomeWork Board, y:
- ✓ Remueva una batería del paquete de baterías.

ACTIVIDAD #5: COMO GUARDAR VALORES Y CUENTAS

Esta actividad introduce variables, que se usan en programas PBASIC para guardar valores. Los programas posteriores para él Boe-Bot en este libro descansa fuertemente en las variables. El punto más importante acerca de ser capaz de guardar valores es que el programa pueda usarlas para contar. Tan pronto como su programa pueda contar, podrá controlar y mantener el rastro del número de veces que algo ocurre.

Sus servos no necesitan ser conectados a la energía para esta actividad.

- ✓ Si tiene un Board of Education, coloque el switch de 3 posiciones a la posición 1. El BASIC Stamp, Vdd, Vin, y Vss estarán todos conectados a la energía, pero no habrá energía conectada a los servo puertos.
- ✓ Si tiene un BASIC Stamp HomeWork Board, conecte la batería de 9 V a su clip para energizar el BASIC Stamp, Vdd, Vin, y Vss. Sólo dejé una batería fuera del paquete de baterías para mantener la energía desconectada de los servos.

Usando Variables para guardar valores, operaciones matemáticas y Conteo

Las variables pueden ser usadas para guardar valores. Antes de que empiece usar una variable en PBASIC, tendrá que darle un nombre y especificar su tamaño. Esto es llamado declaración de una variable.

variableName **VAR** ***Size***

Puede declarar 4 tamaños diferentes de variables en PBASIC:

Tamaño	**–**	**Guarda**
Bit	–	0 a 1
Nib	–	0 a 15
Byte	–	0 a 255
Word	–	0 a 65535, o -32768 to + 32767

El siguiente programa ejemplo involucra un par de variables tipo word:

```
value          VAR     Word
anotherValue   VAR     Word
```

Después de haber declarado una variable como un también puede inicializar la, lo que significa darle un valor inicial.

```
value = 500
anotherValue = 2000
```

Valor por defecto - si no inicializa una variable, el programa automáticamente empezará por guardar el número cero en esa variable. Esto se llama valor por defecto de la variable.

El signo "=" en valor **= 500** es un ejemplo de un operador. Puede usar otros operadores para ser matemáticas con variables. He aquí un par de ejemplos de multiplicación:

```
value = 10 * valor
anotherValue = 2 * valor
```

Programa Ejemplo: VariablesAndSimpleMath.bs2

Este programa demuestra como declarar, inicializar y ejecutar operaciones con variables.

- ✓ Antes del correr el programa, prediga lo que desplegará cada comando **DEBUG**.
- ✓ Introduzca, salve y corra VariablesAndSimpleMath.bs2.
- ✓ Compare los resultados con sus predicciones y explique las diferencias.

```
' Robotica con el Boe-Bot - VariablesAndSimpleMath.bs2
' Declara variables y las usa para resolver algunos problemas aritméticos.
' {$STAMP BS2}
' {$PBASIC 2.5}

value          VAR     Word                 ' Declara variables
anotherValue   VAR     Word

value = 500                                 ' Inicializa variables
anotherValue = 2000

DEBUG ? valor                               ' Despliega valores
DEBUG ? anotherValue

value = 10 * anotherValue                   ' Ejecuta operaciones

DEBUG ? valor                               ' Despliega valores otra vez
DEBUG ? anotherValue

END
```

Cómo trabaja VariablesAndSimpleMath.bs2

Este código declara dos variables tipo word, valor y **anotherValue**.

```
value          VAR     Word      ' Declara variables
anotherValue   VAR     Word
```

éstos comandos son ejemplos de inicialización de variables a valores que usted determina. Después de que estos dos comandos son ejecutados, valor guardará 500, y **anotherValue** guardará 2000.

```
value = 500                     ' Inicializa variables
anotherValue = 2000
```

Estos comandos **DEBUG** le ayudan a ver lo que cada variable guarda después de inicializada. Puesto que a valor le fue asignado 500 y a **anotherValue** le fue asignado 2000, éstos comandos **DEBUG** enviarán los mensajes "value = 500" y "anotherValue = 2000" a la terminal de depuración.

```
DEBUG ? valor                   ' Despliega valores
DEBUG ? anotherValue
```

El formateador del comando DEBUG "?" Puede ser usado antes de una variable para hacer que la terminal de depuración despliegue su nombre, el valor decimal que está guardando, y un retorno de carro es muy útil para ver el contenido de una variable.

La pregunta en las siguientes 3 líneas es "¿Qué es lo que se desplegará?" La respuesta es que valor será igualada a 10 veces **anotherValue**. Puesto que **anotherValue** es 2000, valor sera igualada a 20,000. La variable **anotherValue** no es cambiada.

```
value = 10 * anotherValue      ' Ejecuta operaciones

DEBUG ? valor                   ' Despliega valores otra vez
DEBUG ? anotherValue
```

Su Turno – Cálculos con Números Negativos

Si quiere hacer cálculos que involucren números negativos, puede usar el formateador **SDEC** del comando **DEBUG** para desplegarlos. He aquí un ejemplo que puede hacerse modificando VariablesAndSimpleMath.bs2.

- ✓ Borre esta porción de VariablesAndSimpleMath.bs2:

```
value = 10 * anotherValue      ' Ejecuta operaciones

DEBUG ? valor                   ' Despliega valores otra vez
```

- ✓ Reemplácelo con lo siguiente:

```
value = valor - anotherValue     ' Respuesta = -1500

DEBUG "value = ", SDEC value, CR ' Despliega valores otra vez
```

- ✓ Corra el programa modificado y verifique que valor cambia de 500 a -1500.

Contando y Controlando Repeticiones

La forma más conveniente de controlar el número de veces que una parte de código es ejecutado es con un ciclo **FOR…NEXT**. La sintáxis es:

FOR *Counter* = *StartValue* TO *EndValue* {STEP *StepValue*}…NEXT

Los 3 puntos "..." indican que puede poner uno o mas commandos entre las palabras **FOR** y **NEXT**. Asegúrese de declarar una variable para usarla en el argumento ***Counter***. Los argumentos ***StartValue*** y ***EndValue*** pueden ser numerous o variables (o incluso alguna expresión). Cuando vea algo entre llaves { } en una descripción de sintáxis quiere decir que es un argumento opcional. En otras palabras, el ciclo **FOR…NEXT** trabajará sin ello, pero puede usarlo para algún propósito especial.

No tiene que nombre a la variable "counter." Por ejemplo, puede llamarla "myCounter."

```
myCounter       VAR     Word
```

He aquí un ejemplo de un ciclo **FOR…NEXT** que usa la variable **myCounter** para contar. También despliega el valor de la variable **myCounter** cada vez que se ejecuta el ciclo.

```
FOR myCounter = 1 TO 10
  DEBUG ? myCounter
  PAUSE 500
NEXT
```

Programa Ejemplo: CountToTen.bs2

- ✓ Introduzca, salve y corra CountToTen.bs2.

```
' Robotica con el Boe-Bot – CountToTen.bs2
' Usa una variable en un ciclo FOR...NEXT .

' {$STAMP BS2}
' {$PBASIC 2.5}

myCounter          VAR     Word

FOR myCounter = 1 TO 10
  DEBUG ? myCounter
  PAUSE 500
NEXT

DEBUG CR, "All done!"

END
```

Su Turno – Distintos valores de inicio y fin y Contando en Pasos

Puede usar valores diferentes para los argumentos ***StartValue*** y ***EndValue***.

- ✓ Modifique el ciclo **FOR…NEXT** para que se vea así:

```
FOR myCounter = 21 TO 9
  DEBUG ? myCounter
  PAUSE 500
NEXT
```

- ✓ Corra el programa modificado. ¿Notó que BASIC Stamp contó hacia abajo en vez de hacia arriba? Hará esto siempre que el argumento ***StartValue*** sea mayor que el argumento ***EndValue***.

¿Recuerda el argumento opcional {**STEP** ***StepValue***}? Puede usarlo para hacer que **myCounter** cuente en pasos. En vez de 9, 10, 11…, puede hacer que cuente en pares (9, 11, 13…) o en pasos de 5 (10, 15, 20…), o cualquier ***StepValue*** que desee, hacia adelante o hacia atrás. He aquí un ejemplo que lo usa para contar hacia abajo en pasos de 3:

- ✓ Agregue **STEP 3** al ciclo **FOR…NEXT** para que se vea así:

```
FOR myCounter = 21 TO 9 STEP 3
  DEBUG ? myCounter
  PAUSE 500
NEXT
```

- ✓ Corra el programa modificado y verifique que cuenta hacia atrás en pasos de 3.

ACTIVIDAD #6: PROBANDO LOS SERVOS

Hay una última cosa por hacer antes de ensamblar su Boe-Bot, y es probar los servos. En esta actividad, correrá programas que hacen girar a los servos a diferentes velocidades y direcciones. Al hacer esto, verificará que sus servos están trabajando adecuadamente antes de ensamblar su Boe-Bot. Este es un ejemplo de prueba de un subsistema y es un hábito digno de desarrollar, puesto que ¡no es nada agradable desarmar un robot para arreglar un problema que pudo haber arreglado de otra forma antes de armarlo todo junto!

Prueba de subsistema es la práctica de probar los componentes individuales antes de que se coloquen en un dispositivo mayor. Es una estrategia valiosa que puede ayudarle a ganar concursos de robótica. Es también una habilidad esencial usada por ingenieros en todo el mundo para desarrollar desde juguetes, carros y juegos de video hasta naves espaciales y robots a Marte. Especialmente en dispositivos más complejos, puede ser casi imposible descifrar un problema si los componentes individuales no han sido previamente probados. En Proyectos Aeroespaciales, por ejemplo, desensamblar un prototipo para arreglar un problema puede costar cientos de miles, o quizá millones de dólares. En esa clase de proyectos, la prueba de subsistemas es rigurosa y completa.

El ancho de Pulso Controla Velocidad y Dirección

Al centrar los servos aprendimos que una señal con un ancho de pulso de 1.5 ms causó que los servos se quedaran quietos. Esto fue hecho usando un comando **PULSOUT** con una ***Duration*** de 750. ¿Qué habría pasado si el ancho de pulso no hubiera sido 1.5 ms?

En la sección Su Turno de la Actividad #2, programó el BASIC Stamp para enviar una serie de pulsos de 1.3 ms a un LED. Veamos más de cerca a la serie de pulsos y averigüemos cómo puede usarse para controlar un servo. La Figura 2-25 muestra como un servo de Rotación Continua Parallax gira a velocidad plena hacia la derecha cuando le envía pulsos de 1.3 ms. La velocidad plena varía entre 50 a 60 RPM.

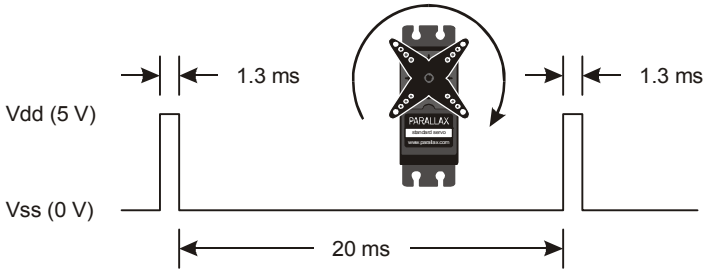


Figura 2-25
Un tren de pulsos de 1.3 ms hace girar al Servo a la derecha a velocidad plena

¿Qué es RPM? Revoluciones Por Minuto. Es el número de vueltas completas que algo gira en un minute.

¿Qué es un tren de pulso? Así como un ferrocarril es una serie de carros, un tren de pulsos es una serie de pulsos.

ServoP13Clockwise.bs2 envía este tren de pulsos al servo conectado a P13.

- ✓ Introduzca, salve y corra ServoP13Clockwise.bs2.
- ✓ Verifique que la cruz del servo esté rotando entre 50 y 60 RPM a la derecha.

```
' Robotica con el Boe-Bot – ServoP13Clockwise.bs2
' Corre el servo conectado a P13 velocidad plena hacia la derecha.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 13, 650
  PAUSE 20
LOOP
```

Note que un pulso de 1.3 ms requiere un argumento ***Duration*** del comando **PULSOUT** de 650, que es menor a 750. Todos los anchos de pulso menores a 1.5 ms, y por lo tanto los argumentos ***Duration*** de **PULSOUT** menores a 750, causarán que el servo gire a la derecha.

Programa Ejemplo: ServoP12Clockwise.bs2

Cambiando el argumento ***Pin*** del comando **PULSOUT** de 13 a 12, puede hacer que el servo conectado a P12 gire a velocidad plena hacia la derecha.

- ✓ Salve ServoP13Clockwise.bs2 como ServoP12Clockwise.bs2.
- ✓ Modifique el programa actualizando los comentarios y el argumento ***Pin*** del comando **PULSOUT** a 12.
- ✓ Corra el programa y verifique que el servo conectado a P12 está ahora girando entre 50 y 60 RPM hacia la derecha.

```
' Robotica con el Boe-Bot – ServoP12Clockwise.bs2
' Corre el servo conectado a P12 a velocidad plena hacia la derecha.
' {$STAMP BS2}
' {$PBASIC 2.5}
```

```
DEBUG "Program Running!"

DO
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

Programa Ejemplo: ServoP12Counterclockwise.bs2

Probablemente ya anticipó que haciendo el argumento ***Duration*** del comando **PULSOUT** mayor que 750 causa que el servo gire a la izquierda. Una ***Duration*** de 850 enviará pulsos de 1.7 ms. Esto hará que el servo gire a velocidad plena hacia la izquierda como se muestra en la Figura 2-26.

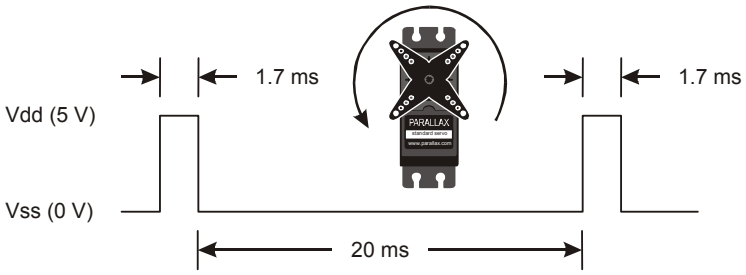


Figura 2-26
Un Tren de Pulsos de 1.7 ms hace que el Servo gire a velocidad plena hacia dla izquierda

- ✓ Salve ServoP12Clockwise.bs2 como ServoP12Counterclockwise.bs2.
- ✓ Modifique el programa cambiando el argumento ***Duration*** del comando **PULSOUT** de 650 to 850.
- ✓ Corra el programa y verifique que el servo conectado a P12 está ahora girandok entre 50 y 60 RPM hacia la izquierda.

```
' Robotica con el Boe-Bot – ServoP12Counterclockwise.bs2
' Corre el servo conectado a P12 a velocidad plena hacia la izquierda.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 12, 850
  PAUSE 20
LOOP
```

Modulación del ancho de Pulso. Un voltaje que se mantiene cierta cantidad de tiempo en 2 estados distintos puede ser considerado como una serie de estados en descanso y pulsos. Una lista de señales de pulso que controlan la velocidad y dirección de su servo es:

- Figura 2-22, pág. 49: 1.5 ms en nivel alto hace que el servo se quede quieto.
- Figura 2-25, pág. 58: 1.3 ms en nivel alto hace que el servo gire a la derecha.
- Figura 2-26, pág. 60: 1.7 ms en nivel alto hace que el servo gire a la izquierda.

Estas señales se mantienen en cortos períodos de tiempo en niveles altos (pulsos) que están separados por señales en nivel bajo (estados en descanso). Un programa puede ajustar la duración del pulso, que es el tiempo que la señal está en nivel alto. Esta duración es comunmente llamada ancho de pulso porque el tiempo que la señal está en nivel alto se ve más ancho o estrecho en un diagrama de tiempo o en un dispositivo como un osciloscopio que traza el voltaje contra el tiempo.

Modulación es el proceso de ajustar una propiedad de una señal que está siendo transmitida para hacerla transmitir cierta información. Con un servo, la propiedad que es modulada es el ancho de pulso, el tiempo que la señal está en nivel alto. La información que transmite es la velocidad y dirección del servo.

Las señales de servo control son ejemplos de pulsos positivos con estados de descanso en nivel bajo y estados activos en nivel alto. Los pulsos negativos serían la versión inversa con estados de descanso en nivel alto y estados activos en nivel bajo.

Su Turno – P13Clockwise.bs2

- ✓ Modifique el argumento ***Pin*** del comando **`PULSOUT`** para que haga que el servo conectado a P13 gire hacia la izquierda.

Programa Ejemplo: ServosP13CcwP12Cw.bs2

Puede usar dos comandoTambién puede hacer que giren en direcciones opuestas.

- ✓ Introduzca, salve y corra ServosP13CcwP12Cw.bs2.
- ✓ Verifique que el servo conectado a P13 está girando a velocidad plena a la izquierda mientras que el que está conectado a P12 lo hace a la derecha.

```
' Robotica con el Boe-Bot - ServosP13CcwP12Cw.bs2
' Corre el servo conectado a P13 a velocidad plena a la izquierda
' y el servo conectado a P12 a velocidad plena a la derecha.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

Esto será importante en breve. Piénselo: cuando los servos son montados a cada lado del chasis, uno tendrá que girar a la derecha mientras que el otro gire a la izquierda para hacer que el Boe-Bot gire sobre su eje. ¿Parece extraño? Si puede visualizarlo, intente esto:

- ✓ Sujete sus servos juntos espalda a espalda y vuelva a corer el programa.

Su Turno – Ajustando la Velocidad y Dirección

Hay 4 combinaciones diferentes de argumentos ***Duration*** y **PULSOUT** que serán usados repetídamente cuando programe los movimientos de su Boe-Bot en los próximos Capítulos. ServosP13CcwP12Cw.bs2 envía uno de estas combinaciones, 850 a P13 y 650 a P12. Probando varias combinaciones posibles y llenando la columna de Descripción de la Tabla 2-1, se familiarizará con ellas y hará una referencia para usted mismo. Llenará la columna Comportamiento después de que su Boe-Bot esté completamente ensamblado, cuando pueda ver cómo hace que se mueva cada combinación.

- ✓ Intente las siguientes combinaciones ***Duration*** y **PULSOUT** y complete la columna Descripción con sus resultados.

Tabla 2-1: Combinaciones PULSOUT y Duration			
Duraciones		**Descripción**	**Comportamiento**
P13	**P12**		
850	650	Velocidad plena, servo P13 gira a la izquierda, servo P12 a la derecha.	
650	850		
850	850		
650	650		
750	850		
650	750		
750	750	Ambos servos deben estar quietos por los ajustes de centrado en la Actividad #4.	
760	740		
770	730		
850	700		
800	650		

FOR…NEXT para Controlar el tiempo de movimiento del Servo

Probablemente para ahora ya comprende que el ancho de pulso controla la velocidad y dirección del servo de Rotación Continua Parallax. Es una forma bastante simple de controlar la velocidad y dirección del motor. También hay una forma simple de controlar la cantidad de tiempo que corre el motor y es con un ciclo **FOR…NEXT**.

He aquí un ejemplo de ciclo **FOR…NEXT** que hará que el servo gire por unos pocos segundos:

```
FOR counter = 1 TO 100
  PULSOUT 13, 850
  PAUSE 20
NEXT
comando
```

Veamos la cantidad exacta de tiempo que este código hará que corra el servo. Cada vez en el ciclo el comando **PULSOUT** dura 1.7 ms, **PAUSE** dura 20 ms, y le toma alrededor de 1.3 ms para ejecutar el ciclo.

Una vez en el ciclo = 1.7 ms + 20 ms + 1.3 ms = 23.0 ms.

Puesto que el ciclo se ejecuta 100 veces, el tiempo es 23.0 ms por 100.

$$\begin{aligned} time &= 100 \times 23.0ms \\ &= 100 \times 0.0230s \\ &= 2.30s \end{aligned}$$

Digamos que quiere que el servo corra por 4.6 segundos. Su ciclo **FOR…NEXT** tendrá que ejecutarse el doble de veces:

```
FOR counter = 1 TO 200
  PULSOUT 13, 850
  PAUSE 20
NEXT
```

Programa Ejemplo: ControlServoRunTimes.bs2

- ✓ Introduzca, salve y corra ControlServoRunTimes.bs2.
- ✓ Verifique que el servo P13 gire a la izquierda por alrededor de 2.3 segundos, seguido por el servo P12 girando por el dible de tiempo.

```
' Robotica con el Boe-Bot - ControlServoRunTimes.bs2
' Corre el servo P13 a velocidad plena a la izquierda por 2.3 s, luego
' corra el servo P12 por el doble de tiempo.
' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter VAR Byte

FOR counter = 1 TO 100
  PULSOUT 13, 850
  PAUSE 20
NEXT

FOR counter = 1 TO 200
  PULSOUT 12, 850
  PAUSE 20
NEXT

END
```

Digamos que quiere correr ambos servos, el servo P13 a un ancho de pulso de 850 y el servo P12 a un ancho de pulso de 650. Ahora, cada vez que pase por el ciclo, tomará:

1.7ms	*–*	*Servo conectado a P13*
1.3 ms	*–*	*Servo conectado a P12*
20 ms	*–*	*Duración de la Pausa*
1.6 ms	*–*	*Tiempo del código*
---------		*----------*
24.6 ms	*–*	*Total*

Si quiere corer los servos por una cierta cantidad de tiempo, puede calcularlo así:

Númeor de pulsos = Tiempo s / 0.0246 s = Tiempo / 0.0246

Digamos que queremos corer los servos por 3 segundos. Esto es:

Número de pulsos = 3 / 0.0246 = 122

Ahora, puede usar el valor 122 en ***EndValue*** del ciclo **FOR…NEXT**, y se verá así:

```
FOR counter = 1 TO 122
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

Programa Ejemplo: BothServosThreeSeconds.bs2

He aquí un ejemplo de cómo hacer que los servos giren en una dirección por 3 segundos, luego invirtiendo su dirección.

- ✓ Introduzca, salve y corra BothServosThreeSeconds.bs2.

```
' Robotica con el Boe-Bot - BothServosThreeSeconds.bs2
' Corre ambos servos en direcciones opuestas por 3 segundos, luego invierte
' la direccion de ambos servos y corre otros 3 segundos.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter VAR Byte

FOR counter = 1 TO 122
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT

FOR counter = 1 TO 122
  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20
NEXT

END
```

- ✓ Verifique que cada servo giró en una dirección por tres segundos, luego cambiaron de dirección y giraron por tres segundos más. ¿Notó que mientras que los servos cambiaron de dirección al mismo tiempo siempre estuvieron girando en direcciones opuestas? ¿Cómo podría ser esto útil?

Su Turno – prediga el tiempo de giro del Servo

- ✓ escoja un tiempo (seis segundos o menos), que desee que sus servos giren.
- ✓ Divida el número de segundos entre 0.024.
- ✓ Sus respuestas el número de ciclos que necesitará.
- ✓ Modifique BothServosThreeSeconds.bs2 para hacer que sus servos giren por el tiempo que seleccionó.
- ✓ Compare su tiempo de giro predicho contra el tiempo de giro actual.
- ✓ Recuerde desconectar la energía de su sistema (tarjeta y servos) cuando haya terminado.

TIP – para medir el tiempo de giro, mantenga presionado el botón Reset en su Board of Education (o BASIC Stamp HomeWork Board). Cuando esté listo para empezar a medir el tiempo, libere el botón Reset.

RESUMEN

Este Capítulo le guió a través de la conexión, ajuste y prueba de los servos de rotación continúa Parallax. Al hacerlo, se presentaron una variedad de comandos PBASIC. El comando **PAUSE** hace que el programa se detenga por un breve o por largos períodos de tiempo, dependiendo del argumento ***Duration*** que utilice. **DO…LOOP** hace repetir uno solo o un grupo de comandos PBASIC una y otra vez. **HIGH** y **LOW** fueron presentados como una forma de hacer que el BASIC Stamp conecte un pin I/O a Vdd or Vss. Señales de alto y bajo nivel fueron vistas con ayuda de un circuito LED. Estas señales fueron usadas para presentar los diagramas de tiempo.

El comando **PULSOUT** fue presentado como una forma más precisa de entregar una señal de alto o bajo nivel, y un circuito LED también fue usado para ver las señales enviadas por el comando **PULSOUT**. Luego, se usaron **DO…LOOP**, **PULSOUT**, y **PAUSE** para enviar una señal para mantener quietos a los servos de rotación continúa Parallax, que es un tren de pulsos de 1.5 ms cada 20 ms. El servos ajustó con un desarmador mientras que recibía los pulsos de 1.5 ms hasta que se mantuvieron quietos. Este proceso es llamado "centrado" del servo.

Luego de que los servos fueron centrados, las variables fueron presentadas como una forma de guardar valores. Las variables pueden ser usadas en operaciones matemáticas y para conteo. Se presentaron los ciclos **FOR…NEXT** como una forma de contar. Los ciclos **FOR…NEXT** controlan el número de veces que se ejecutan las líneas de código que se encuentran entre los comandos **FOR** y **NEXT**. Entonces los ciclos **FOR…NEXT** fueron usados

para controlar el número de pulsos entregados a un servo, lo cual controla la cantidad de tiempo que gira un servo.

Preguntas

1. ¿En qué difieren los servos de rotación continúa Parallax de los servos estándar?
2. ¿Cuanto dura un milisegundo? ¿Como lo abrevia?
3. ¿Qué comandos PBASIC puede usar para hacer que otros comandos PBASIC se ejecuten una y otra vez?
4. ¿Qué comando causa que él BASIC Stamp internamente conecte uno de sus pines I/O a Vdd? ¿Qué comando causa la misma conexión, pero a Vss?
5. ¿Cuales son los nombres de los diferentes tamaños de variables que pueden ser declarados en un programa PBASIC? ¿Cual es el tamaño de los valores que cada tamaño de variable puede guardar?
6. ¿Cual es la clave para controlar la dirección y velocidad de un servo de rotación continúa Parallax? ¿Cómo se relaciona esto con los diagramas de tiempo? ¿Qué comando y argumento puede ajustar para controlar la dirección y velocidad de un servo de rotación continúa?

Ejercicios

1. Escriba un comando **PAUSE** t que haga que el BASIC Stamp no haga nada por 10 segundos.
2. Modifique este ciclo **FOR…NEXT** PARA que cuente del 6 al 24 en pasos de 3. También, escriba la declaración de variable que necesitará para hacer que este programa funcione.

```
FOR counter = 9 TO 21
  DEBUG ? counter
  PAUSE 500
NEXT
```

Proyecto

1. escriba un programa que cause que un LED conectado a P14 se encienda atenuado (encendido/apagado con cada pulso) mientras que el servo P12 este girando.
2. Escriba un programa que haga funcionar los servos durante tres segundos en cada una de las cuatro diferentes combinaciones de rotación. Dato: necesitará cuatro ciclos diferentes **FOR…NEXT**. Primero, ambos servos deben rotar a la izquierda, luego del en ambos rotar a la derecha. Luego, el servo P12 debe girar

a la derecha mientras que el servo P13 rota la izquierda y, finalmente, el servo P12 debe girar a la izquierda mientras que el servo P13 gira a la derecha.

Soluciones

Q1. en vez de mantener una cierta posición como un servo estándar, los servos de rotación continúa Parallax giran en una cierta dirección y a una cierta velocidad.
Q2. Un milisegundo dura una milésima de segundo, y se abrevia "ms".
Q3. El comando **`DO…LOOP`** se usa para hacer que otros comandos PBASIC se ejecuten una y otra vez.
Q4. **`HIGH`** conecta un pin I/O a Vdd, **`LOW`** conecta un pin I/O a Vss.
Q5. Los tamaños de variables son bit, nib, byte, y word.

`Bit` – guarda 0 a 1
`Nib` – guarda 0 a 15
`Byte` – guarda 0 a 255
`Word` – guarda 0 a 65535 o -32768 a +32767

Q6. el ancho de pulso controlan la dirección y velocidad del servo. Como se puede ver en un diagrama de tiempo, el ancho de pulso es el tiempo en nivel alto. En PBASIC, el pulso puede ser generado con el comando **`PULSOUT`**. El argumento ***Duration*** del comando **`PULSOUT`** ajusta la velocidad y dirección.

E1. **`PAUSE 10000`**
E2. la clave para escribir la declaración de variable es escoger un tamaño de variable lo suficientemente grande para guardar el valor 24. Un **`Nib`** no trabajará, puesto que el valor máximo que un nibble puede guardar es 15. Por lo tanto, escoja una variable Byte.

```
counter VAR Byte
FOR counter = 6 TO 24 STEP 3
  DEBUG ? counter
  PAUSE 500
NEXT
```

P1. La clave para resolver este problema es enviar un tren de pulsos al LED así como también al servo.

```
' Robotica con el Boe-Bot - Ch02Prj01_DimlyLitLED.bs2
' Corre el servo y envia la misma señal para atenuar el LED en P14.
'{$STAMP BS2}
'{$PBASIC 2.5}

DEBUG "Program Running!"

DO
  PULSOUT 12, 650                          ' P12 servo a la derecha
  PULSOUT 14, 650                          ' P14 LED prende atenuado
  PAUSE 20
LOOP
```

P2. Primero, calcule el número de ciclos necesarios para hacer que los servos corren por tres segundos, para cada combinación de rotación. Según lo indicado en la página 65, el tiempo propio del código es 1.6 ms.

Cuatro combinaciones (1, 2, 3, 4).
Cada combinación: Determine los argumentos **PULSOUT** ***Duration***:

1. Ambos a la izquierda: 12, 850 y 13, 850
2. Ambos a la derecha: 12, 650 y 13, 650
3. 12 derecha y 13 izquierda: 12, 850 y 13, 650
4. 12 izquierda y 13 derecha: 12, 650 y 13, 850

Cada combinación: Calcule cuánto tomará cada ciclo:

1. Un ciclo = 1.7 + 1.7 + 20 ms + 1.6 = 25.0 ms = 0.025 s
2. Un ciclo = 1.3 + 1.3 + 20 ms + 1.6 = 24.2 ms = 0.0242 s
3. Un ciclo = 1.7 + 1.3 + 20 ms + 1.6 = 24.6 ms = 0.0246 s
4. Un ciclo = 1.3 + 1.7 + 20 ms + 1.6 = 24.6 ms = 0.0246 s

Cada combinación: Calcule el número de pulsos necesarios para correr 3 s:

1. Número de pulsos = 3 s / 0.025 s = 120
2. Número de pulsos = 3 s / 0.0242 s = 123.9 = 124
3. Número de pulsos = 3 s / 0.0246 s = 121.9 = 122
4. Número de pulsos = 3 s / 0.0246 s = 121.9 = 122

Ahora escriba cuatro ciclos **FOR…NEXT**, usando el número de pulsos calculados para el argumento ***EndValue***. Incluye los argumentos **PULSOUT** correctos para la combinación de rotación.

```
' Robotica con el Boe-Bot - Ch02Prj02_4RotationCombinations.bs2
' Mueve servos en las 4 combinaciones de rotaciones derecha/izquierda

'{$STAMP BS2}
'{$PBASIC 2.5}

DEBUG "Program Running!"

counter        VAR     Word

FOR counter = 1 TO 120             ' ciclo por tres segundos
  PULSOUT 13, 850                  ' servo P13 a la izquierda
  PULSOUT 12, 850                  ' servo P12 a la izquierda
  PAUSE 20
NEXT

FOR counter = 1 TO 124             ' ciclo por tres segundos
  PULSOUT 13, 650                  ' servo P13 a la derecha
  PULSOUT 12, 650                  ' servo P12 a la derecha
  PAUSE 20
NEXT

FOR counter = 1 TO 122             ' ciclo por tres segundos
  PULSOUT 13, 650                  ' servo P13 a la derecha
  PULSOUT 12, 850                  ' servo P12 a la izquierda
  PAUSE 20
NEXT

FOR counter = 1 TO 122             ' ciclo por tres segundos
  PULSOUT 13, 850                  ' servo P13 a la izquierda
  PULSOUT 12, 650                  ' servo P12 a la derecha
  PAUSE 20
NEXT

END
```

Capítulo 3: Ensamble y Pruebe su Boe-Bot

Este Capítulo contiene instrucciones para construir y probar su Boe-Bot. Es especialmente importante completar la parte de pruebas antes de continuar con el siguiente capítulo. Al hacerlo, ayudará a evitar un número de errores comunes que le llevarían a interpretar erróneamente el comportamiento del Boe-Bot. He aquí un resumen de lo que tendrá que hacer en cada una de las actividades:

Actividad	Resumen
1	Construir el Boe-Bot.
2	Volver a probar los servos para asegurarse que están bien conectados.
3	Conectar y probar un parlante para saber cuándo están bajas las baterías del Boe-Bot.
4	Controlar y probar la velocidad del servo con la Terminal de Depuración.

ACTIVIDAD #1: ENSAMBLANDO EL ROBOT BOE-BOT

Esta actividad le guiará a través del ensamblado del Boe-Bot, paso a paso. En cada paso, reunirá algunas de las partes y entonces las ensamblará para que iguale las imágenes. Cada imagen tiene instrucciones; asegúrese de seguirlas cuidadosamente.

Herramientas y partes del Servo

Todas las herramientas mostradas en la Figura 3-1 son comunes y pueden ser encontradas en la mayoría de las casas y talleres escolares. También pueden ser compradas en ferreterías locales.

Herraminetas

(1) desarmador Parallax
(Phillips #1, incluído)
(1) llame mixta de 1/4″
(Opcional pero útil)
(1) pinzas de punta (opcionales)

Figura 3-1
herramientas para ensamble del Boe-Bot

Montando los elementos físicos superiores

- ✓ Empiece por reunir esta lista de partes.
- ✓ Luego, siga las instrucciones.

Lista de partes:

Vea la Figura 3-2.

(1) chasis del Boe-Bot
(4) extensiones de 1″
(4) tornillos, 1/4″ 4-40
(1) gomas para barrenos, 13/32″

Instrucciones:

- ✓ Inserte las gomas de 13/32″ En los barrenos del centro del chasis del Boe-Bot.
- ✓ Asegúrese de que la secta en la parte exterior de la goma está asentada sobre el borde del barreno del chasis.
- ✓ Use los cuatro tornillos 1/4″ 4-40 para unir las cuatro extensiones al chasis como se muestra.

Partes del Boe-Bot - las partes para el Boe-Bot están ya sea incluidas en el kit completo del Boe-Bot o en una combinación del Kit completo Board of Education y del Kit de Partes de Robótica. Si esta usando un HomeWork Board, necesitará un paquete de baterías con puntas estañadas y dos conectores de tres pines adicionales. Vea el Apéndice A: Lista de Partes y Opciones en la página 289 para más información.

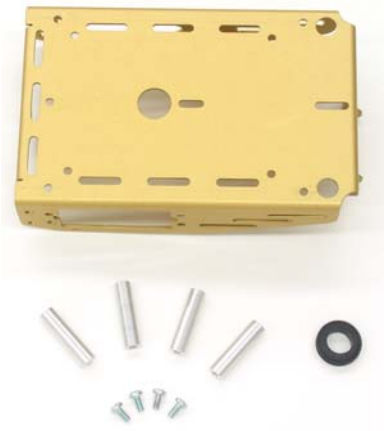


Figura 3-2
Chasis y elementos físicos superiores

Partes (izq.); ensamble (derecha)

Removiendo las cruces de control de los Servos

- ✓ Desconecte la energía de su BASIC Stamp y de sus servos.
- ✓ Remueva todas las baterías AA del paquete de baterías.
- ✓ Desconecte los servos de su tarjeta.

Lista de partes:

Vea la Figura 3-3.

(2) servos de rotación continúa Parallax, previamente centrados

Instrucciones:

- ✓ Use un desarmador Phillips para remover los tornillos que sostienen las cruces de control de los servos en sus ejes de salida.
- ✓ Jale cada Cruz hacia arriba y afuera del eje de salida del servo.
- ✓ Guarde los tornillos; serán usados en un paso posterior.

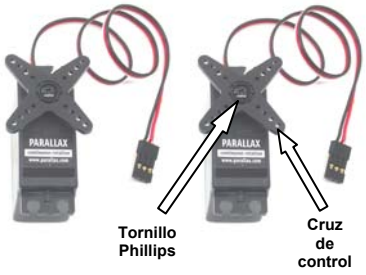


Figura 3-3
Chasis y elementos físicos superiores

Partes (izquierda); ensamble (derecha)

¡Alto!

- ✓ Antes del siguiente paso, debe haber completado las siguientes actividades del Capítulo 2: Los Servomotores de su Boe-Bot
  - Actividad #3: ; página 40.
  - Actividad #4: Centrando los Servos; página 49.

Montando los Servos en el chasis

Opciones de montaje- maniobras ágiles vs. acceso al potenciómetro y mantenimiento

Las fotografías en este texto muestran a los servos montados desde adentro y orientados para que el puerto de acceso del potenciómetro quede viendo al centro del chasis. Esto posiciona los ejes cerca del centro del Boe-Bot, permitiendo un maniobraje ágil. Si centra sus servos diligentemente antes de construir su Boe-Bot, esto no causará problemas.

Muchos educadores optan por montar los servos desde afuera y orientarlos para que el acceso al potenciómetro quede viendo al frente del Boe-Bot. Esto facilita el acceso para ajustar estos potenciómetros en un robot ensamblado, y también para reemplazo fácil de los servos dañados. Sin embargo, el Boe-Bot tendrá una base de giro más larga y ancha y será un poco más lento en maniobras. Quizá tenga que ajustar algunos valores en sus programas ligeramente para lograr los mismos resultados. La elección es suya.

Lista de partes:

Vea la Figura 3-4.

(2) Chasis del Boe-Bot (semi armado).
(2) servos de rotación continúa Parallax
(8) tornillos, 3/8″ 4-40
(8) tuercas, 4-40

Instrucciones:

- ✓ Una los servos al Chasis usando los tornillos Phillips y las tuercas.
- ✓ Use masking tape para etiquetar los servos izquierda (L) y derecha (R).

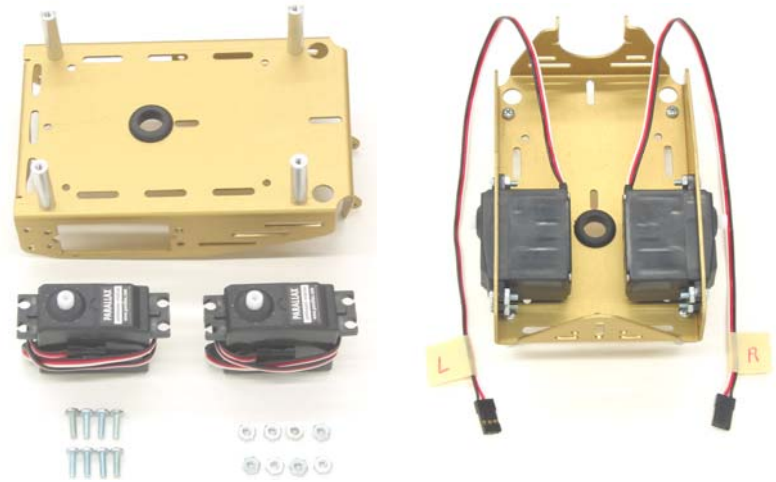


Figura 3-4 montando los Servos en el Chasis

Partes (izquierda); ensamble (derecha)

Montando el paquete de baterías

La Figura 3-5 muestra dos conjuntos diferentes de partes. Use las partes a la izquierda si tiene un Board of Education, y las partes a la derecha si tiene un HomeWork Board.

Lista de partes para Boe-Bot con un Board of Education:

Véa la Figura 3-5 (lado izquierdo).

(1) Chasis Boe-Bot (semi armado)
(2) tornillos Phillips cabeza plana,3/8″4-40
(2) tuercas, 4-40
(1) Paquete de baterías con plug centro +

Lista de partes para Boe-Bot con un HomeWork Board:

Véa Figura 3-5 (lado derecho).

(1) Chasis Boe-Bot (semi armado)
(2) tornillos Phillips cabeza plana,3/8″4-40
(2) tuercas, 4-40
(1) Paquete de baterías, puntas estañadas

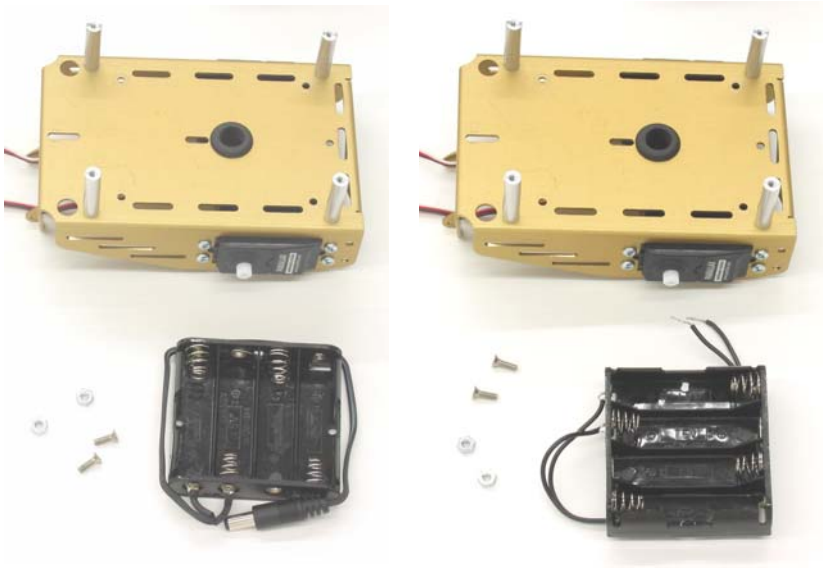


Figura 3-5 elementos físicos para montaje del Paquete de baterías

Instrucciones:

- ✓ Use los tornillos de cabeza plana y tuercas para unir el paquete de baterías a la cara inferior del chasis del Boe-Bot (lado izquierdo de la Figura 3-6).
- ✓ Asegúrese de insertar los tornillos a través del paquete de baterías y luego apriete las tuercas en la parte superior del chasis.

- ✓ Pase el cable de energía del paquete de baterías a través del barreno cubierto con la goma en el centro del chasis (lado derecho de la Figura 3-6).
- ✓ A las las líneas del servo a través del mismo barreno.
- ✓ Acomode las líneas del servo y el cable de alimentación como se muestra.

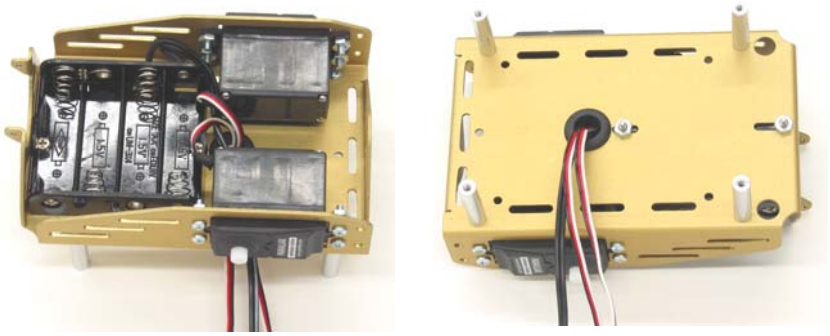


Figura 3-6
Paquete de baterías instalado

Montando las ruedas

Lista de partes:

(1) Boe-Bot semi armado (no se muestra)
(1) horquilla 1/16″
(1) llanta posterior tipo esfera
(2) llantas tipo liga
(2) ruedas plásticas maquinadas
(2) tornillos guardados al remover las cruces de control de los servos

Figura 3-7
elementos físicos de la rueda

Instrucciones:

El lado izquierdo de la Figura 3-8 muestra la rueda posterior del Boe-Bot montada en el chasis. La rueda posterior es solamente una bola de plástico con un barreno a través de su centro. Una horquilla la sostiene al chasis y funciona como eje de la rueda.

- ✓ Alinee el barreno en la llanta posterior y los de la porción trasera del chasis.
- ✓ Pase la horquilla por los 3 barrenos (chasis izquierdo, rueda, chasis derecho).
- ✓ Doble las puntas de la horquilla separandolas para que la rueda no pueda salir.

La Figura 3-8 (der.) muestra las ruedas de tracción del Boe-Bot montadas en los servos.

- ✓ Estire cada llanta tipo liga y colóquela sobre el perímetro de cada rueda.
- ✓ Cada rueda de plástico tiene un saque o caja que embona en el eje de salida del servo. Presione cada rueda de plástico sobre el eje de salida del servo asegurándose que le sea en línea y se hunde en el saque.
- ✓ Use los tornillos que guardó cuando removió la Cruz de los servos para unir las fuerzas a los ejes de salida de los servos.

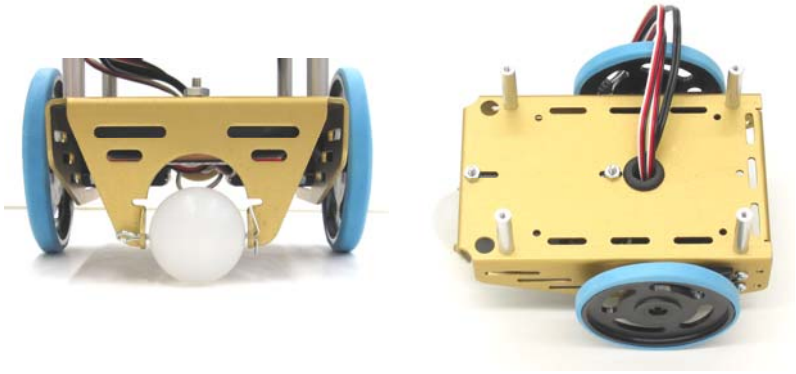


Figura 3-8
montando las ruedas

Rueda posterior (izquierda); ruedas de tracción (derecha)

Uniendo la tarjeta al Chasis

Lista de partes para Boe-Bot con un Board of Education:

Véa el lado izquierdo de la Figura 3-9.

(1) Chasis Boe-Bot (semi ensamblado)
(4) tornillos cabeza de gota, 1/4″ 4-40
(1) Board of Education con BASIC Stamp 2

Lista de partes para Boe-Bot con un HomeWork Board:

Véa el lado derecho de la Figura 3-9.

(1) Chasis Boe-Bot (semi ensamblado)
(4) tornillos cabeza de gota, 1/4″ 4-40
(1) BASIC Stamp HomeWork Board

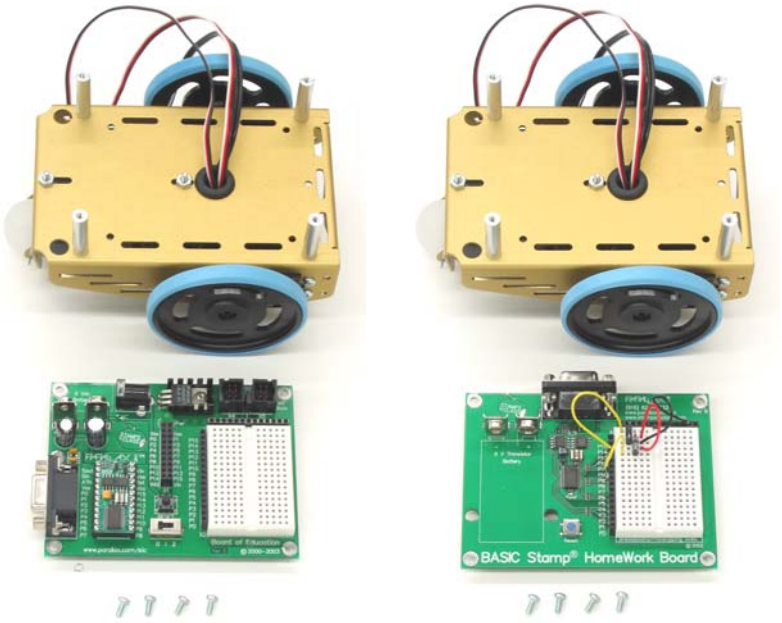
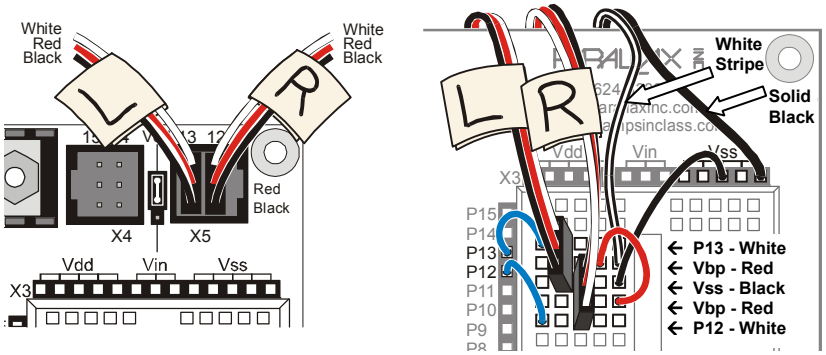


Figura 3-9
Chasis Boe-Bot y tarjetas

Board of Education (izquierda); HomeWork Board (derecha)

La Figura 3-10 muestra los servo puertos reconectados tanto para el Board of Education (lado izquierdo) y el HomeWork Board (lado derecho).

- ✓ Reconecte los servos a los servo cabezales.
- ✓ Asegúrese de conectar el plug etiquetado ‘L’ al puerto P13 y el plug etiquetado ‘R’ al puerto P12.

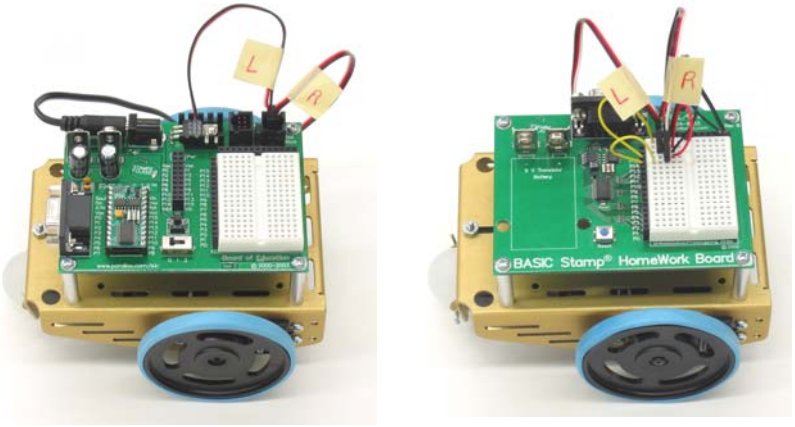


En el Board of Education *En el HomeWork Board*

Figura 3-10 servo puertos reconectados

La Figura 3-11 muestra el Chasis del Boe-Bot con sus respectivas tarjetas unidas.

- ✓ Coloque la tarjeta en las cuatro extensiones para que se alineen con los 4 barrenos de las 4 esquinas de la tarjeta.
- ✓ La tableta debe quedar cerca de las ruedas de tracción, no a la rueda posterior.
- ✓ Una la tarjeta a las extensiones con los tornillos de cabeza de gota.

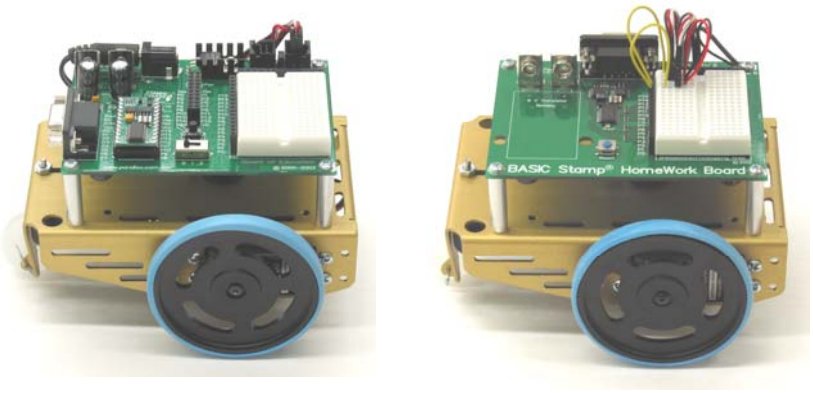


Con el Board of Education *Con el HomeWork Board*

Figura 3-11 tarjetas unidas al chasis del Boe-Bot

La Figura 3-12 muestra los robots Boe-Bot ensamblados, el izquierdo construido con un Board of Education (Serial Rev C) y el derecho construido con un HomeWork Board.

- ✓ Desde la parte inferior del chasis, jale cualquier exceso de cable de batería o de servos a través del barreno cubierto con la goma.
- ✓ Pliegue la longitud del cable excedente entre los servos y el chasis.



Con el Board of Education *Con el HomeWork Board*

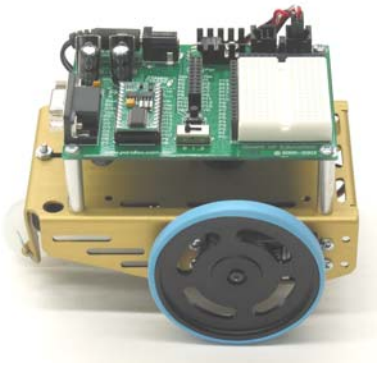
Figura 3-12
Robots Boe-Bot ensamblados

ACTIVIDAD #2: VUELVA A PROBAR LOS SERVOS

En esta Actividad, probará para estar seguro que las conexiones eléctricas entre su tarjeta y los servos son correctas. La Figura 3-13 indica las partes frontal, trasera, izquierda y derecha de su Boe-Bot. Debemos estar seguros de que el servo a la derecha gira cuando recibe pulsos de P12 y que el servo a la izquierda gira cuando recibe pulsos de P13.

Left

Back



Front

Right

Figura 3-13
lados frontal, trasero, izquierdo y derecho de su robot Boe-Bot

Probando la rueda derecha

El siguiente programa ejemplo probará el servo conectado a la rueda derecha, mostrado en la Figura 3-14. El programa para que esta rueda gire a la derecha por 3 segundos, luego se detendrá por 1 segundo, y luego girará a la izquierda por 3 segundos..

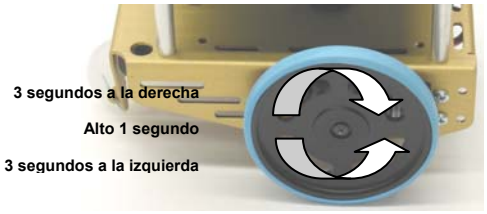


Figura 3-14
probando la rueda derecha

Programa Ejemplo: RightServoTest.bs2

- ✓ Apoye el Boe-Bot sobre su parte frontal para que las ruedas de tracción no se apoyen al piso o superficie de trabajo.
- ✓ Coloque las baterías en el paquete de baterías.
- ✓ Si tiene un Board of Education, coloque el switch en la posición 2.
- ✓ Si tiene un BASIC Stamp HomeWork Board, conecte la bateria de 9 V a su clip.
- ✓ Introduzca, salve y corra RightServoTest.bs2.
- ✓ Verifique que la rueda derecha gira a la derecha por 3 segundos, se detiene 1 y luego gira a la izquierda por 3.
- ✓ Si la rueda/servo derechos no se comportan como se esperaba, véa la sección de Solución de Problemas de Servos. Esta justo después de RightServoTest.bs2.
- ✓ Si la rueda/servo derechos se comporta como se esperaba, entonces continue con la sección Su Turno, donde probará la rueda izquierda.

```
' Robotica con el Boe-Bot - RightServoTest.bs2
' El servo derecho gira a la derecha 3 segundos, se detiene 1, luego gira
' a la izquierda 3 segundos.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter        VAR     Word

FOR counter = 1 TO 122                      ' Derecha justo abajo de 3 segs.
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

```
FOR counter = 1 TO 40                          ' Para un segundo.
  PULSOUT 12, 750
  PAUSE 20
NEXT

FOR counter = 1 TO 122                         ' A la izquierda 3 segundos.
  PULSOUT 12, 850
  PAUSE 20
NEXT

END
```

Su Turno – Probando la rueda izquierda

Ahora es momento de correr la misma prueba en la rueda izquierda como se muestra en la Figura 3-15. Esto implica modificar RightServoTest.bs2 para que los commandos **PULSOUT** sean enviados al servo conectado a P13 en vez de al servo conectado a P12.

Todo lo que tiene que hacer es cambiar los 3 comandos **PULSOUT** para que se lean **PULSOUT 13** en vez de **PULSOUT 12**.

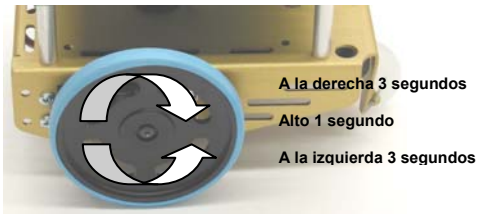


Figura 3-15
Probando la rueda izquierda

- ✓ Salve RightServoTest.bs2 como LeftServoTest.bs2.
- ✓ Cambie los 3 comandos **PULSOUT** para que se lean **PULSOUT 13** en vez de **PULSOUT 12**.
- ✓ Salve y corra el programa.
- ✓ Verifique que hace que el servo izquierdo gire a la derecha por 3 segundos, pare por 1y luego hace que gire a la izquierda por 3 segundos.
- ✓ Si la rueda/servo izquierdos no se comportan como se esperaba, véa la caja de Solución de Problemas de los Servos a continuación.
- ✓ Si la rueda/servo izquierdos se comportan como se esperaba, entonces su Boe-Bot funciona adecuadamente y está listo para continuar con la siguiente Actividad.

Solución de Problemas de los Servos : he aquí una lista de los síntomas más comunes.

El servo simplemente no gira.

- ✓ Si esta usando un Board of Education, asegúrese de que el switch de 3 posiciones esta colocado en la posición 2. Luego puede volver a correr programa presionando y liberando el botón Reset.
- ✓ Si esta usando un BASIC Stamp HomeWork Board, asegúrese de que el Paquete de baterías tiene baterías nuevas, todas orientadas adecuadamente.
- ✓ Vuelva a revisar las conexiones de su servo use la Figura 3-10 en la página 81 como guía. Si esta usando un HomeWork Board, quizá quiera volver a ver la Figura 2-18 en la página 47.
- ✓ Revise y asegúrese de haber cargado al programa correctamente.

El servo derecho no gira, pero el izquierdo si.

Los servos están intercambiados. El servo que esta conectado a P12 debería estar conectado a P13, y el servo que esta conectado a P13 debería estar conectado a P12.

- ✓ Desconecte la energía.
- ✓ Desconecte ambos servos.
- ✓ Conecte el servo que fue conectado a P12 a P13.
- ✓ Conecte el otro servo (que fue conectado a P13) a P12.
- ✓ Reconecte la energía.
- ✓ Vuelva a correr RightServoTest.bs2.

La rueda no se detiene completamente; gira lentamente.

El servo pudiera no estar exactamente centrado. Hay dos formas de arreglarlo:

- ✓ Ajuste en hardware: regrese y repita la Actividad #4 Capítulo 2: Centrando los Servos en la página 49. Si los servos no están montados para dar fácil acceso a los puertos de los potenciómetros, considere reorientarlos para re-ensamblar.
- ✓ Ajuste en software: si la rueda gira lentamente a la izquierda, use un valor ligeramente más pequeño que 750. Si está girando a la derecha, use un valor ligeramente mayor que 750. Este nuevo valor será usado en lugar de 750 para todos los comandos **PULSOUT** para esa rueda en los experimentos en este libro.

La rueda no se detiene por un segundo entre las votaciones derecha e izquierda.

La rueda puede girar rápidamente por 3 segundos en una dirección y 4 en la otra. También puede girar rápidamente por 3 segundos, luego ligeramente más lento por 1 segundo, y luego rápidamente de nuevo por 3 segundos. O, puede girar rápidamente en la misma dirección por 7 segundos. No importa, todo esto significa que el potenciómetro está fuera de ajuste.

- ✓ Remuevan las ruedas, desmonte los servos y repita el ejercicio en la Actividad #4: Centrando los Servos en la página 49.

ACTIVIDAD #3: PROGRAMA Y CIRCUITO INDICADOR INICIO/RESET

Cuando la fuente de voltaje cae por debajo del nivel que un dispositivo necesita para funcionar adecuadamente, se conoce como decaimiento. El BASIC Stamp se protege a si mismo del decaimiento haciendo que sus integrados de procesador y memoria de programa duerman hasta que la fuente regrese a niveles normales. Una caída por debajo de 5.2 V en Vin resulta en una caída por debajo de 4.3 V en la salida del regulador de voltaje interno del BASIC Stamp. Un circuito llamado detector de decaimiento en el BASIC Stamp está siempre atento a esto. Cuando ocurre el decaimiento, el detector de decaimiento deshabilita el procesador del BASIC Stamp y la memoria del programa.

Cuando la fuente de voltaje regresa por arriba de 5.2 V, el BASIC Stamp empieza a correr otra vez, pero no en el mismo lugar en el programa. Empieza desde el principio del programa. Esto es exactamente lo mismo que pasa cuando desconecta la energía y la vuelva conectar, y es lo mismo que pasa si presiona y suelta el botón Reset en su tarjeta.

Cuando las baterías de su Boe-Bot corren con bajo voltaje, los requerimientos pueden causar que el programa reinicie cuando no lo espera. Esto puede llevarlo a verdaderas confusiones del comportamiento del Boe-Bot. En algunos casos, el Boe-Bot estará corriendo el curso que se le haya programado a navegar y súbitamente, pareciera perderse el ir en una dirección inesperada. Si las baterías bajas son la causa, pudiera ser el hecho de que el programa del Boe-Bot regresará al inicio y empezar a otra vez. En otros casos, el Boe-Bot puede terminar haciendo una danza confusa porque cada vez que los servos empiezan a girar, sobrecargan las ya de por sí bajas baterías. El programa intenta hacer que los servos giren por una fracción de segundo, entonces reinicia una y otra vez.

Todo esto hace que un indicador de inicio/Reset sea un elemento diagnóstico extremadamente útil así como una herramienta útil al robot. Una forma de indicar resets es incluir una señal inconfundible al principio de todos los programas. La señal ocurre cada vez que la energía se conecta, pero también ocurre cada vez que ocurre un Reset debido a condiciones de decaimiento. Una señal efectiva para Reset es un parlante que emite un tono cada vez que el programa BASIC Stamp corre desde el inicio o se resetea.

Instrucciones Especiales del BASIC Stamp HomeWork Board

Aún cuando el indicador de reset le dirá cuando la batería de 9 V que suple al BASIC Stamp está corriendo en bajo voltaje, no le dirá cuando la fuente de los servos (el paquete de baterías) está corriendo en bajo voltaje. Siempre podrá distinguir esto porque los servos gradualmente se moverán más lento durante operación normal. Cuando observe este síntoma, reemplace las baterías muertas con baterías nuevas o recién cargadas.

Este ejercicio le presentará un dispositivo llamado parlante piezoeléctrico (piezoparlante) que podrá usar para generar tonos. Este parlante puede hacer diferentes tonos dependiendo de la frecuencia de las señales en nivel alto/bajo que reciba del BASIC Stamp. Su símbolo esquemático y dibujo de parte se muestran en la Figura 3-16. Este parlante será usado para emitir tonos cuando el BASIC Stamp sea reseteado en esta actividad así como en el resto de las actividades en este texto.

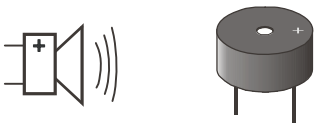


Figura 3-16
Piezoparlante

¿Qué es frecuencia? Es la medición de qué tan frecuentemente ocurre algo en un tiempo determinado.

¿Qué es un elemento piezoeléctrico y cómo puede hacer ruido? Es un cristal que cambie de forma ligeramente cuando se le aplica voltaje. Al aplicar rápidamente voltajes altos y bajos a un cristal piezoeléctrico, causa que el cristal cambie de forma rápidamente. El resultado es vibración. Los objetos que vibra causan que el aire a su alrededor también vibre. Esto es lo que nuestro oído detecta como sonidos y tonos. Por ejemplo, si rasga una sola cuerda de una guitarra, vibrará a una frecuencia y escuchará un tono particular. Si rasga una cuerda diferente vibrará a una frecuencia diferente y hará un tono diferente.

Los elementos Piezoeléctricos tienen muchos usos. Por ejemplo, se aplica fuerza a un elemento piezoeléctrico, puede crear voltaje. Algunos elementos piezoeléctricos tienen una frecuencia a la que naturalmente vibran. Esto puede usarse para crear voltajes a frecuencias que funcionen como el reloj oscilador para muchas computadoras y microcontroladores.

Partes requeridas

(1) Boe-Bot ensamblado y probado
(1) Piezoparlante
(Diversos) cables conectores

Si su piezoparlante tiene una etiqueta que diga "Remove seal after washing" quítela y proceda. ¡Su piezoparlante no debe ser lavado!

Construyendo el circuito indicador de inicio/Reset

La Figura 3-17 muestra un esquemático para un circuito de alarma con piezoparlante tanto para el Board of Education y para el BASIC Stamp HomeWork Board. La Figura 3-18 muestra un diagrama de conexiones para cada tarjeta.

¡Siempre desconecte la energía antes de construir o modificar circuitos!

- ✓ Si tiene un Board of Education, coloque el switch de 3 posiciones en la posición 0.
- ✓ Si tiene un BASIC Stamp HomeWork Board, desconecte la batería de 9 V de su clip y remueva una batería del paquete de baterías.

✓ Construya el circuito mostrado en la Figura 3-17 y la Figura 3-18.

Los circuitos de piezoparlante y servo por el resto de las actividades en este texto.

Todos los esquemáticos de circuito a partir de este punto mostrarán circuitos que deben ser agregados a los circuitos existentes servo y piezoparlante.

Todos los diagramas de conexiones mostrarán el circuito del esquemático que viene justo antes de el junto con las conexiones de circuito servo y piezoparlante.

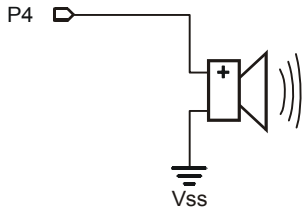


Figura 3-17
esquemático del circuito indicador Inicio/Reset del programa

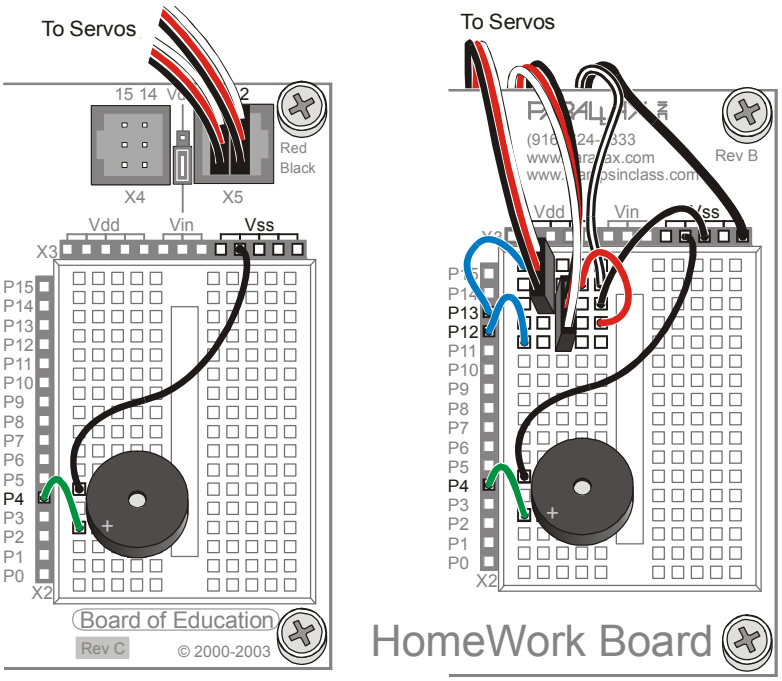


Figura 3-18
diagrama de conexiones para el circuito indicador de inicio/Reset del programa

Board of Education (izquierda) y HomeWork Board (derecha)

Programando el indicador inicio/Reset

El siguiente programa ejemplo prueba el piezoparlante. Usa el comando **FREQOUT** para enviar señales precisas en tiempo de nivel alto/bajo al parlante. La sintaxis **FREQOUT** es:

FREQOUT *Pin*, *Duration*, *Freq1* {,*Freq2*}

He aquí un ejemplo de un comando **FREQOUT** que es usado en el programa ejemplo.

```
FREQOUT 4, 2000, 3000
```

El argumento ***Pin*** es 4, significa que las señales en alto/bajo serán enviadas al pin I/O P4. El argumento ***Duration***, que es qué tanto durarán las señales en alto/bajo, es 2000, que es 2000 ms o 2 s. El argument ***Freq1*** es la frecuencia de las señales de nivel alto/bajo. En este ejemplo, las señales de nivel alto/bajo harán un tono de 3000 hertz, o 3 kHz.

La Frecuencia puede ser medida en hertz (Hz). Los hertz es una medición de frecuencia de cuántas veces por Segundo algo pasa. Un Hertz es símplemente una vez por Segundo, y se abrevia 1 Hz. Un kilohertz es mil veces por segundo y se abrevia 1 kHz.

FREQOUT sintetiza tonos digintalmente. El comando **FREQOUT** aplica pulsos alto/bajo de varias duraciones que hace que la vibración del piezoparlante se semeje más cercanamente a las vibraciones naturales de las cuerdas musicales.

Programa Ejemplo: StartResetIndicator.bs2

Este programa ejemplo hace un "beep" al principio del programa, luego enciende para corer un programa que envía mensajes **DEBUG** cada medio Segundo. Estos mensajes continuarán indefinidamente porque están anidados entre **DO** y **LOOP**. Si se interrumpe la energía del BASIC Stamp cuando está a la mitad del **DO…LOOP**, el programa comenzará al principio nuevamente. Cuando enpieza de nuevo hará "beep" nuevamente. Puede similar una condición de decaimiento ya sea presionando y liberando el botón Reset en su tarjeta o desconectando y reconectando su tarjeta nuevamente.

¡Aprenda a crear efectos de sonido y música con su BASIC Stamp! Baje ¿Qué es un *Microcontrolador?* de www.parallax.com/go/WAM, y pruebe el circuito ejemplo y los programas en el Capítulo 8.

Para aprender aún más acerca de las señales que **FREQOUT** genera, baje *Entendiendo Señales con el PropScope* de www.parallax.com/go/PropScope, y lea el Capítulo 7.

- ✓ Reconecte la energía a su tarjeta.
- ✓ Introduzca, salve y corra StartResetIndicator.bs2.
- ✓ Verifique que el piezoparlante hizo un tono audible claro por 2 segundos antes del mensaje "Waiting for reset…" empiece a desplegarse en la Terminal de Depuración.
- ✓ Si no escuchó un tono, busque errores en su cableado y su código. Repita hasta que obtenga un tono audible de su parlante.
- ✓ Si escuchó un tono audible, intente simular la condición de decaimiento presionando y liberando el botón Reset en su tarjeta. Verifique que el piezoparlante hace un tono audible claro después de cada reset.
- ✓ También intente conectando y desconectando su batería y verifique que esto también entrega el tono de advertencia en el caso de reset.

```
' Robotica con el Boe-Bot - StartResetIndicator.bs2
' Prueba el circuito del piezoparlante.
' {$STAMP BS2}                          ' Directiva Stamp.
' {$PBASIC 2.5}                         ' Directiva PBASIC.
```

```
DEBUG CLS, "Beep!!!"                    ' Despliega mientras suena el parlante
FREQOUT 4, 2000, 3000                   ' Señal de programa en inicio/reset.

DO                                      ' DO...LOOP
  DEBUG CR, "Waiting for reset…"        ' Despliega mensaje
  PAUSE 500                             ' cada 0.5 segundos
LOOP                                    ' hasta un reset de hardware.
```


Cómo trabaja StartResetIndicator.bs2

StartResetIndicator.bs2 empieza desplegando en mensaje "Beep!!!". Luego, inmediatamente después de imprimir el mensaje, el commando **FREQOUT** toca un tono de 3 kHz en el parlante piezoeléctrico por 2 s. Debido a que las instrucciones son ejecutadas muy rápidamente por el BASIC Stamp, debe parecer que el mensaje es desplegado al mismo tiempo que el piezoparlante empieza a tocar el tono.

Cuando termina el tono, el programa entra en un ciclo **DO…LOOP**, desplegando el mismo mensaje "Waiting for reset…" una y otra vez. Cada vez que el botón reset en el Board of Education es presionado o la energía es desconectada y reconectada, el programa empieza de nuevo, con el mensaje "Beep!!!" y el tono de 3 kHz.

Su Turno – Agregando StartResetIndicator.bs2 a un Programa Diferente

El comando **FREQOUT** en el programa indicador de batería sera usado al inicio de cada programa ejemplo de aquí en adelante. Puede considerarlo parte de la "rutina de inicialización" o "rutina de carga" para cada programa Boe-Bot.

Una rutina de inicialización esta conformada por todos los commandos necesarios para que un dispositivo o programa esté a punto y corra. Frecuentemente incluye la parametrización de ciertos valores variables, sonidos indicadores y, para dispositivos más complejos, autopruebas y calibración.

- ✓ Abra HelloOnceEverySecond.bs2.
- ✓ Copie el comando **FREQOUT** del programa StartResetIndicator.bs2 en el programa HelloOnceEverySecond.bs2, arriba de la sección **DO…LOOP**.
- ✓ Corra el programa modificado y verifique que responde con un tono de aviso cada vez que al BASIC Stamp se le aplica reset (ya sea presionando o liberando el botón Reset en la tarjeta o desconectando y reconectando la batería).

ACTIVIDAD #4: PROBANDO EL CONTROL DE VELOCIDAD CON LA TERMINAL DE DEPURACIÓN

En esta actividad, graficará la velocidad del servo vs. el ancho de pulso. Una cosa que puede hacer a este proceso ir mucho más rápido es la ventana de Transmisión de la Terminal de depuración (Figura 3-19). Puede usarla para enviar los mensajes al BASIC Stamp. Al enviar mensajes que indiquen al BASIC Stamp qué ancho de pulso entregar al servo, puede probar la velocidad del servo a diversos anchos de pulso.

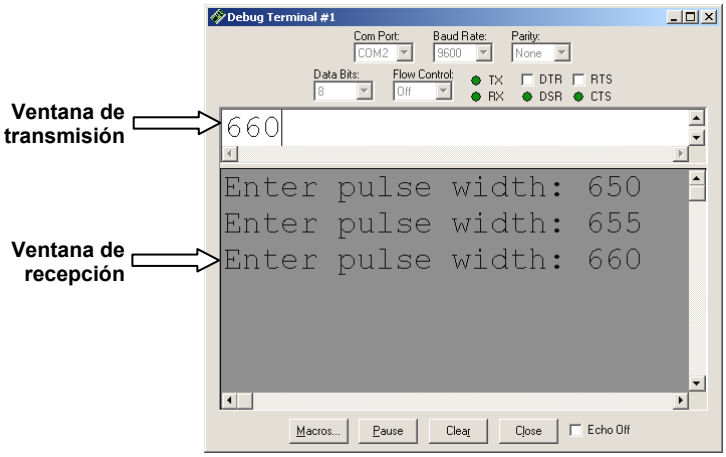


Figura 3-19
Ventanas de Recepción y Transmisión de la Terminal de Depuración

El ancho de pulso es una forma común de describir cuánto dura un pulso. La razón de que se le llame "ancho" de pulso es porque el tiempo que dura un pulso esta relacionada a qué tan ancho es en un diagrama de tiempo. Los pulsos que duran más son más anchos en los diagramas de tiempo y viceversa.

Usando el Comando DEBUGIN

Hasta ahora, probablemente ya esté familiarizado con el comando **DEBUG** y cómo puede ser usado para enviar mensajes desde el BASIC Stamp a la Terminal de Depuración. El lugar donde son vistos los mensajes es llamado ventana de recepción porque es el lugar donde son desplegados los mensajes recibidos desde el BASIC Stamp. La Terminal de Depuración también tiene una ventana de Transmisión, que le permite enviar información al BASIC Stamp mientras que un programa corre. Puede usar el comando **DEBUGIN** para hacer que el BASIC Stamp reciba lo que usted escriba en la ventana de Transmisión y lo guarde en una o más variables.

El comando **DEBUGIN** coloca el valor que escribe en la ventana de Transmisión en una variable. En el siguiente programa ejemplo, una variable tipo word llamada **pulseWidth** será usada para guardar los valores que el comando **DEBUGIN** reciba.

```
pulseWidth    VAR    Word
```

El comando **DEBUGIN** puede ser usado para capturar un valor decimal que introduzca en la ventana de Transmisión de la Terminal de Depuración y guardarlo en **pulseWidth**:

```
DEBUGIN DEC pulseWidth
```

Entonces puede programar el BASIC Stamp para usar este valor. Aquí es usado en el argumento ***Duration*** del comando **PULSOUT**:

```
PULSOUT 12, pulseWidth
```

Programa Ejemplo: TestServoSpeed.bs2

Este programa le permite establecer el argumento ***Duration*** del commando **PULSOUT** introduciéndolo en la ventana de transmission de la Terminal de Depuración.

- ✓ Continúe esta actividad con el Boe-Bot apoyado en su parte frontal para que las ruedas no se apoyen en el piso.
- ✓ Introduzca, salve y corra TestServoSpeed.bs2.
- ✓ Apunte a la ventana de transmision de la Terminal de Depuración con su ratón y haga click para activar el cursor en esa ventana para escribir.
- ✓ Escriba 650 y presione la tecla Enter.
- ✓ Verifique que el servo gira a velocidad plena a la derecha por 6 segundos.

Cuando el servo termine de girar, se le pedirá que introduzca otro valor.

- ✓ Escriba 850 y presione la tecla Enter.
- ✓ Verifique que el servo gire a velocidad plena a la izquierda.

Intente medir la velocidad rotacional de la rueda en RPM (revoluciones por minuto) para un rango de anchos de pulso entre 650 y 850. He aquí como:

- ✓ Haga una marca en la rueda para que pueda ver cuánto gira en 6 segundos.

- ✓ Use la Terminal de Depuración para probar cuánto gira la rieda con cada uno de los siguientes anchos de pulso: 650, 660, 670, 680, 690, 700, 700, 710, 720, 730, 740, 750, 760, 770, 780, 790, 800, 810, 820, 830, 840, 850
- ✓ Para cada ancho de pulso, multiplique el número de vueltas por 10 para obtener las RPM. Si la rueda hace 3.65 vueltas completas estaba girando a 36.5 RPM.
- ✓ Explique en sus propias palabras como puede usar el ancho de pulso para controlar la velocidad del servo de rotación continua.

```
' Robotica con el Boe-Bot - TestServoSpeed.bs2
' Introduce el ancho de pulso, luego cuenta las revoluciones de la rueda.
' La rueda girara por 6 segundos
' Multiplique por 10 para obtener las revoluciones por minuto (RPM).

'{$STAMP BS2}
'{$PBASIC 2.5}

counter        VAR     Word
pulseWidth     VAR     Word
pulseWidthComp VAR     Word

FREQOUT 4, 2000, 3000                    ' Señal de programa en inicio/reset.

DO

  DEBUG "Enter pulse width: "

  DEBUGIN DEC pulseWidth

  pulseWidthComp = 1500 - pulseWidth

  FOR counter = 1 TO 244
    PULSOUT 12, pulseWidth
    PULSOUT 13, pulseWidthComp
    PAUSE 20
  NEXT

LOOP
```

Cómo Trabaja TestServoSpeed.bs2

Tres variables son declaradas, **`counter`** para el ciclo **`FOR…NEXT`**, **`pulseWidth`** para los comandos **`DEBUGIN`** y **`PULSOUT`** y **`pulseWidthComp`** que guarda un valor usado en un segundo comando **`PULSOUT`**.

```
counter        VAR     Word
pulseWidth     VAR     Word
pulseWidthComp VAR     Word
```

El comando **FREQOUT** es usado para indicar que el programa ha comenzado.

```
FREQOUT 4,2000,3000
```

El resto del programa esta anidado en un ciclo **DO…LOOP**, por lo que se ejecutará una y otra vez. Al operador de la Terminal de Depuración (usted) se le pide que introduzca un ancho de pulso. El comando **DEBUGIN** guarda este valor en la variable **pulseWidth**.

```
DEBUG "Enter pulse width: "

DEBUGIN DEC pulseWidth
```

Para hacer la medición mas precisa, deben ser enviados dos comandos **PULSOUT**. Al hacer a uno de los comandos **PULSOUT** del mismo valor bajo 750 como al otro arriba de 750, la suma de los dos argumentos **PULSOUT** ***Duration*** siempre es 1500. Esto asegura que los dos comandos **PULSOUT** combinados toman la misma cantidad de tiempo. El resultado es que no importa la ***Duration*** de su commando **PULSOUT,** el ciclo **FOR…NEXT** tomará la misma cantidad de tiempo para ejecutarse. Esto hará más precisas las mediciones de RPM que tomará en la sección Su Turno.

El siguiente comando toma el ancho de pulso que introdujo y calcula un ancho de pulso que resulte en 1500 cuando los dos se suman juntos. Si introduce un ancho de pulso de 650, **pulseWidthComp** será 850. Si introduce un ancho de pulso de 850, **pulseWidthComp** será 650. Si introduce un ancho de pulso de 700, **pulseWidthComp** será 800. Intente otros ejemplos. Todos sumarán 1500.

```
pulseWidthComp = 1500 - pulseWidth
```

Un ciclo **FOR…NEXT** que corre por 6 segundos manda pulsos al servo de la derecha (P12). El valor de **pulseWidthComp** se envía al servo de la izquierda (P13), haciéndolo girar en dirección opuesta.

```
FOR counter = 1 TO 244
  PULSOUT 12, pulseWidth
  PULSOUT 13, pulseWidthComp
  PAUSE 20
NEXT
```

Su Turno – Tema avanzado: Grafica Ancho de Pulso vs. Velocidad Rotacional

La Figura 3-20 muestra un ejemplo de una curva de transferencia para un servo de rotación continua. El eje horizontal muestra el ancho de pulso en ms y el eje vertical

muestra la velocidad rotacional en RPM. En esta gráfica, giro a la derecha es negativo y a la izquierda es positiva. Esta curva de transferencia de servo en particular varía de cerca de -48 a 48 RPM sobre el rango de ancho de pulso de prueba que va de 1.3 a 1.7 ms.

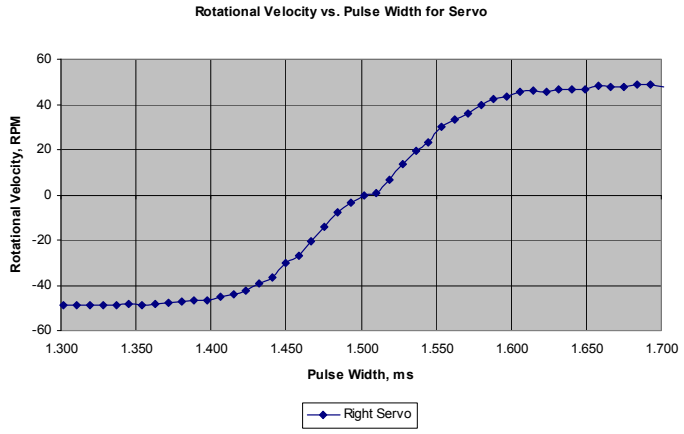


Figura 3-20
Ejemplo de Curva de Transferencia para un Servo de Rotación Continua Parallax

Recuerde que el argumento ***Duration*** del comando **`PULSOUT`** esta en unidades de 2 µs. **`PULSOUT 12, 650`** envía pulsos a P12que duran 1.3 ms**`. PULSOUT 12, 655`** envía pulsos de 1.31 ms, **`PULSOUT 12, 660`** envía pulsos de 1.32 ms, etc.

$$Duration = 650 \times 2\ \mu s = 650 \times 0.000002\ s = 0.0013\ s = 1.3\ ms$$

$$Duration = 655 \times 2\ \mu s = 655 \times 0.000002\ s = 0.00131\ s = 1.31\ ms$$

$$Duration = 660 \times 2\ \mu s = 660 \times 0.000002\ s = 0.00132\ s = 1.32\ ms$$

Puede usar la Tabla 3-1 para grabar los datos para su propia curva de transferencia. Mantenga en mente que el programa ejemplo esta controlando la rueda derecha con los valores que introduzca. La rueda izquierda gira en dirección opuesta.

- ✓ Marque su rueda dercha para que tenga un punto de referencia para contar las revoluciones.
- ✓ Corra TestServoSpeed.bs2.

Tabla 3-1: Ancho de Pulso y RPM para Servo de Rotación Continua Parallax							
Ancho de Pulso (ms)	**Velocidad Rotacional (RPM)**	**Ancho de Pulso (ms)**	**Velocidad Rotacional (RPM)**	**Ancho de Pulso (ms)**	**Velocidad Rotacional (RPM)**	**Ancho de Pulso (ms)**	**Velocidad Rotacional (RPM)**
1.300		1.400		1.500		1.600	
1.310		1.410		1.510		1.610	
1.320		1.420		1.520		1.620	
1.330		1.430		1.530		1.630	
1.340		1.440		1.540		1.640	
1.350		1.450		1.550		1.650	
1.360		1.460		1.560		1.660	
1.370		1.470		1.570		1.670	
1.380		1.480		1.580		1.680	
1.390		1.490		1.590		1.690	
						1.700	

- ✓ Haga Click en la ventana de Transmisión de la Terminal de Depuración.
- ✓ Introduzca el valor 650.
- ✓ Cuente cuántas vueltas hace la rueda.

Puesto que el servo gira por 6 segundos, puede multiplicar este valor por 10 para obtener revoluciones por minuto (RPM).

- ✓ Multiplique este valor por 10 e introduzca el resultado cerca al dato 1.3 ms.
- ✓ Introduzca el valor 655, y cuente cuantas vueltas dio la rueda.
- ✓ Multiplique este valor por 10 e introduzca el resultado cerca al dato 1.31 ms.
- ✓ Mantenga incrementando las duraciones por 5 (0.01 ms) hasta que llegue a 850 (1.7 ms).
- ✓ Use una hoja de cálculo, calculadora o papel graficador para graficar los datos.
- ✓ Repita este proceso para su otro servo.

Para repetir estas mediciones para la rueda izquierda, modifique el comando **`PULSOUT`** para que los pulsos con una ***Duration*** de **`pulseWidth`** sean enviados a P13 y los pulsos con una ***Duration*** de **`pulseWidthComp`** sean enviados a P12.

RESUMEN

Este Capítulo cubrió el ensambe y prueba del Boe-Bot. Esto involucró un ensamble mecánico, como conectar las diversas partes móviles al chasis del Boe-Bot. También involucró el ensamblado del circuito, conectando los servos y el piezoparlante. La prueba involucró volver a probar los servos después de que fueron desconectados para construir el Boe-Bot.

El concepto de decaimiento fue presentado junto con lo que hace esta condición a un programa ejecutándose en el BASIC Stamp. El decaimiento causa que él BASIC Stamp se apague, y luego reinicien programa desde el principio. Un piezoparlante fue agregado para indicar el inicio de un programa. Si el piezoparlante suena a la mitad de un programa corriendo cuando no se supone que lo haga, esto puede indicar una condición de decaimiento las condiciones de decaimiento pueden indicar baterías bajas. Para hacer que el piezoparlante ejecute un tono para indicar un Reset, se presentó el comando **`FREQOUT`**. Este comando es parte de una rutina de inicialización que será usada al principio de todos los programas del Boe-Bot.

Hasta este Capítulo, la Terminal de Depuración ha sido usada para desplegar mensajes enviados a la computadora por el BASIC Stamp. Estos mensajes fueron desplegados en la ventana de recepción. La Terminal de Depuración también tiene una ventana de transmisión que puede usar para enviar valores al BASIC Stamp. El BASIC Stamp puede capturar estos valores ejecutando el comando **`DEBUGIN`**, el cual recibe un valor enviado por la ventana de transmisión de la terminal de depuración y lo guarda en una variable. Entonces el valor puede ser usado por el programa PBASIC. Esta técnica fue usada para establecer el ancho de pulso para controlar y probar la dirección y velocidad del servo. También se usó para colectar datos para graficar la curva de transferencia de un servo de rotación continúa Parallax.

Preguntas

1. ¿cuales son algunos de los síntomas del decaimiento en el Boe-Bot?
2. ¿Como puede usarse un piezoparlante para detectar decaimiento?
3. ¿Que es un reset?
4. ¿Que es una rutina de inicialización?
5. ¿Cuales son tres (o más) posibles errores que pueden ocurrir cuando se conectan y re-conectan los servos?
6. ¿Qué comando tiene que cambiar en RightServoTest.bs2 para probar la rueda izquierda en vez de la rueda derecha?

Ejercicios

1. Escriba un comando **`FREQOUT`** eran un tono que suene diferente del tono de detección de Reset para indicar el fin de un programa.
2. Escriba un comando **`FREQOUT`** que haga un tono (diferente al de inicio o fin) que indique que un paso intermedio en un programa ha sido completado. Intente un valor con una duración de 100 ms a una frecuencia de 4 kHz.

Proyectos

1. Modifique RightServoTest.bs2 para que haga un tono que indique que la prueba está completa.
2. Modifique TestServoSpeed.bs2 para que pueda usar **`DEBUGIN`** para introducir el ancho de pulso para el servo izquierdo y derecho así como un número de pulsos a entregar en un ciclo **`FOR…NEXT`**. Use este programa para controlar el movimiento de su Boe-Bot's a través de la ventana de transmisión de la terminal de depuración.

Soluciones

Q1. los síntomas incluyen comportamiento errático confundir en direcciones inesperadas o hacer una danza confusa.

Q2. Un comando **FREQOUT** al principio de todos los programas Boe-Bot causa que el piezoparlante ejecute un tono. Este tono ocurrirá entonces cada vez que un Reset accidental ocurra debido a condiciones de decaimiento.

Q3. Un reset es cuando la energía es interrumpida y el programa BASIC Stamp empieza a correr nuevamente desde el principio del programa.

Q4. Una rutina de inicialización consiste de líneas de código que son usadas al principio del programa. Estas líneas de código se ejecutan cada vez que programa inicia desde el principio.

Q5. 1) Las líneas entre P12 y P13 son intercambiadas. 2) uno o ambos servos están conectados al revés, y entonces el código de color blanco-rojo-negro es incorrecto. 3) el switch de energía no está en posición-2. 4) La batería de 9V o las AA no están instaladas. 5) el potenciómetro de centrado del servo está fuera de ajuste.

Q6. Los comandos **PULSOUT** deben ser cambiados para que digan **PULSOUT 13** en vez de **PULSOUT 12**.

E1. La clave es modificar el comando **FREQOUT** usado por el programa StartResetIndicator.bs2, esto es, **FREQOUT, 4, 2000, 3000**. Por ejemplo: **FREQOUT, 4, 500, 3500** trabajaría.

E2. **FREQOUT 4, 100, 4000**.

P1. La clave para resolver este programa es agregar la línea del ejercicio 1 arriba del comando **END** en el programa RightServoTest.bs2.

```
' Robotica con el Boe-Bot - Ch03Prj01_TestCompleteTone.bs2
' El Servo derecho gira 3 segundos a la derecha, alto 1 segundo, luego
' gira a la izquierda 3 segundos. Un tono indica que la prueba esta
' completa.

' {$STAMP BS2}
' {$PBASIC 2.5}
DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000        ' señal para inicio de programa.

FOR counter = 1 TO 122       ' a la derecha justo abajo de 3 segundos.
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

```
FOR counter = 1 TO 40             ' alto un segundo.
  PULSOUT 12, 750
  PAUSE 20
NEXT

FOR counter = 1 TO 122            ' a la izquierda 3 segundos.
  PULSOUT 12, 850
  PAUSE 20
NEXT

FREQOUT 4, 500, 3500              ' señal para fin de programa

END
```

P2. Para resolver este problema, TestServoSpeed.bs2 debe ser expandido para recibir 3 datos: el ancho de pulso del servo izquierdo, ancho de pulso del servo derecho y número de pulsos. Luego, un ciclo **FOR…NEXT** con 2 comandos **PULSOUT** deben ser agregados para de hecho mover los servo motores. Más aún, todas las variables deben ser declaradas al principio del programa. Un ejemplo de solución se muestra a continuación. Nota: este proyecto se prueba mejor con las ruedas del Boe-Bot levantadas.

```
' Robotica con el Boe-Bot - Ch03Prj02_DebuginMotion.bs2
' Entra ancho de pulso y duracion de 2 ruedas en terminal de depuracion
'{$STAMP BS2}
'{$PBASIC 2.5}

ltPulseWidth   VAR     Word            ' ancho de pulso servo izquierdo
rtPulseWidth   VAR     Word            ' ancho de pulso servo derecho
pulseCount     VAR     Byte            ' Numero de pulsos al servo
counter        VAR     Word            ' contador de ciclo

DO
  DEBUG "Enter left servo pulse width: "  ' introduce valores en
  DEBUGIN DEC ltPulseWidth                ' la Terminal de Depuracion

  DEBUG "Enter right servo pulse width: "
  DEBUGIN DEC rtPulseWidth

  DEBUG "Enter number of pulses:        "
  DEBUGIN DEC pulseCount

  FOR counter = 1 TO pulseCount     ' Envia numero especifico de pulsos
    PULSOUT 13, ltPulseWidth        ' Left servo motion
    PULSOUT 12, rtPulseWidth        ' Right servo motion
    PAUSE 20
  NEXT

LOOP
```

Capítulo 4 : Navegación del Boe-Bot

El Boe-Bot puede ser programado para ejecutar una variedad de maniobras. Las maniobras y técnicas de programación presentadas en este capítulo serán reutilizadas en capítulos posteriores. La única diferencia es que en este capítulo, el Boe-Bot ejecutará ciegamente las maniobras. En capítulos posteriores, el Boe-Bot ejecutar maniobras similares en respuesta a condiciones que detecte con sus sensores.

Este Capítulo también presenta maneras de ajustar y calibrar la navegación del Boe-Bot. Se incluyen técnicas para enderezar el manejo hacia el frente del Boe-Bot's, dar vueltas mas precisas y calcular distancias.

Actividad	Resumen
1	Programa el Boe-Bot para ejecutar maniobras básicas: adelante, atrás, gira a la izquierda, gira a la derecha giros sobre un eje.
2	Afinar las maniobras de la Actividad #1 para que sean mas precisas.
3	Calcular matemáticamente el número de pulsos a entregar para hacer que el Boe-Bot viaje una distancia predeterminada.
4	En vez de programar al Boe-Bot para hacer paros y arranques abruptos, escribir programas que hagan que el Boe-Bot acelere gradualmente y desacelere después una maniobra.
5	Escribir subrutinas para ejecutar maniobras básicas y que cada subrutina pueda ser usada una y otra vez en un programa.
6	Registrar maniobras complejas en el módulo de memoria sin utilizar del BASIC Stamp y escribir programas que vuelvan a ejecutar estas maniobras.

ACTIVIDAD #1: MANIOBRAS BÁSICAS DEL BOE-BOT

La Figura 4-1 muestra el frente, parte trasera, izquierda y derecha de su Boe-Bot. Cuando el Boe-Bot vaya al frente, en la imagen, tendrá que desplazarse hacia el extremo derecho de la hoja. Hacia atrás será hacia el extremo izquierdo de la página. Una vuelta izquierda hará que el Boe-Bot esté listo a dirigirse hacia la parte superior de la página y una vuelta a la derecha hará que esté de frente a la parte baja de la página.

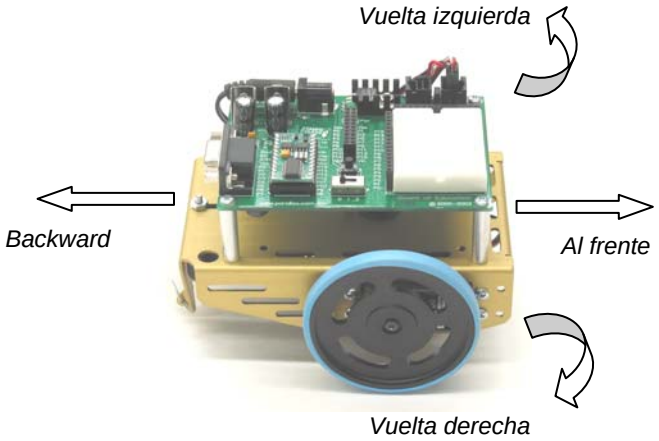


Figura 4-1
Your Boe-Bot y Driving Directions

Moviéndose hacia el frente

un detalle curioso: para hacer que el Boe-Bot vaya al frente, la rueda izquierda del Boe-Bot tiene que girar hacia la izquierda, pero su rueda derecha tiene que girar a la derecha. Si aún no lo ha visualizado, vea la Figura 4-2 y véa si puede convencerse de que es cierto. Visto desde la izquierda, la rueda tiene que girar a la izquierda para que el Boe-Bot se mueva al frente. Visto desde la derecha, la otra rueda tiene que girar hacia la derecha para que el Boe-Bot se mueva al frente.

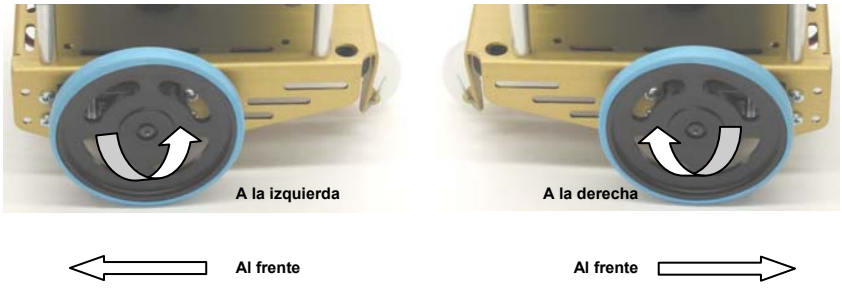


Figura 4-2
rotación de las ruedas para movimiento al frente

Recuerde del Capítulo 2 que el argumento ***Duration*** del comando **PULSOUT** controla la velocidad y dirección de los giros del servo. Los argumentos ***StartValue*** y ***EndValue*** de un ciclo **FOR…NEXT** controlan el número de pulsos que son entregados. Puesto que cada pulso

toma la misma cantidad de tiempo, el argumento ***EndValue*** también controla el tiempo que gira que el servo. He aquí un programa ejemplo que hará que el Boe-Bot vaya hacia el frente por alrededor de 3 segundos.

Programa Ejemplo: BoeBotForwardThreeSeconds.bs2

- ✓ Asegúrese de que la energía esta conectada al BASIC Stamp y a los servos.
- ✓ Introduzca, salve y corra BoeBotForwardThreeSeconds.bs2.


```
' Robotica con el Boe-Bot - BoeBotForwardThreeSeconds.bs2
' Hace que el Boe-Bot vaya al frente por 3 segundos.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                   ' señal de programa en inicio/reset.

FOR counter = 1 TO 122                  ' Corren servos por 3 segundos.

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

NEXT

END
```

Cómo trabaja BoeBotForwardThreeSeconds.bs2

Del Capítulo 2, adquirió experiencia con los elementos de este programa: la declaración de una variable, un ciclo **`FOR…NEXT`**, comandos **`PULSOUT`** con argumentos ***Pin*** y ***Duration*** y comandos **`PAUSE`**. He aquí una revisión de lo que cada uno hace y cómo se relaciona a los movimientos de los servos.

Primero una variable es declarada que será usada en el ciclo **`FOR...NEXT`**.

```
counter VAR Word
```

Debiera reconocer este comando; genera un tono para indicar el inicio del programa. Será usado en todos los programas que corran los servos.

```
FREQOUT 4, 2000, 3000      ' señal de programa en inicio/reset.
```

Este ciclo **FOR…NEXT** envía 122 juegos de pulsos a los servos, uno para cada uno de P13 y P12, pausa por 20 ms después de cada juego y luego regresa al inicio del ciclo.

```
FOR counter = 1 TO 122
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

PULSOUT 13, 850 causa que el servo izquierdo gire a la izquierda mientras que **PULSOUT 12, 650** causa que el servo derecho gira a la derecha. Por lo tanto, ambas ruedas estarán girando hacia el frente del Boe-Bot, causando que se mueva hacia el frente. Le toma cerca del 3 segundos al ciclo **FOR…NEXT** ejecutarse 122 veces, asi es que el Boe-Bot avanza por cerca de 3 segundos.

Su Turno – Ajustando Distancia y Velocidad

- ✓ al cambiar el argumento ***EndValue*** del ciclo **FOR…NEXT** de 122 a 61, puede hacer que el Boe-Bot se mueva al frente por la mitad del tiempo. Esto a su vez hara que el Boe-Bot se mueva al frente la mitad de distancia.
- ✓ Salve BoeBotForwardThreeSeconds.bs2 con un nuevo nombre.
- ✓ Cambie ***EndValue*** en el ciclo **FOR...NEXT** de 122 a 61.
- ✓ Corra programa y verifique que corrió la mitad del tiempo y cubrió la mitad de distancia.
- ✓ Intente estos pasos nuevamente, pero esta vez, cambie ***EndValue*** en el ciclo **FOR…NEXT** a 244.

Los argumentos **PULSOUT** ***Duration*** de 650 y 850 causaron que los servos girasen a cerca de su velocidad maxima. Al modificar cada argumento **PULSOUT** ***Duration*** a un valor cercano al valor de paro de 750, puede reducir la velocidad de su Boe-Bot.

- ✓ Modifique su programa con estos comandos **PULSOUT**:

```
PULSOUT 13, 780
PULSOUT 12, 720
```

- ✓ Corra el programa, y verifique que el Boe-Bot se mueven más lento.

Moviéndose Hacia Atrás, Girando y Giro Sobre un Eje

Todo lo que se necesita para obtener otros movimientos de su Boe-Bot son diferentes combinaciones de los argumentos **PULSOUT** ***Duration***. Por ejemplo, estos dos comandos **PULSOUT** pueden ser usados para hacer que su Boe-Bot vaya en reversa:

```
PULSOUT 13, 650
PULSOUT 12, 850
```

Estos dos comandos harán que su Boe-Bot gire en una vuelta izquierda (giro izquierdo al verlo desde arriba):

```
PULSOUT 13, 650
PULSOUT 12, 650
```

Estos dos comandos harán que su Boe-Bot gire en una vuelta derecha (giro derecho al verlo desde arriba):

```
PULSOUT 13, 850
PULSOUT 12, 850
```

Puede combinar todos estos comandos en un solo programa que haga que el Boe-Bot se mueva al frente, gire izquierda, gire derecha y luego vaya en reversa.

Programa Ejemplo: ForwardLeftRightBackward.bs2

- ✓ Introduzca, salve y corra ForwardLeftRightBackward.bs2.

TIP – Para introducir este programa rápidamente, use las herramientas del menú Edit del editor del BASIC Stamp (Copy y Paste) para hacer cuatro copias del ciclo **FOR…NEXT**. Luego, ajuste sólo los valores de **PULSOUT** ***Duration*** y los ***EndValues*** de los ciclos **FOR…NEXT**.

```
' Robotica con el Boe-Bot - ForwardLeftRightBackward.bs2
' Movimiento al frente, izquierda, derecha y atrás para prueba y ajuste.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                    ' Señal de programa en inicio/reset.
```

```
FOR counter = 1 TO 64                     ' Al frente

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

NEXT

PAUSE 200

FOR counter = 1 TO 24                     ' Giro izquierda - aprox 1/4 vuelta

  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20

NEXT

PAUSE 200

FOR counter = 1 TO 24                     ' Giro derecha - aprox 1/4 vuelta

  PULSOUT 13, 850
  PULSOUT 12, 850
  PAUSE 20

NEXT

PAUSE 200

FOR counter = 1 TO 64                       ' Hacia atras

  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20

NEXT

END
```

Su Turno – Giro sobre un eje

Puede hacer que el Boe-Bot gire sobre el eje de una rueda. El truco es mantener una rueda quieta y la otra girando. Por ejemplo, si mantiene la rueda izquierda quieta y la derecha la hace girar a la derecha (al frente), el Boe-Bot pivoteará a la izquierda.

```
PULSOUT 13, 750
PULSOUT 12, 650
```

Si quiere fin de al frente y a la derecha, simplemente detenga la rueda derecha y haga que la rueda izquierda gire en sentido izquierdo (al frente).

```
PULSOUT 13, 850
PULSOUT 12, 750
```

Estos son los commandos **PULSOUT** para pivotear hacia atrás y a la derecha.


```
PULSOUT 13, 650
PULSOUT 12, 750
```

Finalmente, estos son los commandos **PULSOUT** para pivotear hacia atrás y a la izquierda.

```
PULSOUT 13, 750
PULSOUT 12, 850
```

- ✓ Salve ForwardLeftRightBackward.bs2 como PivotTests.bs2.
- ✓ Substituya los comandos **PULSOUT** recién discutidos en lugar de las rutinas al frente, izquierda, derecha y hacia atrás.
- ✓ Ajuste el tiempo de operación de cada maniobra cambiando cada valor de ***EndValue*** en cada ciclo **FOR…NEXT** a 30.
- ✓ Asegúrese de cambiar el comentario de cada ciclo **FOR…NEXT** para reflejar cada nueva acción de pivoteo.
- ✓ Corra el programa modificado y verifique que trabajan los diferentes pivoteos.

ACTIVIDAD #2: AJUSTANDO LAS MANIOBRAS BÁSICAS

Imagine un programa que le indique al Boe-Bot desplazarse a velocidad plena hacia el frente por 15 segundos. ¿Qué pasará si él Boe-Bot curvea su dirección ligeramente a la izquierda o la derecha durante su viaje cuando se supone que vaya en línea recta? No es necesario volver a abrir el Boe-Bot para re-ajustar los servos con un desarmador. Simplemente puede ajustar programas ligeramente para lograr que ambas ruedas del Boe-Bot viajen a la misma velocidad. Mientras que la solución del desarmador será llamada "ajuste de hardware", la solución de programación es llamada un "ajuste de software".

Alineando el Camino del Boe-Bot

El primer paso es examinar el recorrido del Boe-Bot para averiguar si se curvea a la izquierda o la derecha cuando se supone que vaya en línea recta. 10 segundos de viaje al frente deberían ser suficientes. Esto puede lograrse con una modificación simple a BoeBotForwardThreeSeconds.bs2 de la Actividad previa.

Programa Ejemplo: BoeBotForwardTenSeconds.bs2

- ✓ Abra BoeBotForwardThreeSeconds.bs2.
- ✓ Renómbrelo y sálvelo como BoeBotForwardTenSeconds.bs2.
- ✓ Cambie **`EndValue`** del **`FOR counter`** de 122 a 407 como sigue:

```
' Robotica con el Boe-Bot - BoeBotForwardTenSeconds.bs2
' Hace que el Boe-Bot avance por 10 segundos.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000       ' Señal de programa en inicio/reset.

FOR counter = 1 TO 407      ' Numero de pulsos – tiempo en operacion.

  PULSOUT 13, 850           ' Servo izquierdo a velocidad plena, giro a la izq
  PULSOUT 12, 650           ' Servo derecho a velocidad plena, giro a la der.
  PAUSE 20

NEXT

END
```

- ✓ Corra el programa y observe de cerca par aver si su Boe-Bot se inclina a la derecha o izquierda al avanzar por 10 segundos.

Su Turno – Ajustando la Velocidad del Servo para alinear el rumbo del Boe-Bot

Si su Boe-Bot va en perfecta línea recta, intente este ejemplo de cualquier manera. Si sigue estas Instrucciones, ajustará su Boe-Bot para que curvee su ruta ligeramente a la derecha.

Digamos que el Boe-Bot se carga ligeramente a la izquierda. Hay 2 formas de pensar en este problema: o bien la rueda izquierda está girando muy lento o bien la rueda derecha está girando más rápidamente. Puesto que el Boe-Bot ya está a velocidad plena, acelerar la rueda izquierda no es práctico pero reducir la velocidad de la rueda dercha ayudará a remediar la situación.

Recuerde que la velocidad del servo es determinado por el argumento ***Duration*** del comando **`PULSOUT`**. Entre más cercano es ***Duration*** a 750 más lento será el servo. Esto

significa que debe cambiar el 650 en el comando **PULSOUT 12,650** a algo más cercano a 750. Si el Boe-Bot está ligéramente fuera de curso, quizá **PULSOUT 12,663** resulte. Si los servos están severamente dispares, quizá necesite **PULSOUT 12,690**.

Quizá le tome varios intentos para obtener el valor correcto. Digamos que su primer intent es **PULSOUT 12,663**, pero resulta no ser suficiente porque el Boe-Bot aún gira ligeramente a la izquierda. Entonces intenta **PULSOUT 12,670**. Quizá esto sobrecorrija y resulte que **PULSOUT 12,665** es exacto. Esto se llama proceso iterativo y se refiere al proceso de intentos repetidos y ajuste sucesivo para conseguir el valor correcto.

Si su Boe-Bot se carga a la derecha en vez de a la izquierda, quiere decir que necesita reducir la velocidad de la rueda izquierda reduciendo ***Duration*** de 850 en el comando **PULSOUT 13,850**. Nuevamente, entre más cerca esté este valor a 750, más lento será que el servo.

- ✓ Modifique BoeBotForwardTenSeconds.bs2 para que haga que su Boe-Bot vaya en línea recta.
- ✓ Use masking tape o una etiqueta para marcar cada servo con los mejores valores **PULSOUT**.
- ✓ Si su Boe-Bot ya viaja en línea recta, intente las modificaciones recién discutidas para ver el efecto. Debe causar que el Boe-Bot viaje en línea curveada en vez de línea recta.

Quizá encuentre que es una situación enteramente diferente cuando programa su Boe-Bot para avanzar en reversa.

- ✓ Modifique BoeBotForwardTenSeconds.bs2 para que haga que el Boe-Bot avance en reversa por 10 segundos.
- ✓ Repita la prueba de línea recta.
- ✓ Repita los pasos para corregir el argumento ***Duration*** del comando **PULSOUT** para enderezar el viaje en reversa del Boe-Bot's.

Afinando las vueltas

Pueden hacerse ajustes de Software para hacer que el Boe-Bot gire a un ángulo deseado, como a 90°. La cantidad de tiempo que el Boe-Bot usa para rotar sobre un punto determina que tanto girará. Puesto que el ciclo **FOR…NEXT** controla el tiempo de operación, puede ajustar el argumento ***EndValue*** del ciclo **FOR…NEXT** para llegar muy cerca al ángulo que desea.

He aquí la rutina izquierda de ForwardLeftRightBackward.bs2:

```
FOR counter = 1 TO 24      ' Giro izquierda - aprox 1/4 vuelta

  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20

NEXT
```

Digamos que él Boe-Bot gira un poco más que 90° (1/4 de un círculo completo). Intente **`FOR counter = 1 TO 23`**, o quizá **`FOR counter = 1 TO 22`**. Si no gira lo suficiente, incremente el tiempo de operación de la rotación incrementando el argumento ***EndValue*** del ciclo **`FOR…NEXT`** al valor necesario para completar el cuarto de vuelta.

Si se encuentra con un valor ligeramente sobrepasado a 90° y el otro ligeramente por debajo, intente escoger el valor que haga que gire un poco de más, luego reduzca ligeramente la velocidad de los servos. En el caso de rotación izquierda, ambos argumentos **`PULSOUT`** ***Duration*** deben ser cambiados de 650 a un poco más cercano a 750. Al igual que con el ejercicio de línea recta, este también será un proceso iterativo.

Su Turno – giros de 90°

- ✓ Modifique ForwardLeftRightBackward.bs2 para que haga giros precisos de 90°.
- ✓ Actualice ForwardLeftRightBackward.bs2 con los valores de **`PULSOUT`** que haya determinado para la línea recta y viaje en reversa.
- ✓ Actualice la etiqueta en cada servo con una nota acerca del valor adecuado ***EndValue*** para una vuelta de 90°.

El alfombrado puede causar errores de navegación. Si esta corriendo el Boe-Bot sobre alfombra, ¡no espere resultados perfectos! Una alfombra es como un green de golf—la inclinación del pelo puede afectar la forma en que viaja su Boe-Bot, especialmente en distancias largas. Para maniobras más precisas, use una superficie suave.

ACTIVIDAD #3: CALCULANDO DISTANCIAS

En muchos concursos robóticos, una navegación del robot más precisa resulta en mejores marcadores. Una modalidad popular de concursos robóticos es llamada " reconocimiento de punto muerto". El objeto de este concurso es hacer que su robot vaya a una o más posiciones y luego regrese exactamente a donde empezó.

Y se recuerde preguntando a sus padres la siguiente pregunta, una y otra vez, mientras que el camino a un destino de vacaciones o la casa de algún familiar:

"¿Ya llegamos?"

Quizá cuando creció un poco y aprendió dividir en la escuela, empezó a observar los letreros en el camino para ver qué tan lejos estaba de la ciudad destino. Luego, chocó el velocímetro en su cargo al dividir la velocidad entre la distancia, tuvo una buena estimación del tiempo que todavía les tomaría llegar. Quizá no había pensado en estos términos exactos, pero he aquí la ecuación estaba usando:

$$time = \frac{distance}{speed}$$

Ejemplo – Tiempo para el sistema inglés

si esta a 140 millas de su destino y viaja a 70 millas por hora, le tomará dos horas llegar.

$$\begin{aligned} time &= \frac{140\ miles}{70\ miles/hour} \\ &= 140\ miles \times \frac{1\,hour}{70\,miles} \\ &= 2\,hours \end{aligned}$$

Ejemplo – tiempo para el sistema decimal

si esta a 200 kilómetros de su destino y viaja a 100 km/h, le tomará dos horas llegar.

$$\begin{aligned} time &= \frac{200\,kilometers}{100\,kilometers/hour} \\ &= 200\ km \times \frac{1\,hour}{100\,km} \\ &= 2\,hours \end{aligned}$$

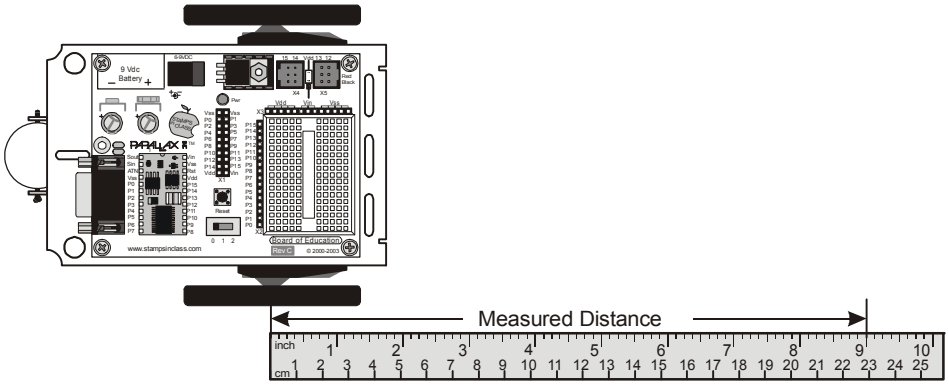
Puede hacer el mismo ejercicio con el Boe-Bot, excepto que tendrá el control de qué tan lejos esta el destino. He aquí la ecuación que usará:

$$servo\ run\ time = \frac{Boe\text{-}Bot\ distance}{Boe\text{-}Bot\ speed}$$

Tendrá que probar la velocidad del Boe-Bot. La forma más fácil de hacer esto es poner el Boe-Bot junto a una regla y hacer que viaje al frente por un segundo. Midiendo qué tan lejos llegó su Boe-Bot, sabrá la velocidad de su Boe-Bot. Si su regla indica pulgadas, su respuesta será en pulgadas por segundo (in/s), si tienes centímetros su respuesta será en centímetros por segundo (cm/s).

- ✓ Introduzca, salve y corra ForwardOneSecond.bs2.
- ✓ Coloque su Boe-Bot junto a una regla como se muestra en la Figura 4-3.
- ✓ Asegúrese de alinear el punto donde la rueda toca el suelo con la marca de 0 in/cm en la regla.

Figura 4-3: siguiendo la distancia de su Boe-Bot



- ✓ Presione el botón Reset en su tarjeta para volver a correr programa.
- ✓ Mida qué tan lejos llegó su Boe-Bot registrando la distancia donde ahora la rueda está tocando el piso:_____ in / cm.

```
' Programa ejemplo: ForwardOneSecond.bs2
' Robotica con el Boe-Bot - ForwardOneSecond.bs2
' Hace que el Boe-Bot avance por un segundo.
' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"
counter        VAR     Word

FREQOUT 4, 2000, 3000                    ' señal de programa en inicio/reset.

FOR counter = 1 TO 41

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT

END
```

También puede ver la distancia que acaba de registrar como la velocidad de su Boe-Bot, en unidades por segundo. Digamos que su Boe-Bot viajó 9 in (23 cm). Puesto que el tomo un segundo a su Boe-Bot para viajar esa distancia, significa que su Boe-Bot viaja alrededor de 9 in/s (23 cm/s). Ahora puede calcular cuántos segundos le toma a su Boe-Bot viajar una distancia en particular.

Pulgadas y centímetros por segundo

la abreviación para pulgadas es in y la abreviación para centímetros es cm. De igual manera, pulgadas por segundo se abrevia in/s y centímetros por segundo se abrevia cm/s. Ambas son medidas de velocidad convenientes para el Boe-Bot. Hay 2.54 cm en 1 in. Puede convertir pulgadas a centímetros multiplicando el número de pulgadas por 2.54. Puede convertir centímetros a pulgadas dividiendo el número de centímetros entre 2.54.

Ejemplo – tiempo para 20 pulgadas

A 9 in/s, su Boe-Bot tiene que viajar por 2.22 s para viajar 20 in.

$$\begin{aligned} time &= \frac{20\,in}{9\,in/s} \\ &= 20\,in \times \frac{1\,s}{9\,in} \\ &= 2.22\,s \end{aligned}$$

Ejemplo – tiempo para 51 cm

A 23 cm/s, su Boe-Bot tiene que viajar por 2.22 s para viajar 51 cm.

$$\begin{aligned} time &= \frac{51\,cm}{23\,cm/s} \\ &= 51\,cm \times \frac{1\,s}{23\,cm} \\ &= 2.22\,s \end{aligned}$$

En el Capítulo 2, Actividad #6, aprendimos que toma 24.6 ms (0.024 s) cada vez que los dos comandos **PULSOUT** y un comando **PAUSE** son ejecutados en un ciclo **FOR…NEXT**. El recíproco de este valor es el número de pulsos por segundo que el ciclo trasmite para cada servo. Un recíproco es cuando intercambia el numerador y el denominador de una división. Otra forma de obtener un recíproco es dividir el número uno entre un número o una fracción. En otras palabras, 1 ÷ 0.024 s/pulso = 40.65 pulsos/s.

Puesto que conoce la cantidad de tiempo que quiere que su Boe-Bot avance (2.22 s) y el número de pulsos que el BASIC Stamp envía a los servos cada segundo (40.65 pulsos/s), puede usar estos valores para calcular cuántos pulsos enviar a Los servos. Este es el número que tendrá que usar para el argumento ***EndValue*** del ciclo **FOR…NEXT**.

$$\begin{aligned} pulses &= 2.22s \times \frac{40.65\ pulses}{s} \\ &= 90.24...\ pulses \\ &\approx 90\ pulses \end{aligned}$$

El cálculo en este ejemplo tomó dos pasos. Primero, descifrar que tanto tiempo los servos tienen que correr para hacer que el Boe-Bot recorrauna cierta distancia, luego descifrar cuántos pulsos le toma a los servos correr por ese tiempo. Al saber que se tiene que multiplicar por 40.65 para pasar de tiempo a pulsos, se puede reducir esto a un solo paso.

$$pulses = \frac{Boe-Bot\ distance}{Boe-Bot\ speed} \times \frac{40.65\ pulses}{s}$$

Ejemplo – tiempo para 20 pulgadas

A 9 in/s, su Boe-Bot tiene que viajar por 2.22 s para recorrer 20 in.

$$\begin{aligned} pulses &= \frac{20\ in}{9\ in/s} \times \frac{40.65\ pulses}{s} \\ &= 20\ in \times \frac{1\ s}{9\ in} \times \frac{40.65\ pulses}{1\ s} \\ &= 20 \div 9 \times 40.65\ pulses \\ &= 90.333...pulses \\ &\approx 90\ pulses \end{aligned}$$

Ejemplo – tiempo para 51 cm

A 23 cm/s, su Boe-Bot tiene que viajar por 2.22 s para recorrer 51 cm.

$$\begin{aligned} pulses &= \frac{51\ cm}{23\ cm/s} \times \frac{40.65\ pulses}{s} \\ &= 51cm \times \frac{1\ s}{23cm} \times \frac{40.65\ pulses}{1s} \\ &= 51 \div 23 \times 40.65\ pulses \\ &= 90.136...pulses \\ &\approx 90\ pulses \end{aligned}$$

Su Turno – La Distancia de su Boe-Bot

Ahora es momento de intentarlo con distancias que usted escoja.

- ✓ Si aún no lo ha hecho, use una regla y el programa ForwardOneSecond.bs2 para determinar la velocidad de su Boe-Bot en in/s o cm/s.
- ✓ Decide que tan lejos quiere que viaje su Boe-Bot.
- ✓ Use la ecuación de pulsos para determinar cuántos pulsos entregar a los servos del Boe-Bot:

$$pulses = \frac{Boe-Bot\ distance}{Boe-Bot\ speed} \times \frac{40.65\ pulses}{s}$$

- ✓ Modifique BoeBotForwardOneSecond.bs2 para que entregue el número de pulsos que determinó para su distancia.
- ✓ Correr programa y por ende para ver qué tan cerca estuvo.

Esta técnica tiene fuentes de error. La Actividad que acaba de completar no toma en cuenta que al Boe-Bot le toma un cierto número de pulsos llegar a velocidad plena. Ni tomó en cuenta la distancia que este pudo recorrer antes de llegar al paro total. Las velocidades del servo también serán más lentas a medida que las baterías pierdan su carga.

Puede incrementar la precisión de las distancias de su Boe-Bot con dispositivos llamados encoders, los cuales contabilizan los hoyos en las ruedas del Boe-Bot a medida que pasan. Kits de encoders y otros accesorios específicos del Boe-Bot están disponibles en www.parallax.com/go/Boe-Bot.

ACTIVIDAD #4: MANIOBRAS—RAMPEO

El rampeo es una forma de incrementar gradualmente o decrementar la velocidad de los servos en vez de arrancar o parar abruptamente. Esta técnica puede incrementar la esperanza de vida tanto de las baterías del Boe-Bot como de sus servos.

Programación para rampeo

La clave para rampear es usar variables junto con constantes para el argumento **`Duration`** del comando **`PULSOUT`**. La Figura 4-4 muestra un ciclo **`FOR…NEXT`** que puede rampear la velocidad desde paro total a velocidad plena al frente. Cada vez que el ciclo **`FOR…NEXT`** se repite la variable **`pulseCount`** se incrementa en 1. En la primera pasada **`pulseCount`** es 1 y es como si usara los comandos **`PULSOUT 13, 751`** y **`PULSOUT 12, 749`**. La segunda pasada a través del ciclo, el valor de **`pulseCount`** es 2 y es como usar los comandos **`PULSOUT 13, 752`** y **`PULSOUT 12, 748`**. A medida que se incrementa el valor de la variable **`pulseCount`**, también lo hace la velocidad de los servos. La centésima pasada por el ciclo la variable **`pulseCount`** vale 100, es como usar los commandos **`PULSOUT 13, 850`** y **`PULSOUT 12, 650`**, que es la velocidad plena al frente.

```
pulseCount    VAR     Word

FOR pulseCount = 1 TO 100

  PULSOUT 13, 750 + pulseCount
  PULSOUT 12, 750 - pulseCount
  PAUSE 20

NEXT
```

1, 2, 3, …100

Figura 4-4
Ejemplo de rampeo

Retomando del Capítulo 2, Actividad #5, los ciclos **FOR…NEXT** también pueden contar hacia abajo, de un numero mayor a uno menor. Puede usar esto para rampear la velocidad hacia abajo nuevamente con **FOR pulseCount = 100 TO 1**. He aquí un programa que usa ciclos **FOR…NEXT** para rampear hacia arriba hasta velocidad plena y luego hacia abajo.

Programa Ejemplo: StartAndStopWithRamping.bs2

- ✓ Introduzca, salve y corra StartAndStopWithRamping.bs2.
- ✓ Verifique que el Boe-Bot gradualmente acelera a velocidad plena, mantiene velocidad plena por un tiempo y luego gradualmente desacelera hasta paro total.

```
' -----[ Titulo ]-------------------------------------------------------------
' Robotica con el Boe-Bot - StartAndStopWithRamping.bs2
' Rampea hacia arriba, avanza, rampea hacia abajo.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

pulseCount    VAR     Word               ' Ciclo contador FOR...NEXT.

' -----[ Inicializacion ]-----------------------------------------------

FREQOUT 4, 2000, 3000                    ' Señal de programa en inicio/reset.

' -----[ Rutina principal]-----------------------------------------------

' Rampeo hacia arriba.

FOR pulseCount = 1 TO 100               ' Ciclo rampea arriba por 100 pulsos.
  PULSOUT 13, 750 + pulseCount          ' Pulso = 1.5 ms + pulseCount.
  PULSOUT 12, 750 - pulseCount          ' Pulso = 1.5 ms – pulseCount.
  PAUSE 20                              ' Pausa por 20 ms.

NEXT

' Continua al frente por 75 pulsos.

FOR pulseCount = 1 TO 75                  ' Ciclo envía 75 pulsos al frente.
  PULSOUT 13, 850                         ' Pulso de 1.7 ms al servo izq.
  PULSOUT 12, 650                         ' Pulso de 1.3 ms servo der.
  PAUSE 20                                ' Pausa por 20 ms.
NEXT
```

```
' Rampeo hacia abajo, dese marcha al frente a paro total.

FOR pulseCount = 100 TO 1                ' Ciclo rampea abajo por 100 pulsos.
  PULSOUT 13, 750 + pulseCount           ' Pulso = 1.5 ms + pulseCount.
  PULSOUT 12, 750 - pulseCount           ' Pulso = 1.5 ms - pulseCount.
  PAUSE 20                               ' Pausa por 20 ms.
NEXT

END                                      ' Parado hsta un reset.
```


Su Turno

También puede crear rutinas para combiner rampeos arriba o abajo con las otras maniobras. He aquí un ejemplo de como rampear arriba hasta velocidad plena yendo en reversa en vez de hacia al frente. La única diferencia entre esta rutina y la rutina de rampeo hacia al frente es que el valor de **pulseCount** es restado de 750 en el comando **PULSOUT 13**, donde antes fue sumado. Igualmente, **pulseCount** es sumado al valor de 750 en el commando **PULSOUT 12**, donde antes era restado.

```
' Rampeo hacia arriba hasta velocidad plena en reversa

FOR pulseCount = 1 TO 100

  PULSOUT 13, 750 - pulseCount
  PULSOUT 12, 750 + pulseCount
  PAUSE 20

NEXT
```

También puede hacer una rutina para reampear en una vuelta agregando el valor de **pulseCount** a 750 en ambos comandos **PULSOUT**. Restando **pulseCount** a 750 en ambos comandos **PULSOUT**, puede rampear en una vuelta hacia otra dirección. He aquí un ejemplo de un cuarto de vuelta con rampeo. Los servos no tienen oportunidad de llegar a velocidad plena antes de que tengan que reducir su velocidad nuevamente.

```
' Rampeo hacia arriba en vuelta a la dercha.

FOR pulseCount = 0 TO 30

  PULSOUT 13, 750 + pulseCount
  PULSOUT 12, 750 + pulseCount
  PAUSE 20

NEXT
```

```
' Rampeo hacia abajo en vuelta a la derecha

FOR pulseCount = 30 TO 0

  PULSOUT 13, 750 + pulseCount
  PULSOUT 12, 750 + pulseCount
  PAUSE 20

NEXT
```

- ✓ Abra ForwardLeftRightBackward.bs2 de la Actividad #1, y salvelo como ForwardLeftRightBackwardRamping.bs2.
- ✓ Modifique el programa nuevo para que su Boe-Bot rampee hacia y desde cada maniobra. Nota: puede usar los extractros de código arriba indicados y extractos semejantes de StartAndStopWithRamping.bs2.

ACTIVIDAD #5: SIMPLIFIQUE LA NAVEGACIÓN CON SUBRUTINAS

En el siguiente Capítulo, su Boe-Bot tendrá que ejecutar maniobras para evitar obstáculos. Uno de los ingredientes clave para evitar obstáculos es ejecutar maniobras pre-programadas. Una forma de ejecutar maniobras pre-programadas es con subrutinas. Esta actividad introduce subrutinas, y también dos estrategias diferentes para crear maniobras reutilizables con subrutinas.

Dentro de la Subrutina

Hay dos partes de una subrutina PBASIC. Una parte es la llamada de la subrutina. Es el comando en el programa que le indica brincar a la parte reutilizable del código, luego regresar cuando termine. La otra parte es la propia subrutina. Empieza con una etiqueta que sirve como su nombre y termina con un comando **`RETURN`**. Los comandos entre la etiqueta y el comando **`RETURN`** delimitan el bloque de código que hace el trabajo que quiere que haga la subrutina.

La Figura 4-5 muestra parte de un programa PBASIC que contiene una llamada de subrutina y una subrutina. La llamada de subrutina es el comando **`GOSUB My_Subroutine`**. La propia subrutina es todo desde la etiqueta **`My_Subroutine:`** hasta el commando **`RETURN`**. He aquí como trabaja. Cuando el programa llega al commando **`GOSUB My_Subroutine`**, busca la etiqueta **`My_Subroutine:`** . Como lo muestra la flecha (1), el programa brinca a la etiqueta **`My_Subroutine:`** y empieza a ejecutar comandos. El programa continua hacia abajo línea por línea desde la etiqueta, así es que verá el

mensaje “Command in subroutine” en su Terminal de Depuración. **PAUSE 1000** causa una pausa de 1 segundo. Luego, cuando el programa llega al comando **RETURN**, la flecha (2) muestra como brinca de regreso al comando immediatamente después del comando **GOSUB**. En este caso, un comando **DEBUG** despliega el mensaje “After subroutine.”

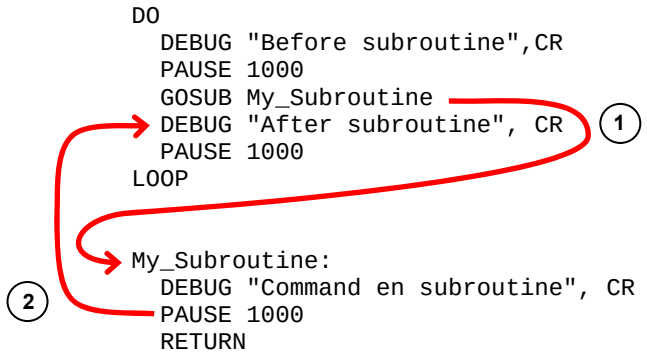


Figura 4-5
Elementos Básicos de una subrutina

Programa Ejemplo– OneSubroutine.bs2

- ✓ Introduzca, salve y corra OneSubroutine.bs2.

```
' Robotica con el Boe-Bot - OneSubroutine.bs2
' Este programa demuestra una simple llamada a su rutina.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Before subroutine",CR
PAUSE 1000
GOSUB My_Subroutine
DEBUG "After subroutine", CR
END

My_Subroutine:
  DEBUG "Command in subroutine", CR
  PAUSE 1000
  RETURN
```

- ✓ Observe su Terminal de depuración, y presione el botón Reset unas cuantas veces. Debería tener el mismo juego de 3 mensajes en el orden correcto cada vez.

He aquí un programa ejemplo que tiene dos subrutinas. Una subrutina hace un tono agudo mientras que el otro hace un tono grave. Los comandos entre **DO** y **LOOP** llaman cada subrutina en turno. Intente este programa y note el efecto.

Programa Ejemplo – TwoSubrutinas.bs2

- ✓ Introduzca, salve y corra TwoSubrutinas.bs2.

```
' Robotica con el Boe-Bot - TwoSubrutinas.bs2
' Demuestra que una subrutina es un bloque de comandos reutilizable.

' {$STAMP BS2}
' {$PBASIC 2.5}

DO
  GOSUB High_Pitch
  DEBUG "Back en main", CR
  PAUSE 1000
  GOSUB Low_Pitch
  DEBUG "Back en main again", CR
  PAUSE 1000
  DEBUG "Repeat...",CR,CR
LOOP

High_Pitch:
  DEBUG "High pitch", CR
  FREQOUT 4, 2000, 3500
  RETURN

Low_Pitch:
  DEBUG "Low pitch", CR
  FREQOUT 4, 2000, 2000
  RETURN
```

Intentemos poner las rutinas de navegación hacia el frente, izquierda, derecha y atrás dentro de subrutinas. He aquí un ejemplo:

Programa Ejemplo– MovementsWithSubrutinas.bs2

- ✓ Introduzca, salve y corra MovementsWithSubrutinas.bs2. Tip: puede usar el menú Edit en el editor del BASIC Stamp para copiar y pegar bloques de código de un programa a otro.

```
' Robotica con el Boe-Bot - MovementsWithSubrutinas.bs2
' Hace movimientos adelante, izq, der y atras en sus rutinas reutilizables.
' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                    ' señal de programa en inicio/reset.

GOSUB Forward
GOSUB Left
GOSUB Right
GOSUB Backward

END

Forward:
  FOR counter = 1 TO 64
    PULSOUT 13, 850
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  PAUSE 200
  RETURN

Left:
  FOR counter = 1 TO 24
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  PAUSE 200
  RETURN

Right:
  FOR counter = 1 TO 24
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  PAUSE 200
  RETURN

Backward:
  FOR counter = 1 TO 64
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```

Debería reconocer el patrón de movimientos que hace su Boe-Bot; ese mismo hecho por ForwardLeftRightBackward.bs2. Claramente hay muchas formas diferentes de estructurar un programa que resultarán en movimientos semejantes. Una tercera forma se da en el ejemplo a continuación.

Programa Ejemplo – MovementsWithVariablesAndOneSubroutine.bs2

He aquí otro programa ejemplo que causa que su Boe-Bot ejecute las mismas maniobras, pero sólo use una subrutina y algunas variables para hacerlo.

Seguramente ha notado que, hasta este punto, cada maniobra del Boe-Bot ha sido lograda con bloques de códigos semejantes. Compare estos dos segmentos:

```
' Frente velocidad plena
FOR counter = 1 TO 64
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

```
' Rampeo abajo, de vel. plena en reversa
FOR pulseCount = 100 TO 1
  PULSOUT 13, 750 - pulseCount
  PULSOUT 12, 750 + pulseCount
  PAUSE 20
NEXT
```

Lo que hace que estos dos bloques de código ejecuten maniobras diferentes son cambios en los argumentos **FOR** ***StartValue*** y ***EndValue***, y los argumentos **PULSOUT** ***Duration***. Estos argumentos pueden ser variables, y estas variables pueden ser cambiadas repetidamente durante la ejecución del programa para generar maniobras diferentes. En vez de usar subrutinas separadas con argumentos específicos **PULSOUT** ***Duration*** para cada maniobra, usa la misma subrutina una y otra vez. La clave para hacer maniobras diferentes es establecer las variables en valores correctos para la maniobra que quiere ejecutar antes de llamar a la subrutina.

- ✓ Introduzca, salve y corra MovementWithVariablesAndOneSubroutine.bs2.

```
' Robotica con el Boe-Bot - MovementWithVariablesAndOneSubroutine.bs2
' Hace una rutina de navegación que acepta parámetros.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"
```

```
counter        VAR     Word
pulseLeft      VAR     Word
pulseRight     VAR     Word
pulseCount     VAR     Byte

FREQOUT 4, 2000, 3000                    ' señal de programa en inicio/reset.

' Forward
pulseLeft = 850: pulseRight = 650: pulseCount = 64: GOSUB Navigate

' Left turn
pulseLeft = 650: pulseRight = 650: pulseCount = 24: GOSUB Navigate

' Right turn
pulseLeft = 850: pulseRight = 850: pulseCount = 24: GOSUB Navigate

' Backward
pulseLeft = 650: pulseRight = 850: pulseCount = 64: GOSUB Navigate

END

Navigate:
  FOR counter = 1 TO pulseCount
    PULSOUT 13, pulseLeft
    PULSOUT 12, pulseRight
    PAUSE 20
  NEXT
  PAUSE 200
  RETURN
```


¿Su Boe-Bot ejecutó la conocida secuencia al frente-izquierda-derecha-atrás? Este programa puede ser difícil de leer al principio, porque las instrucciones están arregladas en una nueva forma. En vez de tener cada declaración de variable y cada comando **GOSUB** en líneas diferentes, están agrupados y separados por dos puntos. Aquí, la función de los dos puntos es la misma que un retorno de carro para separar cada instrucción PBASIC. Usar dos puntos de esta forma permite que todos los valores de una variable para una maniobra determinada sean guardados juntos y en la misma línea con la llamada a la subrutina.

Su Turno

Aquí está su concurso de "reconocimiento de punto muerto" mencionado antes.

- ✓ Modifique MovementWithVariablesAndOneSubroutine.bs2 para hacer que su Boe-Bot dibuje un cuadrado, hacia el frente en los primeros dos lados y en reversa en los otros dos. Tip: necesitará usar sus propios argumentos **PULSOUT** ***EndValue*** que determinó en la Actividad #2, página 109.

ACTIVIDAD #6: TEMA AVANZADO—HACIENDO MANIOBRAS COMPLEJAS EN EEPROM

Cuando baja programas PBASIC a su BASIC Stamp, el editor BASIC Stamp convierte su programa en valores numéricos llamados tokens. Estos tokens son lo que usa el BASIC Stamp como instrucciones para ejecutar el programa. Son guardados en uno de los 2 circuitos negros más pequeños en la parte superior de su BASIC Stamp. Este circuito es un tipo especial de memoria de computadora llamada EEPROM, que quiere decir memoria de sólo lectura programable y borrable eléctricamente (**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory – EEPROM). La EEPROM del BASIC Stamp puede guardar 2048 bytes (2 KB) de información. Lo que no se usa para guardar programas (de la dirección 2047 a la 0) puede ser usado para guardar datos (de la dirección 0 a la 2047).

Si los datos que guarde en EEPROM chocan con su programa, el programa PBASIC no se ejecutará adecuadamente.

La memoria EEPROM es diferente al almacenaje variable de la RAM (**R**andom **A**ccess **M**emory – memoria de acceso aleatorio) en algunos aspectos:

- A la EEPROM le toma más tiempo para guardar un valor, algunas veces hasta varios milisegundos.
- La EEPROM acepta un número finito de ciclos de escritura, cerca de 10 millones de escrituras. RAM tiene capacidades de escritura/lectura ilimitadas.
- La función primaria de la EEPROM es guardar programas; los datos pueden ser guardados en el espacio no ocupado.

Puede ver el contenido de la EEPROM del BASIC Stamp en el Editor BASIC Stamp haciendo click en Run y seleccionando Memory Map. La Figura 4-6 muestra el mapa de memoria de MovementsWithSubrutinas.bs2. Note el mapa EEPROM condensado en el lado izquierdo de la figura. Esta area asiurada en el recuadro pequeño en la parte inferior muestra la cantidad de EEPROM que ocupa MovementsWithSubrutinas.bs2.

Las imagenes del mapa de memoria mostradas en esta Actividad fueron tomadas del Editor BASIC Stamp v2.1. Si está usando una version diferente del Editor BASIC Stamp, su mapa de memoria contendrá la misma información, pero estará formateada diferente.

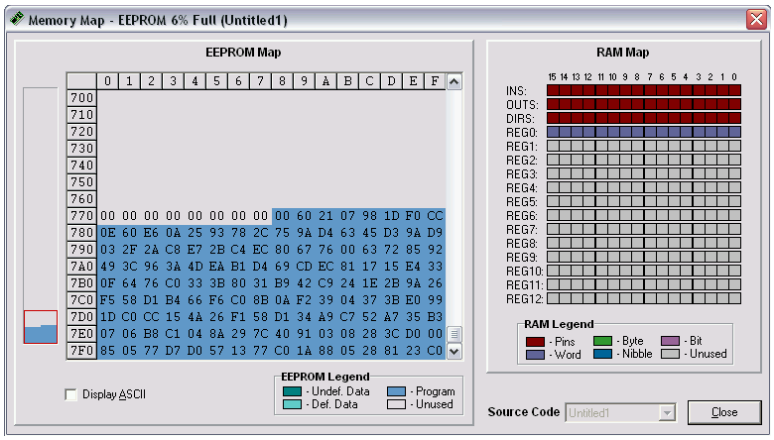


Figura 4-6
Mapa de Memoria del Editor BASIC Stamp

Y ya que estamos aquí, note también que la variable **`counter`** que declaramos word esta visible en el Registro 0 del Map RAM.

Este programa pudo parecerle largo al teclearlo, pero sólo tomó 136 de los 2048 bytes disponibles de la memoria de programa. De hecho hay bastante espacio para una lista bastante larga de instrucciones. Puesto que un carácter ocupa 1 byte en la memoria, hay espacio para 1912 Instrucciones de dirección de un caracter.

Navegación EEPROM

Hasta este punto hemos intentado 3 estrategias de programación diferentes para hacer que su Boe-Bot marche al frente, gire a la izquierda, a la derecha y marche en reversa nuevamente. Cada técnica tiene sus méritos, pero todas serían torpes si quisiera que su Boe-Bot ejecutara un juego de maniobras más largo y complejo. Los siguientes programas ejemplo usarán los bloques de código ahora familiares en subrutinas para cada maniobra básica. A cada maniobra se le da un código de una letra como referencia. Listas largas de estas letras código puede ser guardadas en la EEPROM y luego leídas y de codificadas durante la ejecución del programa esto evita el tedio de repetir listas largas de subrutinas, o tener que cambiar las variables antes de ir a cada comando **`GOSUB`**.

Esta técnica de programación requiere algunas instrucciones PBASIC nuevas: la directiva **`DATA`** y los comandos **`READ`** y **`SELECT...CASE...ENDSELECT`**. Echemos un vistazo a cada uno antes de intentar un programa ejemplo.

A cada una de las maniobras básicas se le da un código de letra único que corresponde a su subrutina: F para **`Forward (al frente)`**, B para **`Backward (atrás)`**, L para **`Left_Turn (vuelta izquierda)`**, y R for **`Right_Turn (vuelta derecha)`**. Movimientos complejos pueden ser rápidamente coreografiados haciendo una cadena de estas letras código. La última letra en esta cadena es una Q, que significa "abandona" ("quit") cuando los movimientos están completos. La lista es guardada en la EEPROM durante la descarga del programa con la directiva **`DATA`**, que se ve como sigue:

```
DATA            "FLFFRBLBBQ"
```

Cada letra es guardada en un byte de la EEPROM, empezando en la dirección 0 (a menos que le digamos que empiece en otro lado). El comando **`READ`** puede ser usado para recuperar esta lista a partir de la EEPROM mientras que programa está corriendo. Estos valores pueden ser leídos dentro de un **`DO…LOOP`** como sigue:

```
DO UNTIL (instruction = "Q")
  READ address, instruction
  address = address + 1
  ' bloque de codigo PBASIC omitido aqui.
LOOP
```

La variable **`address`** es la localidad de cada byte en la EEPROM que está reteniendo una letra código. La variable **`instruction`** retendrá el valor real de cada byte, nuestra letra código. Note que con cada pasada por el ciclo el valor de la variable **`address`** es incrementado en uno. Esto permitirá a cada letra ser leída de bytes consecutivos en la EEPROM, empezando la dirección 0.

El comando **`DO…LOOP`** tiene condiciones opcionales que son útiles para diferentes circunstancias. **`DO UNTIL (`*`condition`*`)...LOOP`** permite que el ciclo se repita hasta que cierta condición ocurra. **`DO WHILE (`*`condition`*`)...LOOP`** permite que el ciclo se repita sólo mientras que cierta condición exista. Nuestro programa ejemplo usará **`DO…LOOP UNTIL (`*`condition`*`).`** En este caso, esto causa que **`DO…LOOP`** se mantenga repitiéndose hasta que el caracter "Q" sea leído de la EEPROM.

`SELECT...CASE...ENDSELECT` puede ser usado para seleccionar una variable y evaluarla caso por caso y ejecutar bloques de código en consecuencia. He aquí el bloque de código que verá en cada valor de letra retenido en la variable de la instrucción y luego llama la subrutina apropiada para cada instancia, o caso, para una letra determinada.

```
  SELECT instruction
```

```
    CASE "F": GOSUB Forward
    CASE "B": GOSUB Backward
    CASE "R": GOSUB Right_Turn
    CASE "L": GOSUB Left_Turn
  ENDSELECT
```

A continuación están todos estos conceptos, todos juntos en un sólo programa.

Programa Ejemplo: EepromNavigation.bs2

- ✓ Lea cuidadosamente las instrucciones del código y los comentarios en EepromNavigation.bs2 para entender lo que hace cada parte del programa.
- ✓ Introduzca, salve y corra EepromNavigation.bs2.

```
' Robotica con el Boe-Bot - EepromNavigation.bs2
' Navega usando caracteres guardados en EEPROM.
' {$STAMP BS2}                      ' Directiva Stamp.
' {$PBASIC 2.5}                     ' Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Variables ]-------------------------------------------------------

pulseCount     VAR     Word         ' Guarda el número de pulsos.
address        VAR     Byte         ' guarda la dirección EEPROM.
instruction    VAR     Byte         ' guarda la instrucción EEPROM.

' -----[ Datos EEPROM ]----------------------------------------------------
-

'      Address: 0123456789          ' estas 2 líneas de comentarios muestran
'               ||||||||||          ' la dirección EEPROM de cada dato.
DATA           "FLFFRBLBBQ"         ' Instrucciones de navegacion.

' -----[ Inicializacion ]---------------------------------------------------

FREQOUT 4, 2000, 3000               ' Señal de programa en inicio/reset.

' -----[ Rutina Principal]--------------------------------------------------

DO UNTIL (instruction = "Q")

  READ address, instruction         ' Datos en address da instruction.
  address = address + 1             ' Suma 1 a address para la sig. lectura.

  SELECT instruction                ' Llama una subrutina diferente
    CASE "F": GOSUB Forward         ' para cada caracter posible
    CASE "B": GOSUB Backward        ' que puede ser obtenido de la
    CASE "L": GOSUB Left_Turn         ' EEPROM.
```

```
    CASE "R": GOSUB Right_Turn
  ENDSELECT

LOOP

END                                 ' Para la ejecucion al reset.

' -----[ Subrutina – Al Frente]-----------------------------------------------

Forward:                            ' Subrutina Al Frente.
  FOR pulseCount = 1 TO 64          ' Manda 64 pulsos al frente.
    PULSOUT 13, 850                 ' pulso de 1.7 ms al servo izq.
    PULSOUT 12, 650                 ' pulse de 1.3 ms al servo der.
    PAUSE 20                        ' Pausa por 20 ms.
  NEXT
  RETURN                            ' Regresa al ciclo de rutina principal.

' -----[ Subrutina – Atras ]--------------------------------------------------

Backward:                           ' Subrutina Atrás.
  FOR pulseCount = 1 TO 64          ' Manda 64 pulsos atras.
    PULSOUT 13, 650                 ' Pulso de 1.3 ms al servo izq.
    PULSOUT 12, 850                 ' Pulso de 1.7 ms al servo der.
    PAUSE 20                        ' Pausa por 20 ms.
  NEXT
  RETURN                            ' Regresa al ciclo de rutina principal.

' -----[ Subrutina - Vuelta_Izq ]----------------------------------------------

Left_Turn:                          ' Subrutina Vuelta a la izquierda.
  FOR pulseCount = 1 TO 24          ' Manda 24 pulsos de rotacion izq.
    PULSOUT 13, 650                 ' Pulso de 1.3 ms al servo izq.
    PULSOUT 12, 650                 ' Pulso de 1.3 ms al servo der.
    PAUSE 20                        ' Pausa por 20 ms.
  NEXT
  RETURN                            ' Regresa al ciclo de rutina principal.

' -----[ Subrutina – Vuelta_Der ]----------------------------------------------

Right_Turn:                         ' Subrutina vuelta a la derecha.
  FOR pulseCount = 1 TO 24          ' Manda 24 pusos de rotacion der.
    PULSOUT 13, 850                 ' Pulso de 1.7 ms al servo izq.
    PULSOUT 12, 850                 ' Pulso de 1.7 ms al servo der.
    PAUSE 20                        ' Pausa por 20 ms.
  NEXT
  RETURN                            ' Regresa al ciclo de rutina principal.
```

¿Su Boe-Bot viaja en un rectángulo, yendo al frente los primeros 2 lados y hacia atrás en los otros 2? Si parece más un trapezoide, quizá deba ajustar los argumentos ***EndValue*** del ciclo **`FOR...NEXT`** en las subrutinas de vueltas para hacer vueltas precisas de 90 grados.

Su Turno

- ✓ Con EepromNavigation.bs2 activo en el Editor BASIC Stamp, haga click en Run y seleccione Memory Map.

Sus Instrucciones guardadas aparecerán destacadas en azul al principo del Mapa de Memoria Detallado de la EEPROM (Figura 4-7). Los números mostrados son los códigos ASCII hexadecimales que corresponden a los caracteres que puso en su directiva **DATA**.

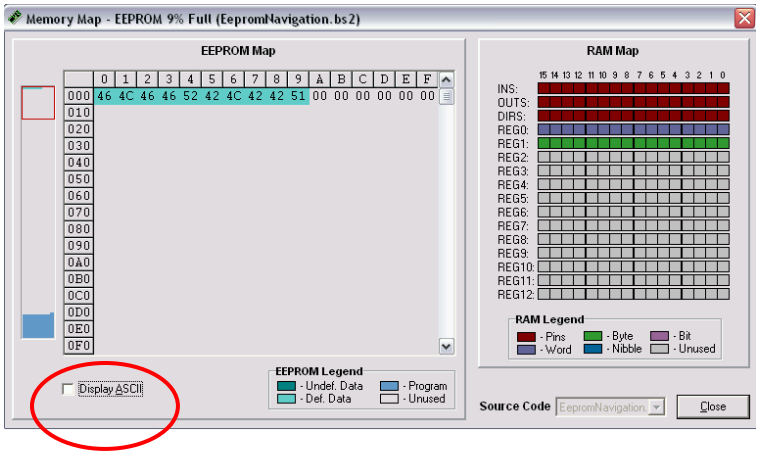


Figura 4-7
Mapa de Memoria con Instrucciones Guardadas Visibles en el Mapa EEPROM

- ✓ Haga Click en el recuadro Display ASCII cerca de la esquina inferior izquierda de la ventana del Mapa de Memoria.

Ahora las Instrucciones de dirección aparecerán en un formato más familiar mostrado en la Figura 4-8. En vez de códigos ASCII, aparecen los caracteres que grabó usando la directiva **DATA**.

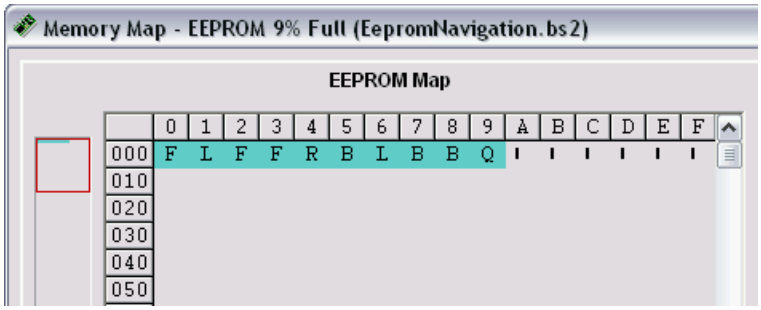


Figura 4-8
Sección del Mapa EEPROM detallado luego de marcar el recuado “Display ASCII Box”

Este programa guarda un total de 10 caracteres en la EEPROM. Estos 10 caracteres fueron accesados por la variable **address** del comando **READ**. La variable **address** fue declarada como un byte y puede accesar hasta 256 localidades, muy por arriba de las 10 que necesitamos. Si **address** es re-declarada como variable word, teoricamente accesaría hasta 65535, muchas más localidades que las disponibles. Recuerde que si su programa crece, disminuyen las direcciones EEPROM disponibles para guardar datos.

Puede modificar la cadena de datos existente por un nuevo juego de direcciones. También puede agregar instrucciones **DATA** adicionales. Como los datos son guardados secuencialmente el primer caracter en la segunda cadena de datos se guardará inmediatamente después del último caracter en la primera cadena de datos.

- ✓ Intente cambiando, agregando y borrando caracteres en la directive **DATA**, y volviendo a correr el programa. Recuerde que el último caracter en la directiva **DATA** debe ser siempre una "Q."
- ✓ Modifique la directiva **DATA** para hacer que su Boe-Bot ejecute la ya conocida secuencia de movimientos al frente-izquierda-derecha-reversa.
- ✓ Intente agregar una segunda directive **DATA**. Recuerde eliminar la "Q" al final de la primera directiva **DATA** y agréguela al final de la segunda. De otra forma, el programa solo ejecutará los comandos en la primera directiva **DATA**.

Programa Ejemplo– EepromNavigationWithWordValues.bs2

El siguiente programa ejemplo parece complicado al principio, pero es una forma muy eficiente de diseñar programas para coreografías personalizadas con el Boe-Bot. Este programa ejemplo usa el almacenaje de datos en EEPROM, pero no usa subrutinas. En vez de ello, se usa un solo bloque de código con variables en el lugar de los argumentos ***EndValue*** y **PULSOUT** ***Duration*** del cliclo **FOR...NEXT**.

Por defecto, la directiva **DATA** guarda bytes de información en la EEPROM. Para guardar elementos de datos de tamaño word, puede agregar el modificador **Word** a la directiva **DATA**, antes de cada dato en su cadena. Cada elemento de tamaño word usará 2 bytes de almacenaje EEPROM, asi que los datos deberán ser accesados en posiciones o direcciones salteadas. Cuando use más de una directiva **DATA**, es más conveniente asignar una etiqueta a cada uno. De esta forma sus comandos **READ** podrán referirse a la etiqueta para recuperar elementos de dstos sin tener que preocuparse en qué dirección EEPROM empieza cada cadena de elementos de datos. Eche un vistazo a este extracto de código:

```
' addressOffset   0         2         4         6        8
Pulses_Count DATA Word 64,  Word 24,  Word 24,  Word 64, Word 0
Pulses_Left  DATA Word 850, Word 650, Word 850, Word 650
Pulses_Right DATA Word 650, Word 650, Word 850, Word 850
```


Cada una de las 3 lineas **DATA** comienza con su propia etiqueta. El modificador **Word** va antes de cada elemento de datos y los elementos están separados por comas. Estas cadenas de datos serán guardadas en la EEPROM una después de otra. No tendremos que hacer cuentas para saber el número de dirección de un elemento de datos ya que lo harán las etiquetas y la variable **addressOffset**. El comando **READ** usa cada etiqueta para determiner la dirección EEPROM en donde inicia esa cadena y luego suma el valor de la **addressOffset** para saber cuántos números de direcciones hay que recorrerse para encontrar el elemento de datos ***DataItem*** correcto. El ***DataItem*** encontrado en la dirección resultante ***Address*** será guardado en el argumento ***Variable*** del comando **READ**. Note que **Word** también va antes de la variable que guarda el valor obtenido en la EEPROM.

```
DO
  READ Pulses_Count + addressOffset, Word pulseCount
  READ Pulses_Left + addressOffset, Word pulseLeft
  READ Pulses_Right + addressOffset, Word pulseRight

  addressOffset = addressOffset + 2

  ' bloque de codigo PBASIC omitido aqui.
LOOP UNTIL (pulseCount = 0)
```

En la primera pasada por el ciclo, **addressOffset** = 0. El primer **READ** recuperará un valor 64 de la primera dirección en la etiqueta **Pulses_Count** y lo colocará en la variable **pulseCount**. El segundo **READ** recupera un valor de 850 de la primera dirección especificada por la etiqueta **Pulses_Left** y lo coloca en la variable **pulseLeft**. El tercer **READ** recupera un valor de 650 de la tercera dirección especificada por la etiqueta **Pulses_Right** y lo coloca en la variable **pulseRight**. Note que estos son los 3 valores en la columna "0" del primer extracto de código anterior. Cuando el valor de esas variables se colocan en el siguiente bloque de código, esto:

```
FOR counter = 1 TO pulseCount
  PULSOUT 13, pulseLeft
  PULSOUT 12, pulseRight
  PAUSE 20
NEXT
```

....se convierte en esto....

```
FOR counter = 1 TO 64
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

¿Reconoce la maniobra básica generada por este bloque de código?

- ✓ Vea las otras columnas del extracto de código de la página 133 y anticipe cómo se verá el ciclo **FOR…NEXT** en su segunda, tercera y cuarta pasadas.
- ✓ Vea la declaración **LOOP UNTIL (pulseCount = 0)** en el siguiente programa. El operador **<>** significa "no es igual a." ¿Qué pasará en la quinta pasada por el ciclo?
- ✓ Introduzca, salve y corra EepromNavigationWithWordValues.bs2.

```
' Robotica con el Boe-Bot - EepromNavigationWithWordValues.bs2
' Guarda listas de valores word que dictan.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

' -----[ Variables ]-------------------------------------------------------

counter        VAR     Word
pulseCount     VAR     Word            ' Guarda numero de pulsos.
addressOffset  VAR     Byte            ' Guarda corrimiento desde la etiqueta.
instruction    VAR     Byte            ' Guarda instruccion EEPROM.
pulseRight     VAR     Word            ' Guarda ancho de pulsos de servo.
pulseLeft      VAR     Word

' -----[ Datos EEPROM]-----------------------------------------------------

' addressOffset       0         2         4         6        8
Pulses_Count DATA     Word 64,  Word 24,  Word 24,  Word 64, Word 0
Pulses_Left  DATA     Word 850, Word 650, Word 850, Word 650
Pulses_Right DATA     Word 650, Word 650, Word 850, Word 850

' -----[ Inicializacion ]--------------------------------------------------

FREQOUT 4, 2000, 3000                    ' Señal de programa en inicio/reset.

' -----[ Rutina Principal ]------------------------------------------------

DO

  READ Pulses_Count + addressOffset, Word pulseCount
  READ Pulses_Left + addressOffset, Word pulseLeft
  READ Pulses_Right + addressOffset, Word pulseRight

  addressOffset = addressOffset + 2

  FOR counter = 1 TO pulseCount
    PULSOUT 13, pulseLeft
```

```
    PULSOUT 12, pulseRight
    PAUSE 20
  NEXT


LOOP UNTIL (pulseCount = 0)

END                                          ' Para ejecucion hasta reset.
```


¿Su Boe-Bot ejecutó los ya familiares movimientos adelante-izquierda-derecha-atrás? ¿Ya le aburre esta secuencia? ¿Quisiera ver a su Boe-Bot hacer algo diferente, o coreografear su propia rutina?

Su turno – Haciendo sus propias rutinas de navegación personalizadas

- ✓ Salve EepromNavigationWithWordValues.bs2. bajo un nuevo nombre.
- ✓ Remplace las directivas **`DATA`** con las que están a continuación.
- ✓ Corra el programa modificado y véa lo que hace su Boe-Bot.

```
Pulses_Count DATA Word 60,  Word 80,  Word 100, Word 110,
                  Word 110, Word 100, Word 80,  Word 60,  Word 0
Pulses_Left  DATA Word 850, Word 800, Word 785, Word 760, Word 750,
                  Word 740, Word 715, Word 700, Word 650, Word 750
Pulses_Right DATA Word 650, Word 700, Word 715, Word 740, Word 750,
                  Word 760, Word 785, Word 800, Word 850, Word 750
```

- ✓ Haga una table con 3 renglones, uno para cada directiva **`DATA`**, y una columna para cada maniobra que quiera que haga su Boe-Bot, mas una para el elemento **`Word 0`** en el renglón **`Pulses_Count`**.
- ✓ Use la tabla para planear la coreografía de su Boe-Bot, completando los argumentos ***EndValue*** y **`PULSOUT`** ***Duration*** del ciclo **`FOR...NEXT`** que necesitará para cada bloque de código de cada manibra.
- ✓ Modifique su programa con recién trazadas directivas **`DATA`**.
- ✓ Introduzca, salve y corra su programa personalizado. ¿Hizo su Boe-Bot lo que quería que hiciera? Insista hasta que lo haga.

RESUMEN

Este Capítulo present las maniobras básicas del Boe-Bot: al frente, atrás, girando sobre su eje para dar vuelta a la derecha o izquierda y pivotear. El tipo de maniobra es determinado por los argumentos ***Duration*** del comando **PULSOUT**. La duración de la maniobra es determinada por los argumentos ***StartValue*** y ***EndValue*** del ciclo **FOR…NEXT**.

El Capítulo 2 incluyó un ajuste de hardware, físicamente centrando los servos del Boe-Bot con un desarmador. Este Capítulo se enfocó en ajustes finos hechos al manipular el software. Específicamente, una diferencia en la velocidad de rotación entre ambos servos fue compensada cambiando el argumento ***Duration*** del comando **PULSOUT** en el servo más velóz. Esto cambia la ruta del Boe-Bot de una curva a una linea recta si los servos no están perfectamente empatados. Para refinar las vueltas a un ángulo deseado pueden ajustarse los argumentos ***StartValue*** y ***EndValue*** en un ciclo **FOR…NEXT**.

Se puede programar el Boe-Bot para viajar una distancia pre-definida midiendo la distancia que viaja en un segundo, con la ayuda de una regla. Usando esta distancia y el número de pulsos en un segundo de trayecto puede calcular el número de pulsos requeridos para cubrir una distancia deseada.

El rampeo fue presentado como un modo de gradualmente acelerar y desacelerar. Es más amable con los servos y recomendamos que use sus propias rutinas de rampeo en vez de las rutinas que abruptamente inician y paran presentadas en los programas ejemplo. El rampeo se considgue tomando la misma variable que se usa como el argumento ***Counter*** en un ciclo **FOR…NEXT** e incrementándo o disminuyéndo progresivamente a partir de 750 el argumento ***Duration*** del comando **PULSOUT**.

Se presentaron subrutinas como un modo de hacer maniobras pre programadas reutilizables por un programa PBASIC. En vez de escribir un ciclo entero **FOR…NEXT** para cada maniobra nueva, una solo subrutina que contenga un ciclo **FOR…NEXT** puede ser ejecutada según se necesite con el comando **GOSUB**. Una subrutina comienza con una etiqueta, y termina con el comando **RETURN**. Una subrutina es llamada a partir del programa principal con un comando **GOSUB**. Cuando se termina la subrutina y encuentra el comando **RETURN**, el siguiente comando a ser ejecutado es el que está inmediatamente después del comando **GOSUB**.

La EEPROM del BASIC Stamp guarda al programa que corre, pero puede tomar ventaja cualquier porción no utilizar del programa para guardar valores. Esta es una excelente

forma de guardar rutinas de navegación personalizadas. La directiva **DATA** puede guardar valores en la EEPROM. Se guardan Bytes por defecto, pero al agregar el modificador **Word** a cada elemento de datos permite guardar valores de hasta 65535 en espacios de memoria EEPROM de dos bytes. Puede leer valores a partir de la EEPROM usando el commando **READ**. Si está recuperando una variable de tamaño word, asegúrese de colocar un modificador **Word** antes de la variable que recibirá el valor que recupere **READ**. Fué presentado **SELECT…CASE** como una forma de evaluar una variable caso por caso, y ejecutar un bloque de código diferente dependiendo del caso. En ciertas circunstancias son útiles algunas condiciones opcionales **DO…LOOP**; **DO UNTIL** ***(Condition)*****...LOOP** y **DO...LOOP UNTIL** ***(Condition)*** fueron demostrados como maneras de mantener ejecutando un **DO…LOOP** hasta que se detecte una condicion particular.

Preguntas

1. ¿En qué dirección debe girar la rueda izquierda para hacer que el Boe-Bot vaya al frente? ¿En qué dirección debe girar la rueda derecha?
2. Cuando el Boe-Bot pivotea a la izquierda, ¿qué hacen las ruedas izquierda y derecha? ¿Qué commandos PBASIC necesita para hacer que el Boe-Bot pivotee a la izquierda?
3. Si su Boe-Bot gira ligeramente a la izquierda cuando esta corriendo un programa para hacer que vaya de frente, ¿cómo corrige esto? ¿Qué comando necesita ser ajustado y qué clase de ajuste haría?
4. Si su Boe-Bot viaja a 11 in/s, ¿cuántos pulsos necesitará para hacer que viaje 36 pulgadas?
5. ¿Cuál es la relación entre el argumento ***Counter*** de un ciclo **FOR…NEXT** y el argumento ***Duration*** de un commando **PULSOUT** que hace posible el rampeo?
6. ¿Qué directiva puede usar para pre-guardar valores en la EEPROM del BASIC Stamp antes de correr un programa?
7. ¿Qué comando puede usar para recuperar un valor guardado en la EEPROM y copiarlo a una variable?
8. ¿Qué bloque de código puede usar para seleccionar una variable en particular y evaluarla caso por caso y ejecutar un bloque diferente para cada caso?
9. ¿Cuáles son las diferentes condiciones que pueden ser usados con **DO…LOOP**?

Ejercicios

1. Escriba una rutina que haga retroceder al Boe-Bot por 350 pulsos.
2. Digamos que probó sus servos y descubrió que les toma 48 pulsos hacer una vuelta de 180° con vuelta a la derecha. Con esta información, escriba unas rutinas que hagan que el Boe-Bot ejecute vueltas de 30, 45, y 60 grados.

3. Escriba una rutina que haga que el Boe-Bot vaya de frente, luego rampee para iniciar y terminar una vuelta con pivoteo y luego continúe en línea recta.

Proyectos

1. Es hora de llenar la columna 3 de la Tabla 2-1 en la página 63. Para hacer esto, modifique los argumentos **PULSOUT** ***Duration*** en el programa BoeBotForwardThreeSeconds.bs2 usando cada par de valores de la columna 1. Registre el comportamiento resultante de su Boe-Bot para cada par for each pair en la columna 3. Una vez completada, esta tabla servirá como una guía de referencia cuando diseñe sus propias maniobras personalizadas del Boe-Bot.
2. La Figura 4-9 muestra 2 cursos simples. Escriba un programa que haga que su Boe-Bot navegue sobre cada figura. Asuma que las distancias en línea recta (incluyendo el diámetro del círculo) son ya sea 1 yd o 1 m.

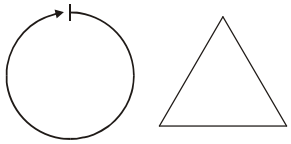


Figura 4-9
Cursos Simple

Soluciones

Q1. La rueda izquierda en sentido izquierdo, la derecha en sentido derecho.
Q2. La rueda derecha gira a la derecha (al frente) y la izquierda no se mueve.

```
PULSOUT 13, 750
PULSOUT 12, 650
```

Q3. Puede desacelerar la rueda derecha para corregir una desviación a la izquierda. El comando **PULSOUT** para la rueda derecha necesita ser ajustado.

```
PULSOUT 12, 650
```

Ajuste el 650 a algo más cercano a 750 para desacelerar la rueda.

```
PULSOUT 12, 663
```

Q4. Dados los siguientes datos, debería tomar 133 pulsos para avanzar 36 pulgadas:
Velocidad del Boe-Bot = 11 in/s
distancia Boe-Bot = 36 in/s

pulsos = (Boe-Bot distancia / Boe-Bot speed) * (40.65 pulsos / s)
= (36 / 11) * (40.65)
= 133.04
= 133

Q5. La variable **`pulseCount`** del ciclo **`FOR…NEXT`** puede ser usada como un ajuste (mas o menos) a 750 (la posición central) en el argumento ***Duration***.


```
FOR pulseCount = 1 to 100
  PULSOUT 13, 750 + pulseCount
  PULSOUT 12, 750 – pulseCount
  PAUSE 20
NEXT
```

Q6. La directiva **`DATA`**.

Q7. El comando **`READ`**

Q8. **`SELECT...CASE...ENDSELECT`**.

Q9. **`UNTIL`** y **`WHILE`**.

E1.
```
FOR counter = 1 to 350     ' Hacia atras
  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20
NEXT
```

E2.
```
FOR counter = 1 to 8       ' Rota a la derecha 30 grados
  PULSOUT 13, 850
  PULSOUT 12, 850
  PAUSE 20
NEXT

FOR counter = 1 to 12      ' Rota a la derecha 45 grados
  PULSOUT 13, 850
  PULSOUT 12, 850
  PAUSE 20
NEXT

FOR counter = 1 to 16      ' Rota a la derecha 60 grados
  PULSOUT 13, 850
  PULSOUT 12, 850
  PAUSE 20
NEXT
```

E3.
```
FOR counter = 1 to 100     ' Al frente
  PULSOUT 13, 850
```

```
  PULSOUT 12, 650
  PAUSE 20
NEXT

FOR counter = 0 TO 30      ' Rampeo de vuelta con pivoteo
  PULSOUT 13, 750 + counter
  PULSOUT 12, 750
  PAUSE 20
NEXT

FOR counter = 30 TO 0
  PULSOUT 13, 750 + counter
  PULSOUT 12, 750
  PAUSE 20
NEXT

FOR counter = 1 to 100           ' Al frente
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
```

P1.

P13	P12	Descripción	Comportamiento
850	650	Velocidad plena: P13 CCW, P12 CW	Al frente
650	850	Velocidad plena: P13 CW, P12 CCW	Atrás
850	850	Velocidad plena: P13 CCW, P12 CCW	Rotación a la derecha
650	650	Velocidad plena: P13 CW, P12 CW	Rotación a la izq.
750	850	P13 Detenido, P12 CCW Velocidad plena	Pivotea atras izq
650	750	P13 CW Velocidad plena, P12 Detenido	Pivotea atras der
750	750	P13 Detenido, P12 Detenido	Detenido
760	740	P13 CCW Lento, P12 CW Lento	Al frente lento
770	730	P13 CCW Med, P12 CW Med	Al frente media vel
850	700	P13 CCW Velocidad plena, P12 CW Medio	Vuelta a la der.
800	650	P13 CCW Medio, P12 CW Velocidad plena	Vuelta a la izq.

P2. El círculo puede ser implementado volteando a la derecha contínuamente. A través de prueba y error y usando un palo de un metro o una yarda podrá llegar al valor de **PULSOUT** correcto. Para un círculo con diámetro de una yarda:

```
' Robotica con el Boe-Bot - Capítulo 4 - Circle.bs2
```

```
' El Boe-Bot navega un circulo de diametro de 1 yarda.

'{$STAMP BS2}
'{$PBASIC 2.5}
DEBUG "Program running!"

pulseCount     VAR     Word        ' Cuenta pulsos a los servos

FREQOUT 4, 2000, 3000              ' Señal de programa en inicio/reset.

' -----[ Rutina Principal ]------------------------------------------
Main:
DO
  PULSOUT 13, 850                   ' Giro a la derecha
  PULSOUT 12, 716
  PAUSE 20
LOOP
```


Para hacer el triángulo, primero calcule el número de puslso requeridos para una línea recta de 1 metro o 1 yarda, como en la pregunta 4. Luego afine sus distancias para empatar su Boe-Bot y la superficie particular. Para el patrón de triángulo, el Boe-Bot debe viajar 1 metro/yarda al frente y luego hacer un giro de 120 grados. Esto debe ser repetido 3 veces para los 3 lados del triángulo. Quizá tenga que ajustar ***EndValue*** de **`pulseCount`** en la subrutina **`Right_Rotate120`** para obtener un giro preciso de 120 grados.

```
' Robotica con el Boe-Bot - Capítulo 4 - Triangle.bs2
' El Boe-Bot navega en forma triangular con lados de 1 yarda.
' Va de frente, luego gira 120 degrees.  Repite 3 veces.

'{$STAMP BS2}
'{$PBASIC 2.5}
DEBUG "Program running!"

counter        VAR     Nib          ' Triangulo tiene 3 lados
pulseCount     VAR     Word         ' Cuenta pulsos a los servos

FREQOUT 4, 2000, 3000               ' Señal de programa en incio/reset.

Main:
  FOR counter = 1 TO 3              ' repite 3 veces para el triangulo
    GOSUB Forward
    GOSUB Right_Rotate120
  NEXT
  END

Forward:
  FOR pulseCount = 1 TO 163          ' Al frente 1 yarda
    PULSOUT 13, 850
```

```
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Right_Rotate120:
  FOR pulseCount = 1 TO 21           ' Gira a la derecha 120 grados
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```

Capítulo 5: Navegación Táctil con Filamentos

Muchos tipos de maquinaria robótica confían en una variedad de interruptores táctiles. Por ejemplo, un interruptor táctil puede detectar cuando un brazo robótico ha encontrado un objeto. El robot puede ser programado para recoger el objeto y colocarlo en otro lado. Las fábricas usan interruptores táctiles para contar objetos en una línea de producción y también para alinear objetos durante sus procesos. En todas estas instancias, los interruptores proveen entradas que dictan alguna otra forma de salida programada. Las entradas son electrónicamente monitoreadas por el producto, siendo este un robot, una calculadora o una línea de producción. En base al estado de los interruptores, el brazo robótico toma un objeto, o la calculadora despliega actualizaciones, o la linea de producción de la fábrica reacciona con motores o servos para guiar los productos.

En este Capítulo construirá interruptores táctiles llamados filamentos y los probará en su Boe-Bot. Luego programará al Boe-Bot para monitorear el estado de estos y para decidir qué hacer cuando encuentre un obstáculo. El fin es la navegación autónoma por palpado.

NAVEGACIÓN TÁCTIL

Los filamentos o bigotes ("whiskers") son así llamados porque es lo que estos interruptores de choque parecen, aunque algunos argumentan que parecen más antenas. Estos filamentos están mostrados en el Boe-Bot en la Figura 5-1. Los filamentos le dan al Boe-Bot la habilidad de sensar el mundo a su alrededor palpando, muy parecido a las antenas de una hormiga o los bigotes de un gato. Las actividades en este Capítulo usan los filamentos en sí mismos, pero pueden ser combinados con otros sensores de los que aprenderán en siguientes Capítulos para incrementar la funcionalidad de su Boe-Bot.

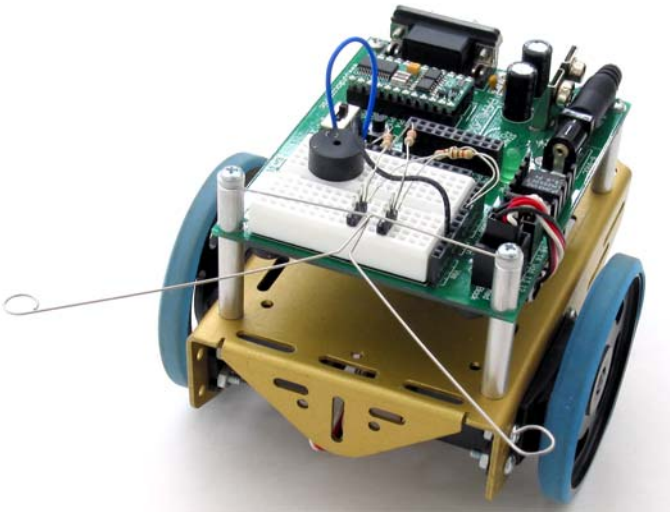


Figura 5-1
Boe-Bot con Filamentos

ACTIVIDAD #1: CONSTRUYENDO Y PROBANDO LOS FILAMENTOS

Antes de proceder con los programas que harán que el Boe-Bot navege en base a lo que pueda tocar, es esencial construir y probar los filamentos primero. Esta Actividad le guiará a través de la construcción y prueba de los filamentos.

Circuito con Filamentos y Ensamble

- ✓ Reúna el hardware de los filamentos mostrados en la Figura 5-2.
- ✓ Desconecte la energía en su tarjeta y en los servos.

Lista de partes

(2) Alambres filamentos
(2) Tornillos Phillips 7/8″ cabeza plana 4-40
(2) Espaciadores de ½″
(2) Rondanas de Nylon #4
(2) Conectores de 3 pines
(2) Resistencias de 220 Ω (rojo-rojo-cafe)
(2) Resistencias de 10 kΩ (cafe-negro-naranja)

Figura 5-2
Hardware de los filamentos

Construyendo los Filamentos

- ✓ Quite los dos tornillos frontales que sostienen su tarjeta a los soportes frontales.
- ✓ Refiérase a la Figura 5-3 mientras que sigue las Instrucciones faltantes.
- ✓ Coloque una rondana de nylon luego un espaciador en cada tornillo de 7/8″.
- ✓ Coloque los tornillos en su tarjeta y dentro de los soportes, pero aún no los apriete completamente.
- ✓ Deslice los extremos con gancho de los alambres alrededor de los tornillos, uno sobre una rondana y el otro bajo la otra rondana, de tal forma que se entrecrucen sin tocarse.
- ✓ Apriete los tornillos a los soportes.

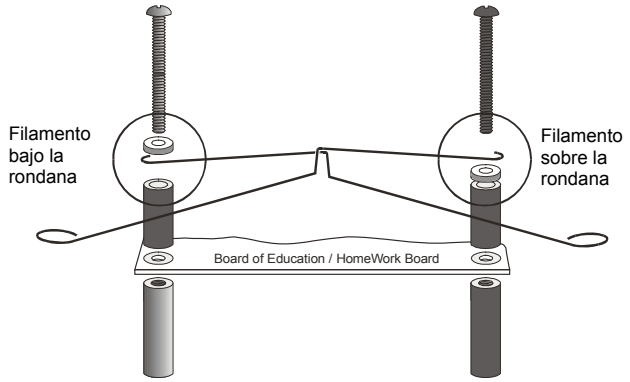


Figura 5-3
Montando los filaments

El siguiente paso es agregar el circuito para filamentos mostrado en la Figura 5-4 a los circuitos de piezoparlante y servo que construyó y probó en el Capítulo 2 y Capítulo 3.

- ✓ Si tiene un Board of Education, construya el circuito para filamentos de la Figura 5-4 usando el diagrama (Figura 5-5) en la página 147 como referencia.
- ✓ Si tiene un HomeWork Board, construya el circuito para filamentos de la Figura 5-4 usando el diagrama (Figura 5-56) en la página 148 como referencia.
- ✓ Asegúrese de ajustar cada filamento de tal forma que estén proximos a tocar, pero no tocan, los conectores de 3 pines en la tableta. Una distancia de aproximádamente 1/8″ (3 mm) es un punto de inicio recomendado.

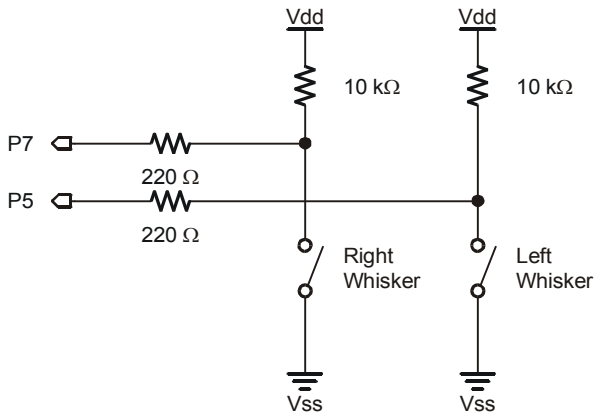
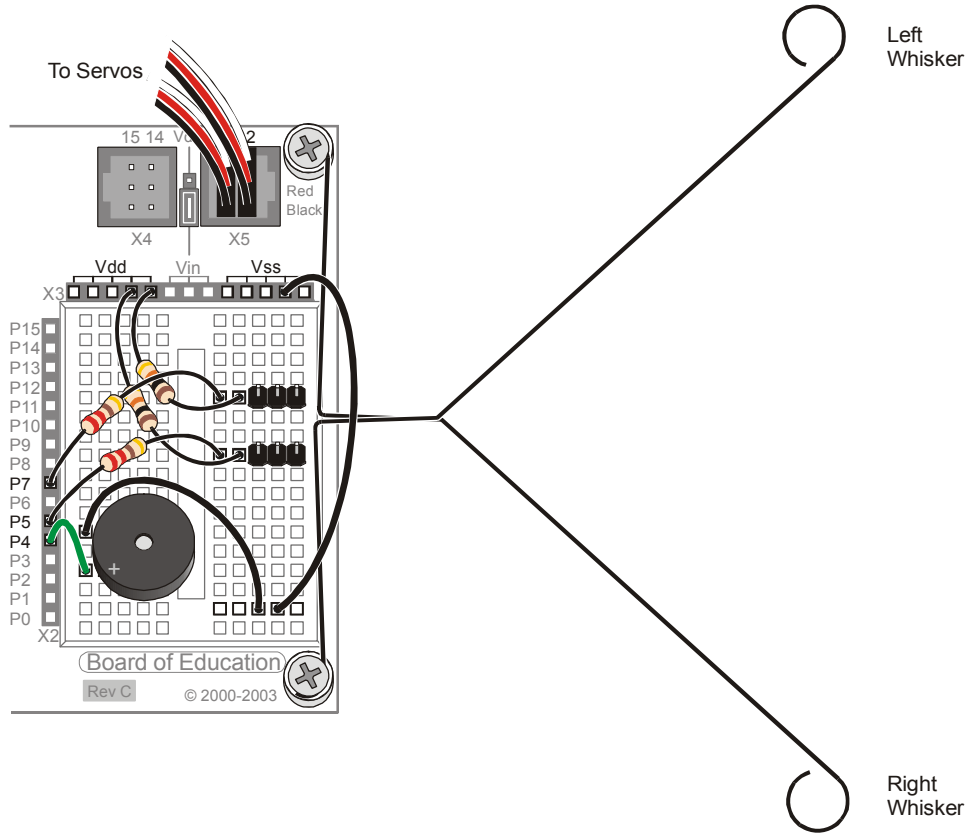


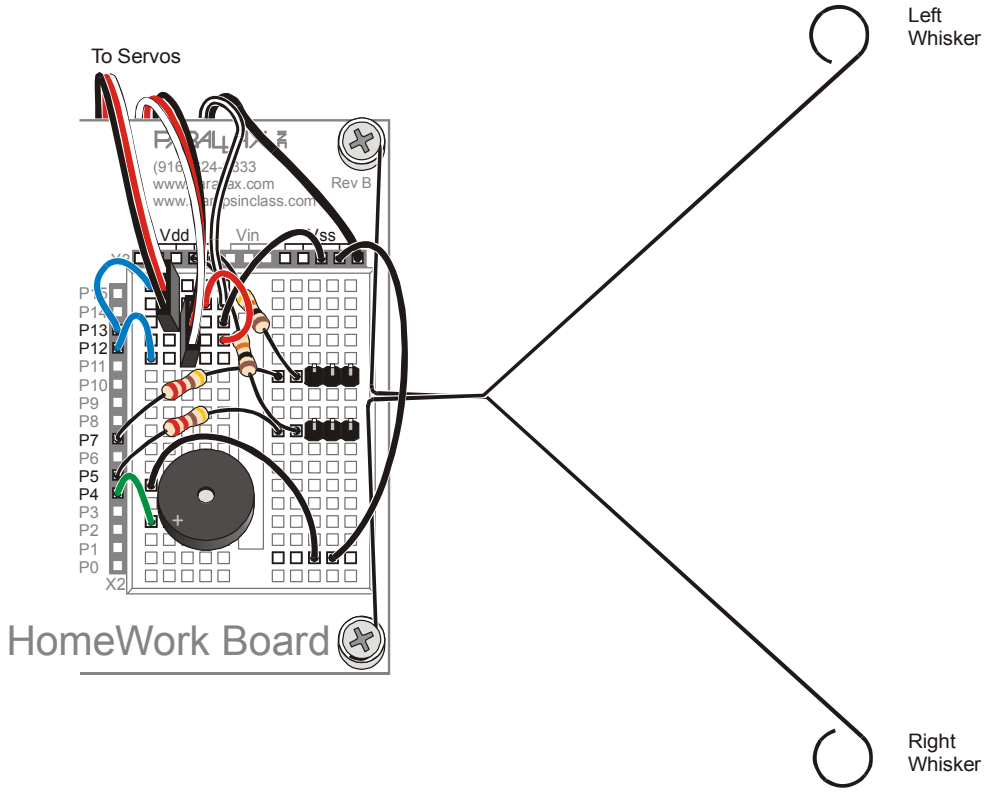
Figura 5-4
Esquemático para filamentos

Figura 5-5: Diagrama de alambrado de filamentos para el Board of Education



Use las resistencias de 220 Ω (rojo-rojo-café) para conectar P5 y P7 a sus correspondientes conectores de 3 pines. Use las resistencias de 10 kΩ (cafe-negro-naranja) para conectar Vdd a cada conector de 3 pines.

Figura 5-6: Diagrama de alambrado de filamentos para el HomeWork Board



Use las resistencias de 220 Ω (rojo-rojo-café) para conectar P5 y P7 a sus correspondientes conectores de 3 pines. Use las resistencias de 10 kΩ (cafe-negro-naranja) para conectar Vdd a cada conector de 3 pines.

Probando los Filamentos

Revise de nuevo el esquemático de la Figura 5-7. Cada filamento es una extensón mecánica y la tierra eléctrica de un circuito con un interruptor normalmente abierto de 1 polo y 1 tiro. La razón por la que los filamentos están conectados a tierra (Vss) es porque los 4 barrenos en los 4 extremos externos de la tarjeta están conectados a Vss. Esto es cierto tanto para el Board of Education y el BASIC Stamp HomeWork Board. Los soportes de metal los tornillos proveen la conexión eléctrica para cada filamento.

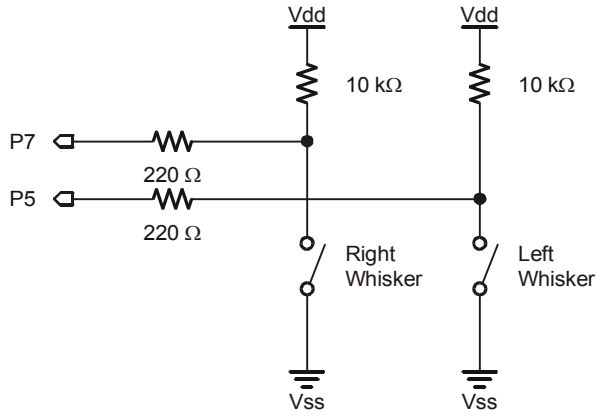


Figura 5-7
Un Segundo vistazo al esquemático para filamentos.

El BASIC Stamp puede ser programado para detectar cuando se presiona un filamento. Los pines de I/O conectados a cada circuito-interruptor monitorean el voltaje en la Resistencia de 10 kΩ. La Figura 5-8 ilustra esto. Cuando un determinado filament no es presionado, el voltaje en el pin I/O conectado a ese filamento es 5 V. Cuando un filament es presionado, la línea de I/O es puesta a tierra (Vss), y entonces la línea I/O vé 0 V.

Todos los pines I/O toman como valor por defecto “input” cada vez que inicia un programa PBASIC. Esto significa que los pines I/O conectados a los filamentos funcionan como entradas automáticamente. Como entrada, un pin I/O conectado a un circuito de filamento causará que su registro de entrada guarde un 1 si el voltaje es 5 V (filamento no presionado) o un 0 si el voltaje es 0 V (filamento presionado). La Terminal de Depuración puede usarse para desplegar estos valores.

¿Cómo hacer que el BASIC Stamp le diga si está leyendo un 1 o un 0?

Ya que el circuito está conectado a P7, el valor 1 o 0 aparecerá en una variable llamada **IN7**. **IN7** es llamadaa un registro de entrada. Las variables de registro de entrada son interconstruídas y no tienen que ser declaradas al inicio del programa. Puede ver los valores que esta variable guarda usando el comando **DEBUG BIN1 IN7**. **BIN1** es un formateador que le indica a la Terminal de Depuración desplegar un dígito binario (1 o 0).

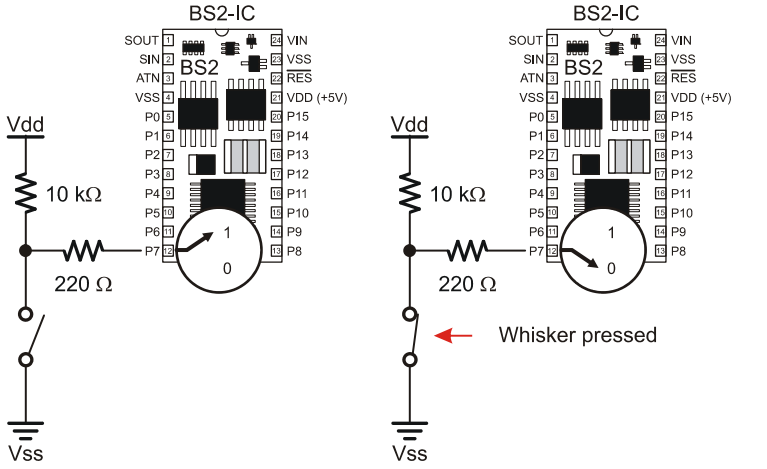


Figura 5-8
Detectando Contactos Eléctricos

Programa Ejemplo: TestWhiskers.bs2

El siguiente programa ejemplo esta diseñado para asegurarse que los filamentos funcionan adecuadamente. Al desplegar los dígitos binarios guardados en los registros de entrada de P7 y P5 (**IN7** y **IN5**), el programa le mostrará si BASIC Stamp ha detectado contacto con un filamento. Si el valor guardado en un registro de entrada determinado es 1 el filamento no está presionado. Si es 0 el filamento está presionado.

- ✓ Reconecte la energía a su tarjeta y a sus servos.
- ✓ Introduzca, salve y corra TestWhiskers.bs2.
- ✓ Este programa hace uso de la Terminal de Depuración, mantenga el cable de programación conectado al BASIC Stamp mientras corre el programa.

```
' Robotica con el Boe-Bot - TestWhiskers.bs2
' Despliega lo que sensan los pines I/O conectados a los filamentos.
' {$STAMP BS2}                              ' directiva Stamp.
' {$PBASIC 2.5}                             ' directiva PBASIC.
```

```
DEBUG "WHISKER STATES", CR,
      "Left     Right", CR,
      "------   ------"

DO
  DEBUG CRSRXY, 0, 3,
        "P5 = ", BIN1 IN5,
        "   P7 = ", BIN1 IN7
  PAUSE 50
LOOP
```


- ✓ Note los valores desplegados en la Terminal de Depuración; debe desplegar que tanto P7 como P5 equivalen a 1.
- ✓ Revise la Figura 5-5 en la página 147 (o la Figura 5-6 en la página 148) para saber cual es el filamento izquierdo y cual el derecho
- ✓ Presione el filamento derecho hacia su conector de 3 pines y note los valores desplegados en la Terminal de Depuración. Debe leerse:
  P5 = 1 P7 = 0
- ✓ Presione el filamento izquierdo hacia su conector de 3 pines y note los valores desplegados en la Terminal de Depuración. Debe leerse:
  P5 = 0 P7 = 1
- ✓ Presione ambos filamentos contra ambos conectores de 3 pines. Debe leerse:
  P5 = 0 P7 = 0
- ✓ Si los filamentos pasaron todas estas pruebas, está listo para continuar; de lo contrario, revise errores en su programa y sus circuitos.

¿Qué es un Cursor? ¿Qué es CRSRXY?

Según eldiccionario Merriam-Webster, un cursor es: “Un elemento móvil usado para marcar una posición como… una pista visual en un display de video que indica posición.” Al mover su ratón, el apuntador que mueve en su pantalla es un cursor. El de la Terminal de Depuración es algo diferente porque no parpadea ni hace algo para indicar su posición. Pero dondequiera que se encuentre este, es allí donde se imprimirá el siguiente caracter.

CRSRXY es un formateador que le permite arreglar la información que su programa manda a la Terminal de Depuración. El formateador **`CRSRXY 0, 3,`** en el comando:

```
DEBUG CRSRXY, 0, 3,
        "P5 = ", BIN1 IN5,
        "   P7 = ", BIN1 IN7
```

...coloca el cursor en la columna 0, renglón 3 en la Terminal de Depuración. Esto hace que se despliegue debajo del encabezado “Whisker States”. Cada vez que pasa por el ciclo, los nuevos valores se sobreescriben a los viejos porque el cursor regresa al mismo lugar.

ACTIVIDAD #2: PROBANDO EN CAMPO LOS FILAMENTOS

Asuma que puede tener que probar los filamentos en un momento posterior lejos de su computadora. Como la Terminal de Depuración no estará disponible, ¿qué puede hacer? Una solución sería programar el BASIC Stamp para que envíe una señal de salida que corresponda a la señal de entrada que está recibiendo. Esto puede hacerse con un par de circuitos LED y un programa que los encienda/apague según la entrada de los filamentos.

Lista de partes:

(2) Resistencias, 220 Ω (rojo-rojo-cafe)
(2) LEDs, rojos

Construyendo los circuitos LED de prueba de Filamentos

- ✓ Desconecte la energía de su tarjeta y servos.
- ✓ Si tiene un Board of Education, agruege el circuito mostrado en la Figura 5-9 con la ayuda del diagrama en la Figura 5-10 (página 153).
- ✓ Si tiene un HomeWork Board, agruege el circuito mostrado en la Figura 5-9 con la ayuda del diagrama en la Figura 5-11 (página 154).

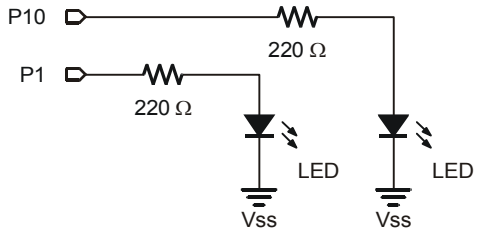


Figura 5-9
Esquemático LED de prueba de Filamentos

Agregue estos circuitos LED.

Recuerde que un LED es una válvula de corriente de una sola dirección.

Si se conecta en reversa, no dejará pasar la corriente y no emitirá luz. Para que el LED emita luz cuando el BASIC Stamp envía una señal alta, el ánodo del LED debe estar conectado a la resistencia de 220 Ω y su cátodo debe estar conectado a Vss. Véa la Figura 5-10 o la Figura 5-11.

Figura 5-10 Diagrama de conexiones para Filamentos mas LEDs para el Board of Education

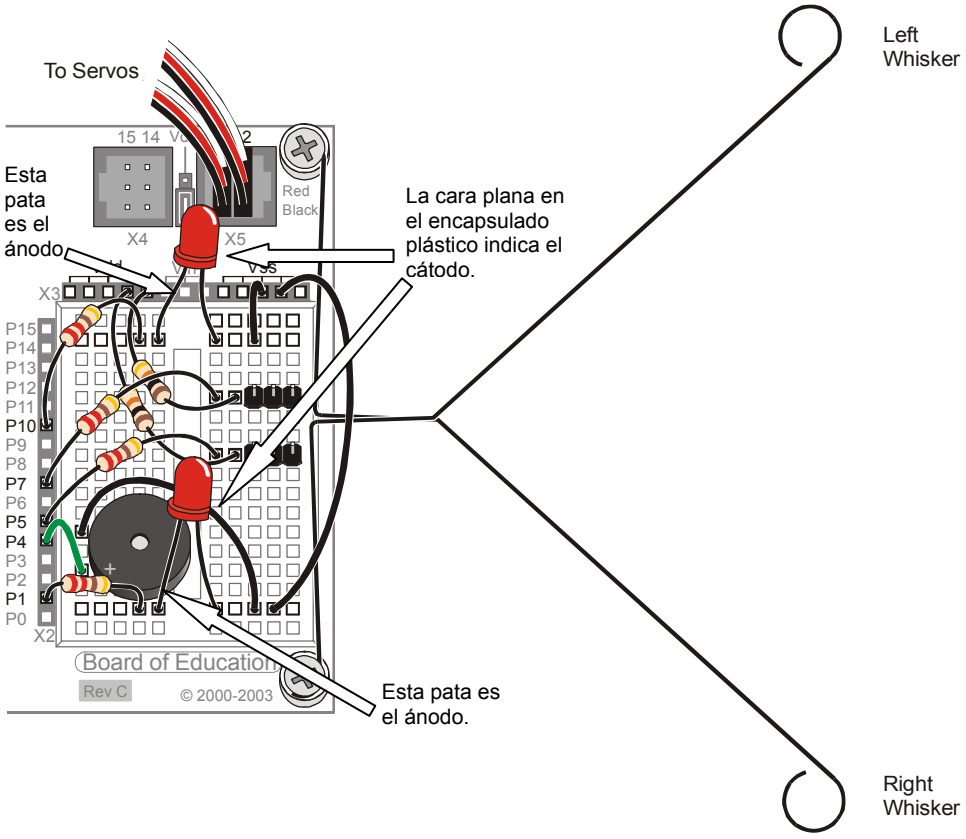
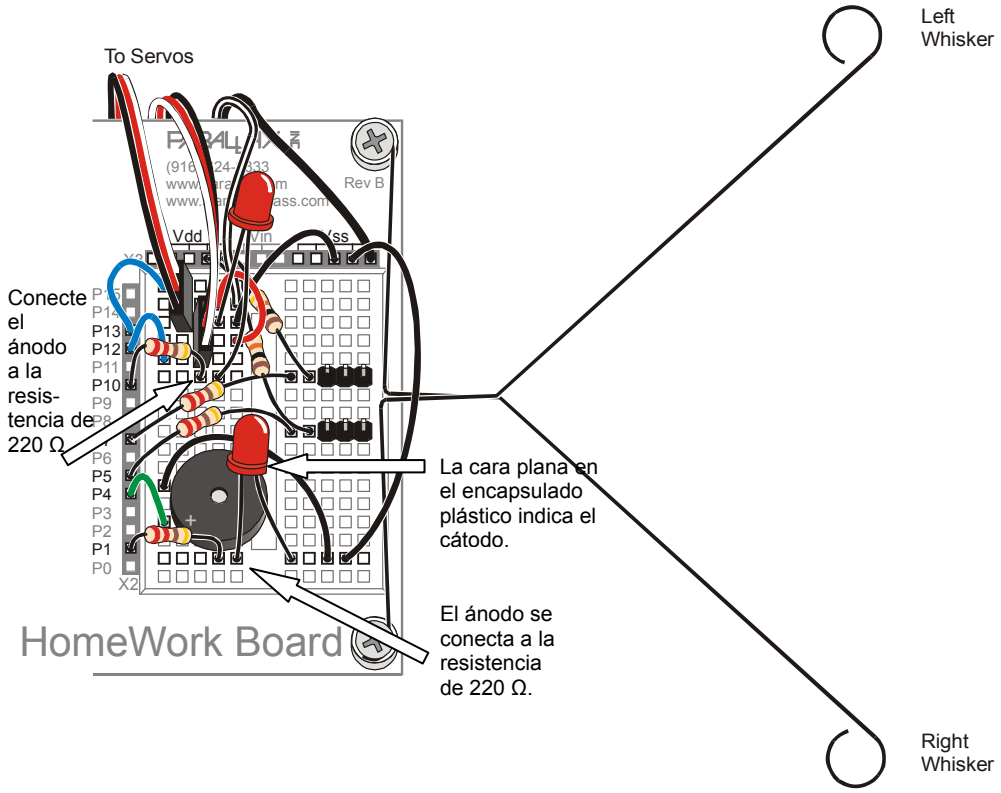


Figura 5-11 Diagrama de conexiones para Filamentos mas LEDs para el HomeWork Board



Programando los circuitos LED de prueba de Filamentos

- ✓ Reconecte la energía a su tarjeta.
- ✓ Salve TestWhiskers.bs2 as TestWhiskersWithLeds.bs2.
- ✓ Inserte las siguientes declaraciones **`IF...THEN`** entre los commandos **`PAUSE 50`** y **`LOOP`**.

```
IF (IN7 = 0) THEN
  HIGH 1
ELSE
  LOW 1
ENDIF
```

```
IF (IN5 = 0) THEN
  HIGH 10
ELSE
  LOW 10
ENDIF
```

Estos son llamadas declarciones **IF…THEN**, y serán mas profundamente presentados en la siguiente Actividad. Estas declaraciones son usadas para tomar decisiones en PBASIC. Ue enciende el LED cuando el filamento conectado a P7 es presionado (**IN7** = 0). La porción **ELSE** de la declaración hace que P1 vaya a bajo, lo que apaga el LED cuando el filamento no es presionado. La segunda declaración **IF…THEN** hace lo mismo para el filamento conectado a P5 y el LED conectado a P10.

- ✓ Corra TestWhiskersWithLeds.bs2.
- ✓ Pruébelo presionando suavemente los filamentos. Los LEDs rojos deben encender cuando cada filamento haga contacto con cada conector de 3 pines.

ACTIVIDAD #3: NAVEGACIÓN CON FILAMENTOS

En la Actividad previa, el BASIC Stamp fue programado para detectar si un filamento fue presionado. En esta, el BASIC Stamp sera programado para tomar ventaja de esta información para guiar al Boe-Bot. Cuando esté avanzando y un filamento sea presionado, querrá decir que el Boe-Bot choca contra algo. Un programa de navegación necesita atender esta entrada, decidir qué significa y llamar a un conjunto de maniobras que hagan que el Boe-Bot se recupere del obstáculo, gire y vaya en otra dirección.

Programando el Boe-Bot para Navegar en base a la entrada de sus filamentos

El siguiente programa hace que el Boe-Bot vaya al frente hasta que encuentre un obstáculo. En este caso, el Boe-Bot sabe cuando encuentra un obstáculo al chocarlo con uno o ambos filamentos. Tan pronto como el obstáculo es detectado por los filamentos, las rutinas y subrutinas de navegación desarrolladas en el Capítulo 4 harán que el Boe-Bot retroceda y gira. Luego, el Boe-Bot retoma su movimiento al frente hasta que golpea otro obstáculo.

Para hacer esto, el Boe-Bot necesita ser programado para tomar decisiones. PBASIC tiene un comando llamado declaración **IF…THEN** que toma decisiones. La sintáxis de la declaración **IF…THEN** es:

IF *(condition)* THEN…{ELSEIF *(condition)*}…{ELSE}…ENDIF

Los "…" significan que puede colocar un bloque de código (uno o más comandos) entre las palabras clave. El siguiente programa ejemplo toma decisiones basado en las entradas de los filamentos y luego llama subrutinas para hacer que el Boe-Bot tome acción. Las subrutinas son similares a las desarrolló en el Capítulo 4. Hé aquí cómo se usa **IF…THEN**.

```
IF (IN5  = 0) y (IN7 = 0) THEN
  GOSUB Back_Up          ' Ambos filamentos detectan obstaculo,
  GOSUB Turn_Left        ' Retrocede y vuelta U(2 izquierdas)
  GOSUB Turn_Left
ELSEIF (IN5  = 0) THEN   ' El filamento izquierdo hace contacto
  GOSUB Back_Up          ' Retrocede y gira a la derecha
  GOSUB Turn_Right
ELSEIF (IN7  = 0) THEN   ' El filamento derecho hace contacto
  GOSUB Back_Up          ' Retrocede y gira a la izquierda
  GOSUB Turn_Left
ELSE                     ' Ambos filamentos en 1, no contactos
  GOSUB Forward_Pulse    ' Aplica un pulso al frente y
ENDIF                    ' checa otra vez
```

Programa Ejemplo: RoamingWithWhiskers.bs2

Este programa demuestra una forma de evaluar las entradas de los filamentos y decidir cual subrutina de navegación llamar usando **IF…THEN**.

- ✓ Reconecte la energía a su tarjeta y servos.
- ✓ Introduzca, salve y corra RoamingWithWhiskers.bs2.
- ✓ Intente dejar correr al Boe-Bot. Cando contacta los obstáculos en su camino retrocede, gira y va en una nueva dirección.

```
' -----[ Titulo ]-------------------------------------------------------------
' Robotica con el Boe-Bot - RoamingWithWhiskers.bs2
' Boe-Bot usa filamentos para detectar objetos y navegar ardedor de estos.

' {$STAMP BS2}                              ' Directiva Stamp.
' {$PBASIC 2.5}                             ' Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Variables ]-----------------------------------------------------------

pulseCount     VAR     Byte                ' Contador de ciclo FOR...NEXT.

' -----[ Inicializacion ]------------------------------------------------------

FREQOUT 4, 2000, 3000                    ' Señal de programa en inicio/reset.
```

```
' -----[ Rutina Principal]-------------------------------------------------

DO
  IF (IN5  = 0) y (IN7 = 0) THEN         ' Ambos filamentos detectan obstaculo
    GOSUB Back_Up                        ' Retrocede y vuelta en U(2 izqs.)
    GOSUB Turn_Left
    GOSUB Turn_Left
  ELSEIF (IN5  = 0) THEN                 ' El filamento izquierdo contacta
    GOSUB Back_Up                        ' Retrocede y vuelta a la derecha
    GOSUB Turn_Right
  ELSEIF (IN7  = 0) THEN                 ' El filamento derecho contacta
    GOSUB Back_Up                        ' Retrocede y vuelta a la izquierda
    GOSUB Turn_Left
  ELSE                                   ' Ambos filamentos en 1, no contacto
    GOSUB Forward_Pulse                  ' Aplica un pulso al frente
  ENDIF                                  ' y checa otra vez
LOOP

' -----[ subrutinas ]-------------------------------------------------------

Forward_Pulse:                           ' Envia un solo pulso al frente.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                               ' Vuelta a la izq, aprox 90-grados.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20               ' Vuelta a la der, aprox 90-grados.
    PULSOUT 13, 850
    PULSOUT 12, 850

    PAUSE 20
  NEXT
  RETURN

Back_Up:                                 ' Retrocede.
  FOR pulseCount = 0 TO 40
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```

Cómo funciona la navegación con Filamentos

Las declaraciones **IF...THEN** en la sección de la Rutina Principal primero checa si en los filamentos hay algun estado que requiera atención. Si ambos filamentos están presionados (**IN5** = 0 y **IN7** = 0), se ejecuta una vuelta en U llamando la subrutina **Back_Up** seguida por dos llamadas seguidas de la subrutina **Turn_Left**. Si solo el filamento izquierdo es presionado (**IN5** = 0), entonces el programa llama la subrutina **Back_Up** seguida por la subrutina **Turn_Right**. Si el filamento derecho es presionado (**IN7** = 0), la subrutina **Back_Up** es llamada, seguida por la subrutina **Turn_Left**. La única posible combinación no cubierta es si ningún filamento es presionado (**IN5** = 1 y **IN7** = 1). El comando **ELSE** llama la subrutina **Forward_Pulse** en este caso.

```
IF (IN5  = 0) y (IN7 = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
  GOSUB Turn_Left
ELSEIF (IN5  = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Right
ELSEIF (IN7  = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
ELSE
  GOSUB Forward_Pulse
ENDIF
```

Las subrutinas **Turn_Left**, **Turn_Right**, y **Back_Up** deben verse bastante familiares, pero la subrutina **Forward_Pulse** tiene un cambio. Solo envía un pulso, luego regresa. Esto es realmente importante, porque significa que el Boe-Bot puede checar sus filamentos entre cada pulso al frente. Esto significa que el Boe-Bot busca obstáculos aproximádamente 40 vecer por Segundo al ir avanzando al frente.

```
Forward_Pulse:
  PULSOUT 12,650
  PULSOUT 13,850
  PAUSE 20
  RETURN
```

Puesto que cada pulso al frente a velocidad plena hace que el Boe-Bot avance aproximadamente medio centímetro, es en verdad una buena idea enviar solo un pulso, luego regresar y revisar los filamentos nuevamente. Puesto que la declaración **IF…THEN** esta dentro de un **DO…LOOP**, cada vez que el programa regresa de un **Forward_Pulse**,

llega a **LOOP**, que envía el programa de regreso hasta **DO**. ¿Qué pasa entonces? La declaración **IF…THEN** checa nuevamente los filamentos otra vez desde el inicio.

Su Turno

Los argumentos ***EndValue*** del ciclo **FOR...NEXT** en las rutinas **Back_Right** y **Back_Left** pueden ser ajustados para mayor o menor giro y la rutina **Back_Up** puede tener su ***EndValue*** ajustado para regresar menos en caso de navegación en espacios más pequeños.

- ✓ Experimente con los argumentos ***EndValue*** del ciclo **FOR...NEXT** en las rutinas de navegación en RoamingWithWhiskers.bs2.

También puede modicar sus declaraciones **IF…THEN** para hacer que los indicadores LED de la Actividad previa indiquen la maniobra en la que se encuentra el Boe-Bot agregando unos comandos **HIGH** y **LOW** para control los circuitos LED. He aquí un ejemplo.

```
IF (IN5  = 0) y (IN7 = 0) THEN
  HIGH 10
  HIGH 1
  GOSUB Back_Up
  GOSUB Turn_Left
  GOSUB Turn_Left
ELSEIF (IN5  = 0) THEN
  HIGH 10
  GOSUB Back_Up
  GOSUB Turn_Right
ELSEIF (IN7  = 0) THEN
  HIGH 1
  GOSUB Back_Up
  GOSUB Turn_Left
ELSE
  LOW 10
  LOW 1
  GOSUB Forward_Pulse
ENDIF
```

- ✓ Modifique la declaración **IF…THEN** en RoamingWithWhiskers.bs2 para hacer que el Boe-Bot indique su maniobra usando sus indicadores LED.

ACTIVIDAD #4: INTELIGENCIA ARTIFICIAL Y DECIDIENDO CUANDO ESTÁ ATORADO

Quizá haya notado que el Boe-Bot se queda atorado en las esquinas. Conforme el Boe-Bot entra a la esquina, su filamento toca la pared a la izquierda y gira a la derecha. Cuando el Boe-Bot se mueve hacia al frente nuevamente, su filamento derecho choca con la pared a la derecha y gira a la izquierda. Luego gira y choca con la pared izquierda nuevamente y así sucesivamente, hasta que alguien lo rescata de su predicamento.

Programando para Escapar a las Esquinas

RoamingWithWhiskers.bs2 puede ser modificado para detectar este problema y actuar en consecuencia. El truco es contar las veces que los filamentos alternativamente hacen contacto. Una cosa importante es que el programa tiene que recordar en qué estado estaba cada filamento durante el contacto previo. Tiene que compararlo contra los estados actuales de los filamentos en el contacto presente. Si son opuestos hay que sumar uno al contador. Si el contador sobrepasa un valor de umbral que usted haya determinado, entonces es hora de dar vuelta en U y reiniciar el contador de esta alternativa.

El siguiente programa también recae en el hecho de que puede "anidar" las declaraciones **IF…THEN**. En otras palabras, el programa checa una condición y si esa condición se cumple, checa otra condición dentro de la primera. He aquí un ejemplo de pseudocódigo de cómo puede ser esto usado.

```
IF condition1 THEN
  Comandos for condition1
  IF condition2 THEN
    Comandos for both condition2 y condition1
  ELSE
    Comandos for condition1 but not condition2
  ENDIF
  ELSE
    Comandos for not condition1
ENDIF
```

En el siguiente programa se presenta un ejemplo de declaraciones **IF…THEN** anidadas en la rutina que detecta contactos de filamentos alternados y consecutivos.

Programa Ejemplo: EscapingCorners.bs2

Este programa causará que su Boe-Bot ejecute una vuelta en U a la cuarta o quinta esquina alternada, dependiendo de cual filamento haya sido presionado primero.

- ✓ Introduzca, salve y corra EscapingCorners.bs2.
- ✓ Pruébelo presionando filamentos alternados conforme avanza el Boe-Bot. Dependiendo de con cual filamento haya iniciado, el Boe-Bot deberá ejecutar su maniobra de vuelta en U luego de 4 o 5 contactos de filamento consecutivos.


```
' -----[ Titulo ]-----------------------------------------------------------
' Robotica con el Boe-Bot - EscapingCorners.bs2
' Boe-Bot navega fuera de esquinas detectando presión de filamentos alternada.
' {$STAMP BS2}                            ' Directiva Stamp.
' {$PBASIC 2.5}                           ' Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Variables ]--------------------------------------------------------

pulseCount     VAR     Byte               ' Contador de cicloFOR...NEXT.
counter        VAR     Nib                ' Cuenta contactos alternados.
old7           VAR     Bit                ' Guarda IN7 previo.
old5           VAR     Bit                ' Guarda IN5 previo.

' -----[ Inicializacion ]---------------------------------------------------

FREQOUT 4, 2000, 3000                     ' Señal de programa en inicio/reset.
counter = 1                               ' Inicia cuenta de esquina alternada
old7 = 0                                  ' Establece valores viejos.
old5 = 1

' -----[ Rutina principal ]-------------------------------------------------

DO

' ---  Detecta Esquinas Alternadas Consecutivas -----------------------
' Vea la seccion "How EscapingCorners.bs2 Works" que sigue a este programa.

  IF (IN7 <> IN5) THEN                      ' Uno u otro es presionado.
    IF (old7 <> IN7) y (old5 <> IN5) THEN  ' Diferente al anterior.
      counter = counter + 1                 ' Cuenta filamento alternado + 1.
      old7 = IN7                            ' Registra esta presion en filam.
      old5 = IN5                            ' para la siguiente comparacion.
      IF (counter > 4) THEN                 ' Si cuenta filamento alternado=4
        counter = 1                         ' reinicia cuenta filamento
        GOSUB Back_Up                       ' y ejecuta vuelta en U.
        GOSUB Turn_Left
        GOSUB Turn_Left
      ENDIF                                 ' ENDIF contador > 4.
    ELSE                                    ' ELSE (old7=IN7) or (old5=IN5),
      counter = 1                           ' no alternado, reinicia cuenta.
    ENDIF                                   ' ENDIF (old7<>IN7) y
                                            ' (old5<>IN5).
  ENDIF                                     ' ENDIF (IN7<>IN5).
```

```
' ---  Misma rutina de navegación de RoamingWithWhiskers.bs2 -----------------

  IF (IN5  = 0) y (IN7 = 0) THEN      ' Ambos filamentos detectan obstaculo
    GOSUB Back_Up                        ' Retrocede y vuelta en U(2 izqs.)
    GOSUB Turn_Left
    GOSUB Turn_Left
  ELSEIF (IN5  = 0) THEN                 ' El filamento izquierdo contacta
    GOSUB Back_Up                        ' Retrocede y vuelta a la derecha
GOSUB Turn_Right
  ELSEIF (IN7  = 0) THEN                 ' El filamento derecho contacta
    GOSUB Back_Up                        ' Retrocede y vuelta a la izquierda
    GOSUB Turn_Left
  ELSE                                   ' Ambos filamentos en 1, no contacto
    GOSUB Forward_Pulse                  ' Aplica un pulso al frente
  ENDIF                                  ' y checa otra vez

LOOP

' -----[ subrutinas ]-----------------------------------------------------

Forward_Pulse:                           ' Envia un solo pulso al frente.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                               ' Vuelta a la izq, aprox 90-grados.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20               ' Vuelta a la der, aprox 90-grados.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN

Back_Up:                                 ' Retrocede.
  FOR pulseCount = 0 TO 40
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```

Cómo trabaja EscapingCorners.bs2

Puesto que este programa es una versión modificada de RoamingWithWhiskers.bs2, solo se discutirán las nuevas características relacionadas a la detección y escape de esquinas.

Se crean 3 variables extra para detectar una esquina. La variable nibble **counter** puede guarder un valor entre 0 y 15. Puesto que nuestro valor objetivo para detector una esquina es 4, el tamaño de la variable es razonable. Recuerde que una variable bit puede guarder un solo bit, sea 1 o 0. Las siguientes 2 variables (**old7** y **old5**) son ambas variables bit. También son del tamaño correcto para este trabajo puesto que se usan para guardar valores anteriores de **IN7** y **IN5**, que también son variables bit.


```
counter         VAR     Nib
old7            VAR     Bit
old5            VAR     Bit
```

Estas variables tienen que ser inicializadas (darles valores iniciales). Para facilidad de lectura del programa, **counter** is puesto en 1, y cuando llega a 4 por el hecho de que el Boe-Bot esta atorado en una esquina, se reestablece a 1. Las variables **old7** y **old5** tienen que ser establecidad para que parezca como que uno de los 2 filamentos fue presionado en un momento antes de que el programa iniciara. Esto tiene que ser hecho porque la rutina para detector esquinas alternas compara un patrón alternante, ya sea (**IN5** = 1 y **IN7** = 0) o (**IN5** = 0 y **IN7** = 1). De manera semejante, **old5** y **old7** tienen que ser diferentes entre sí.

```
counter = 1
old7 = 0
old5 = 1
```

Ahora llegamos a la sección de detección de esquinas alternadas consecutivas. La primera cosa que queremos checar es si uno u otro de los filamentos está presionado. Una forma simple de hacer esto es preguntar "¿es **IN7** diferente a **IN5**?" en PBASIC, podemos usar el operador no-igual <> en una declaración **IF**:

```
IF (IN7 <> IN5) THEN
```

Si un filamento esta presionado, la siguiente cosa a revisar es si se trata o no del patrón exactamente opuesto a la condición previa. En otras palabras, ¿es (**old7 <> IN7**) y es (**old5 <> IN5**)? Si es cierto, entonces es hora de sumar un 1 al contador que rastrea contactos de filamentos alternados. También es momento de recordad el patrón presente de filamentos haciendo **old7** igual al actual **IN7** y **old5** igual al actual **IN5**.

```
IF (old7 <> IN7) y (old5 <> IN5) THEN
  counter = counter + 1
  old7 = IN7
  old5 = IN5
```

Si resulta que esta es el cuarto contacto de filamento consecutivo, es hora de reiniciar **counter** a 1 y ejecutar una vuelta en U.

```
  IF (counter > 4) THEN
    counter = 1
    GOSUB Back_Up
    GOSUB Turn_Left
    GOSUB Turn_Left
```

Este **ENDIF** termina el bloque de código que es ejecutado si **counter** > 4.

```
  ENDIF
```

Esta declaración **ELSE** esta conectada a la declaración **IF (old7 <> IN7) y (old5 <> IN5) THEN**. La declaración **ELSE** cubre lo que pasa si la declaración **IF** no es cierta. En otras palabras, no debe ser un filamento alternado lo que fué presionado, asi es que se reinicia **counter** porque el Boe-Bot no está atorado en una esquina.

```
ELSE
  counter = 1
```

Esta declaración **ENDIF** termina el proceso de toma de decisiones para la declaración **IF (old7 <> IN7) y (old5 <> IN5) THEN**.

```
  ENDIF
ENDIF
```

El resto del programa se mantiene sin cambios respecto al anterior.

Su Turno

Una de las declaraciones **IF...THEN** en EscapingCorners.bs2 ve si **counter** ha llegado a 4.

- ✓ Intente incrementar el valor a 5 y 6 y note el efecto.
- ✓ Intente también reducir el valor y véa si tiene algún efecto en el avance normal.

RESUMEN

En este Capítulo, en vez de navegar a partir de una lista pre-programada, el Boe-Bot fue programado para navegar basado en entradas de sensores. Las entradas de sensores usadas en este Capítulo fueron filamentos, que operaron como interruptores de contactos normalmente abiertos. Al ser adecuádamente alambrados, estos interruptores pueden mostrar un voltaje (5 V) en el punto de contacto del interruptor cuando está abierto y un voltaje diferente (0 V) cuando está cerrado. Los registros de entrada de los pines I/O del BASIC Stamp guardan "1" si detectan Vdd (5 V) y "0" si detectan Vss (0 V).

El BASIC Stamp fue programado para probrar los sensors de filamento y desplegar los resultados de la prueba usando 2 medios distintos, la Terminal de Depuración y los LEDs. Se desarrollaron programas PBASIC para hacer que el BASIC Stamp cheque el estado de los filamentos entre cada pulso de servo. En base al estado de los filamentos, declaraciones **`IF…THEN`** en la sección principal del programa llamaron a subrutinas de navegación similares a las desarrolladas en el Capítulo anterior para guiar al Boe-Bot lejos de los obstáculos. Como un ejemplo simple de inteligencia artificial, se desarrolló una rutina adicional que habilita al Boe-Bot para detectar cuando se ha quedado atorado en una esquina. Esta rutina involucró guardar estados anteriores de los filamentos, compararlos contra el estado actual y contra el número de detecciones alternadas de un objeto.

Este Capítulo introdujo la navegación del Boe-Bot basada en sensores. Los siguientes 3 Capítulos se enfocarán en usar distintos tipos de sensores para dar al Boe-Bot una visión. Ambos, visión y contacto, abren muchas oportunidades para que el Boe-Bot navegue en ambientes cada vez más complejos.

Preguntas

1. ¿qué tipo de conexión eléctrica es un filamento?
2. Cuando se presiona un filamento, ¿qué voltaje ocurre en el pin I/O que lo monitorea? ¿Qué valor binario ocurrirá en el registro de entrada? Si se usa el pin P8 de I/O para monitorear el pin de entrada, ¿qué valor tendrá **`IN8`** cuando se presione un filamento y qué valor tendrá cuando un filamento no sea presionado?
3. Si **`IN7`** = 1, ¿qué significa? ¿Qué significa si **`IN7`** = 0? ¿Y si **`IN5`** = 1 y **`IN5`** = 0?
4. ¿Qué comando se usa para brincar a diferentes subrutinas dependiendo del valor de una variable? ¿Qué comando se usa para decidir a qué subrutina brincará? ¿En qué se basan estas decisiones?
5. ¿Cuál es el propósito de tener declaraciones **`IF…THEN`** anidadas?

Ejercicios

1. Escriba un comando **DEBUG** para TestWhiskers.bs2 que actualize el estado de cada filamento en una nueva línea. Ajuste **PAUSE** para que sea 250 en vez de 50.
2. Usando RoamingWithWhiskers.bs2 como referencia, escriba una subrutina **Turn_Away** que llame a la subrutina **Back_Up** una vez y a la subrutina **Turn_Left** dos veces. Escriba las modificaciones que tendrá que hacer a la sección de la Rutina Principal de RoamingWithWhiskers.bs2

Proyectos

1. Modifique RoamingWithWhiskers.bs2 para que el Boe-Bot haga un sonido de 4 kHz y 100 ms antes de ejecutar la maniobra evasiva. Haga que suene 2 veces si ambos contactos de filamentos son detectados durante el mismo muestreo.
2. Modifique RoamingWithWhiskers.bs2 para que el Boe-Bot avance en un círculo de 1 yarda (o metro) de diámetro. Cuando toque un filamento, causará que viaje en un círculo más cerrado (de menor diámetro). Cuando toque el otro filamento, causara que el Boe-Bot navegue en un círculo de mayor diámetro.

Soluciones

Q1. Un interruptor táctil.

Q2. Cero (0) volts, resultando en un cero binario (0) en el registro de entrada.
IN8 = 0 cuando el filamento está presionado.
IN8 = 1 cuando el filamento no está presionado.

Q3. **IN7** = 1 significa que el filamento derecho no está presionado.
IN7 = 0 significa que el filamento derecho está presionado.
IN5 = 1 significa que el filamento izquierdo no está presionado.
IN5 = 0 significa que el filamento izquierdo está presionado.

Q4. El comando **GOSUB** ejecuta el brinco presente. El comando **IF...THEN** se usa para decidir a qué subrutina brincar. Esa decisión está basada en condiciones, que son declaraciones lógicas que se evalúan como cierto o falso.

Q5. El programa puede checar una condición, y si esa condición es cierta, puede checar otra condición dentro de la primera.

E1. La clave para resolver este problema es usar un segundo comando **CRSRXY** que coloque el estado del filamento derecho en el lugar adecuado en la pantalla. Para alinearse con los encabezados, el texto debe empezar en la columna 9 del renglón 3.

```
' Robotica con el Boe-Bot - TestWhiskers_UpdateEaOnNewLine.bs2
' Actualiza cada estado de filamento en una nueva linea.
' {$STAMP BS2}                              ' Directiva Stamp.
' {$PBASIC 2.5}                             ' Directiva PBASIC.

DEBUG "WHISKER STATES", CR,
      "Left     Right", CR,
      "------   ------"
DO
  DEBUG CRSRXY, 0, 3, "P5 = ", BIN1 IN5  ' Escribe en Columna 0, Reng 3
  DEBUG CRSRXY, 9, 3, "P7 = ", BIN1 IN7  ' Escribe en Columna 9, Reng 3
  PAUSE 250                              ' Cambia de 50 a 250
LOOP
```


E2. Subrutina:

```
Turn_Away:
  GOSUB Back_Up
  GOSUB Turn_Left
  GOSUB Turn_Left
  RETURN
```

Para modificar la Rutina Principal, reemplace los 3 comandos **GOSUB** bajo la primera condition **IF** con esta línea:

```
    GOSUB Turn_Away
```

P1. La clave para resolver este problema es escribir una declaración que haga un sonido con los parámeteros requeridos:

```
  FREQOUT 4, 100, 4000       ' 4kHz beep for 100ms
```

Esta declaración debe ser agregada a la Rutina Principal en puntos adecuados, como se muestra a continuación. El resto del programa permanence igual.

```
' -----[ Rutina Principal]------------------------------------

DO
  IF (IN5  = 0) y (IN7 = 0) THEN  ' Ambos filamentos detectan
    FREQOUT 4, 100, 4000           ' 4 kHz beep por 100 ms
    FREQOUT 4, 100, 4000           '   Repite 2 veces
    GOSUB Back_Up                  ' Retrocede & vuelta en U
GOSUB Turn_Left
    GOSUB Turn_Left
  ELSEIF (IN5  = 0) THEN           ' Filamento izq contacta
    FREQOUT 4, 100, 4000           ' 4 kHz beep por 100 ms
    GOSUB Back_Up                  ' Retrocede y gira der
    GOSUB Turn_Right
```

```
  ELSEIF (IN7  = 0) THEN            ' Filamento der contacta
    FREQOUT 4, 100, 4000            ' 4 kHz beep por 100 ms
    GOSUB Back_Up                   ' Retrocede y gira izq
    GOSUB Turn_Left
  ELSE                              ' Ambos filamentos 1, no
    GOSUB Forward_Pulse             ' contactos
  ENDIF                             ' Aplica 1 pulso al frente
LOOP                                ' y revisa de nuevo
```

P2. Del Capítulo 4 encontramos proyectos que logran círculos de 1 yarda con **`PULSOUT 13, 850`** y **`PULSOUT 12, 716`**. Usando estos valores, el radio puede ser ajustado incrementando o disminuyendo lígeramente el ancho de pulso a partir del valor inicial de 716. Cada vez que un filamento es presionado el programa sumará o restará un poco del ancho de pulso correcto.

```
' Robotica con el Boe-Bot - CirclingWithWhiskerInput.bs2
' Mueve en circulo 1 yarda, incrementa/disminuye radio en respuesta a
' presion de filamentos, un filamento incrementa, el otro disminuye.
' {$STAMP BS2}                       ' Directiva Stamp.
' {$PBASIC 2.5}                      ' Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Variables/Inicializacion ]----------------------------------
pulseWidth     VAR     Word           ' Señal enviada al servo
toneFreq       VAR     Word           ' frecuencia de tono audible
pulseWidth = 716                      ' Del cap 4 para circulo de 1yd
toneFreq = 4000                       ' Tono inicial es 4 kHz

' -----[ Rutina Principal ]------------------------------------------

DO
  PULSOUT 13, 850                     ' Pulso servos en ruta circular
  PULSOUT 12, pulseWidth              ' 12 mas lento que 13 y arquea
  PAUSE 20
  IF (IN5  = 0) THEN                  ' Filamento izq hace circulo
    IF (pulseWidth <= 845) THEN       ' mas pequeño hasta servo max
      pulseWidth = pulseWidth + 5     ' pulseWidth de 850.
      toneFreq = toneFreq + 100
      FREQOUT 4, 100, toneFreq        ' Toca tono como indicador.
    ENDIF
  ELSEIF (IN7  = 0) THEN              ' Filamento der hace circulo
    IF (pulseWidth >= 655) THEN       ' mas grande, hasta servo min
      pulseWidth = pulseWidth - 5     ' pulseWidth de 650.
      toneFreq = toneFreq - 100
      FREQOUT 4, 100, toneFreq        ' Toca tono como indicador.
    ENDIF
  ENDIF
LOOP
```

Capítulo 6: Navegación Fotosensible con Fototransistores

¿Debería guardar este Capítulo para después? Muchas clases brincan a los Capítulos 7 y 8, y luego regresan aquí si el tiempo lo permite. El Capítulo 7 es el major "siguiente paso" después de la navegación con filamentos porque presenta un sensor que el Boe-Bot puede usar para detectar obstáculos sin chocar con ellos. El Capítulo 8 usa ese mismo sensor para detección de distancia y seguir objetos. Eso completará su introducción a la detección de objetos y navegación. Luego, regrese aquí para hacer que su Boe-Bot detecte y responda a algo entéramente diferente y de alguna más retador — luz ambiental.

Baje código de ejemplo selectos: Algunos de los programas ejemplo más largos en este Capítulo están disponibles como descargas en www.parallax.com/go/Boe-Bot. Busque el archivo LightSensorExamples.zip.

La luz tiene muchas aplicaciones en la robótica y el control industrial. Algunos ejemplos incluyen sensar el extreme de un rollo de tela en la industria textile, determiner cuándo activar luces de calle en diferentes temporadas del año, cuándo tomar una fotografía o cuando entregar agua a un grupo de plantas.

Hay muchos sensors de luz diferentes que atienden funciones unícas. Los sensors de luz en su kit Boe-Bot responden a la luz visible junto con un tipo invisible de luz llamado infrarrojo. Estos sensores pueden ser incorporados en algunos circuitos diversos y el BASIC Stamp puede ser programado para interactuar con ellos para detectar variaciones en el nivel de luz. Con esta información, su programa puede ser expandido para hacer que el Boe-Bot reconozca areas con luz o perímetros en penumbra, reportar niveles generales de brillantez y oscuridad y buscar fuentes de luz como el haz de una lámpara y puertas que permitan la entrada de luz en cuartos oscuros.

PRESENTANDO AL FOTOTRANSISTOR

Un transistor es como una válvula que regula la cantidad de corriente eléctrica que pasa a través de 2 de sus terminales. La tercera terminal de un transistor controla cuanta coriente pasa a través de las otras dos. Dependiendo del tipo de transistor, el flujo de corriente puede ser controlado por voltaje, corriente o, en el caso del fototransistor, por luz.

La Figura 6-1 muestra el esquemático y el dibujo de parte del fototransistor en su kit de Robot Boe-Bot. La brilantez de la luz incidiendo en la terminal base del fototransistor (B) determina cuánta corriente dejará pasar entrando por su terminal colector (C), y saliendo por su terminal emisor (E). Luz más brillante resultará en mas corriente; luz menos brillante resultará en menos corriente.

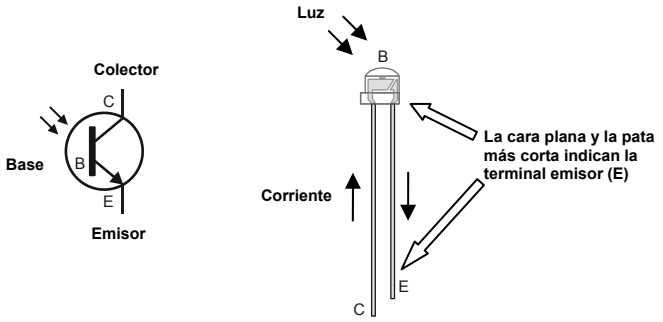


Figura 6-1
Símbolo Esquemático del Fototransistor y Dibujo de Parte

Aunque el fototransistor y el LED son dispositivos diferentes, tienen dos similitudes. Primero, si conecta el fototransitor al revés en el circuito, no trabajará bien. Segundo, el fototransistor tiene dos longitudes de patas diferentes y una cara plana en su encapsulado plástico para identificar sus terminales. La más larga de sus patas indica la terminal collector y la más corta indica el emisor. La Terminal emisor también es la más cercana a una cara plana del encapsulado plástico transparente del fototransistor, que es útil para identificar las terminales si las patas son recortadas.

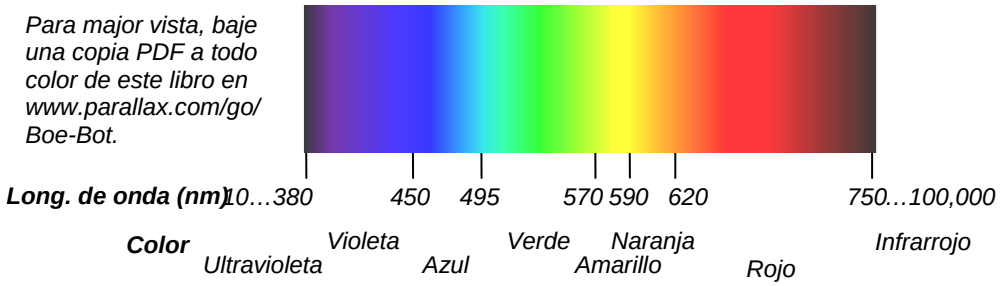
- ✓ Revise la Figura 6-1 y encuentre la cara plana y pata más corta del emitsor.

En el oceano puede medir la distancia entre los picos de dos olas adyacentes en pies o metros. Para la luz, que también viaja en ondas, la distancia entre picos adyacentes es medida en nanómetros (nm) que son billonésimas de metro. La Figura 6-2 muestra la longitud de onda para colores de luz con las que estamos familiarizados junto con algunas que el ojo humano no puede detectar, como el infrarrojo y el ultravioleta.

El fototransistor en el Kit de Partes del Boe-Bot tiene su sensibilidad maxima a 850 nm, que según la Figura 6-2, está en el rango infrarrojo. La luz infrarroja no es visible al ojo humano, pero muchas fuentes de luz diversas emiten cantidades considerables de ella, incluyendo las lámparas halógenas e incandescentes, y especialmente el sol. El fototransistor también responde a la luz visible, aunque es menos sensible a esta, especialmente a longitudes de onda menores a 450 nm, que están a la izquierda del azul en la figura.

Figura 6-2 Longitudes de onda y sus Colores Correspondientes

Para major vista, baje una copia PDF a todo color de este libro en www.parallax.com/go/Boe-Bot.



El diseño de circuitos que usan fototransistores para detección de luz puede ser ajustado para comportarse mejor en ciertos niveles de luz y los circuitos de fototransistor en este capítulo están diseñados para uso en interiores. Luego, si su área para el robot tiene iluminación interior fluorescente, incandescente o halógena indirecta debe trabajar muy bien. Evite los rayos de luz de sol que entre por ventanas cercanas porque sobrecargarán al fototransistor con demasiada luz infrarroja. Si su area esta cercana a ventanas que dejan pasar la esta luz, es buena idea cerrar las persianas antes de empezar. Las lámparas de halógeno apuntando directamente al curso podrían también causar problemas. Deben dar solo luz indirecta, idealmente dirigida hacia arriba para que la luz sea reflejada por el techo. Para mejor resultado, establezca su curso en un área con luz fluorescente brillante.

Iluminancia es el nombre científico para la medición de la luz incidente. Una forma de entender la luz incidente es pensar en la luz de una lampara en una pared. El haz que vea es luz incidente. La unidad de medida de la luminancia es comunmente el "pie-candela" en el sistema Inglés o el "lux" en el sistema métrico. Las mediciones del fototransistor del Boe-Bot no conciernen a niveles de lux, solo si la luz incidente que proviene de cierta dirección es más brillante o más oscura. Luego, el programa del Boe-Bot puede usar las diferencias entre niveles de iluminancia izquierda y derecha para tomar decisiones de navegación.

ACTIVIDAD #1: UN SIMPLE SENSOR DE LUZ BINARIO

Imagine que su Boe-Bot está navegando un curso y que hay una luz brillante al final. Por ejemplo, podría ser una luz brillante indicando un cierto punto. La última tarea en el curso de su Boe-Bot podría ser detenerse debajo de esa luz brillante. Los focos incandescentes en las lámparas de escritorio y de mano son la major fuente de “luz brillante”. Las fuentes compactas fluorescentes y LED no son tan fáciles de reconocer para el circuito en esta Actividad. Si detenerse bajo la luz brillante es la única tarea de

búsqueda de luz de su Boe-Bot, hay un circuito simple que puede usar y que permite al BASIC Stamp detectar luz brillante con un 1 binario o luz ambiental con un 0 binario.

Ambiente. De acuerdo con el diccionario Merriam Webster, la palabra *ambiente* significa existente o presente en todos lados. Para el nivel de luz en un cuarto, piense en luz ambiente como el nivel general de brillantez.

Lista de partes

(1) Fototransistor
(2) Cables de conexión
(1) Resistencia 220Ω (rojo-rojo-cafe)
(1) Resist. 470Ω (amarillo-violeta-cafe)
(1) Resistencia 1kΩ (cafe-negro-rojo)
(1) Resistencia 2kΩ (rojo-negro-rojo)
(1) Resist. 4.7kΩ (amarillo-violeta-rojo)
(1) Resistencia 10 kΩ (cafe-negro-naranja)

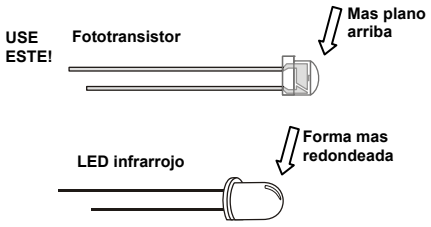


Figura 6-3: Fototransistors vs. LEDs IR

Construyendo el Detector de luz Brillante

La Figura 6-4 muestra el esquemático y el diagrama de conexión de un circuito de voltaje de salida del fototransistor que usará el BASIC Stamp para obtener el valor binario 1 o 0. Después de algunas pruebas y dependiendo de las condiciones de iluminación en su area de trabajo, pudiera tener que cambiar la resistencia de 2 kΩ con otra de la Lista de Partes.

El circuito en la Figura 6-4 es similar a los que encontrará en las luces que se encienden automáticamente en la noche y algunos detectores en transportadeores de banda.

- ✓ Desconecte la energía de su tarjeta y servos.
- ✓ Construya el circuito mostrado en la Figura 6-4.
- ✓ El diagrama de conexiones apunta al pin emisor del fototransistor, que es más corto y más cercano a la cara plana del encapsulado plástico. Vuelva a revisar su cableado usando la Figura como referencia para asegurarse de que el colector y el emisor del fototransistor están correctamente conectados en sus circuitos sensores de luz.

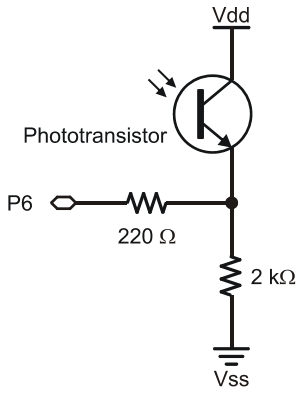
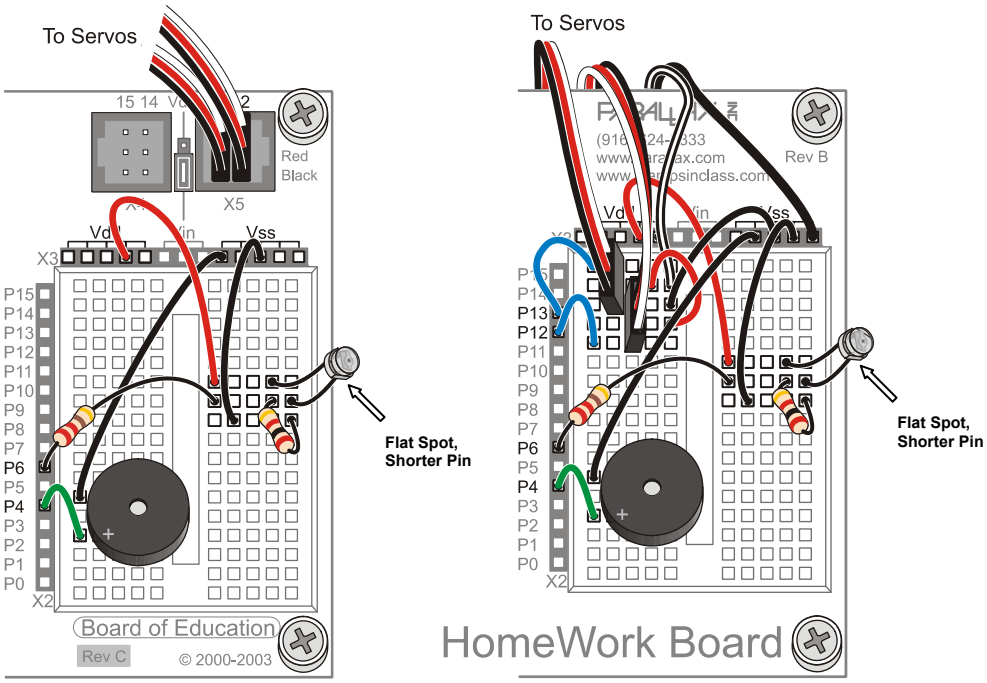


Figura 6-4
Circuito de Voltaje de Salida del Fototransistor

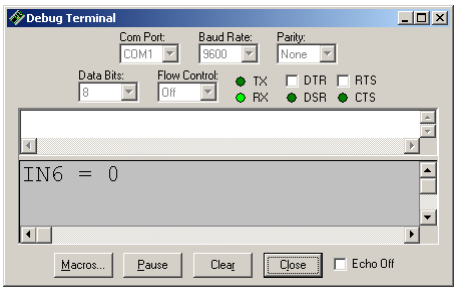
Diagramas de conexión: Board of Education (Izq); HomeWork Board (der)



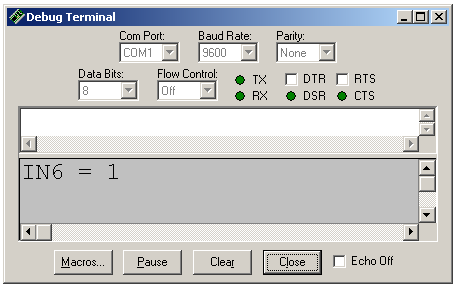
Programa Ejemplo: TestBinaryPhototransistor.bs2

Este programa hace que la Terminal de Depuración despliegue un valor de 0 en un cuarto con luces fluorescentes y sin luz de sol directa. Cuando incide una luz brillante en el fototransistor, el programa desplegará un valor de 1. La Figura 6-5 muestra un ejemplo.

Figura 6-5: La Terminal de Depuración muestra mensajes de TestBinaryPhototransistor.bs2



Luz Fluorescente Ambiental



Luz Brillante

- ✓ Asegúrese de que las terminals del fototransistor no se tocan entre sí. Opcionalmente envuelva las partes expuestas de las mismas con cinta de aislar.
- ✓ Reconecte la energía a su tarjeta.
- ✓ Introduzca, salve y corra TestBinaryPhototransistor.bs2.
- ✓ Observe el valor de **IN6** en la Terminal de Depuración y verifique que guarda un 0 cuando no está bajo luz brillante y un 1 cuando si lo está. Buenas fuentes de luz brillante incluyen lámparas de mano (con focos, no LEDs), de escritorio incandescentes y lámparas halógenas pequeñas.
- ✓ Si la luz ambiental es más brillante que las lámparas fluorescentes y tiene una buena fuente de luz brillante quizá tenga que reemplazar la resistencia de 2 kΩ con una de menor valor. Intente 1 kΩ, 470 Ω, o incluso 220 Ω para fuentes muy brillantes.
- ✓ Si la luz ambiental es baja y está usando una lámpara de escritorio con foco fluorescente o una lámpara de mano con foco LED como fuente de luz brillante quizá tenga que cambiar la resistencia de 2 kΩ por una de 4.7 kΩ o incluso 10 kΩ.

```
' Robotica con el Boe-Bot - TestBinaryPhototransistor.bs2
' Despliega 1 cuando el circuito fototransistor manda mas de 1.4 V a P6
' o 0 cuando manda menos de 1.4 V.

' {$STAMP BS2}
' {$PBASIC 2.5}

PAUSE 1000
DEBUG CLS

DO

  DEBUG HOME, "IN6 = ", BIN IN6
  PAUSE 100

LOOP
```


Su Turno – Haga que el Boe-Bot se detenga bajo la luz brillante

HaltUnderBrightLight.bs2 hará que el Boe-Bot avance hasta que el fototransistor detecte una luz lo suficiente brillante para hacer que **IN6** guarde un 1 binario.

- ✓ Inicie el programa con el Boe-Bot a unos cuantos pies de la luz brillante.
- ✓ Coloque el Boe-Bot para que vaya directo a la luz brillante. ¿Qué tan cerca estuvo el Boe-Bot de detenerse directamente bajo la luz?
- ✓ Intente haciendo ajustes al código y a las resistencias para que el Boe-Bot se detenga justo debajo de la luz brillante.

```
' Robotica con el Boe-Bot - HaltUnderBrightLight.bs2
' Velocidad plena al frente hasta que luz brillante haga que el voltaje de
' salida del circuito del fototransistor exceda 1.4 V, resultando en IN6=1

' {$STAMP BS2}
' {$PBASIC 2.5}

FREQOUT 4, 2000, 3000
DEBUG "Program running... "

DO UNTIL IN6 = 1

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

LOOP
```

Tema Avanzado: Como trabaja el circuito de voltaje de salida del fototransistor

El circuito fototransistor que construyó aplica un voltaje al pin I/O P6. El voltaje marcado V_{P6} en la Figura 6-6 es el voltaje de salida que el circuito aplica al pin I/O. Este voltaje incrementa con más luz y disminuye con menos luz. Puesto que P6 is establecido como entrada, este voltaje causa que **`IN6`** guarde un 1 o un 0 binario. Si el voltaje es mayor que 1.4 V, **`IN6`** guarda un 1 binario; si está por debajo de 1.4 V, **`IN6`** guarda un 0 binario.

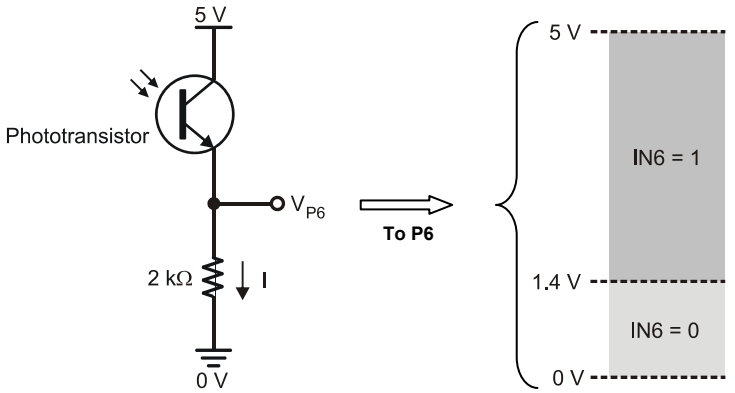


Figura 6-6
Circuito de volltaje de salida del fototransistor y Respuesta de IN6 a V_{P6}

Una resistencia "resiste" al flujo de corriente. El voltaje en un circuito con 1 resistencia puede compararse con la presión de agua. Para una cierta cantidad de corriente eléctrica, se piede más voltaje (presión) a través de una resistencia más grande que a través de una mas pequeña que tiene la misma cantidad de corriente pasando a través de ella. Si ahora varía la corriente y mantiene fija la resistencia, puede medir un voltaje mayor (caída de presión) en la misma resistencia con más corriente o menos voltaje con menos corriente.

Si un pin I/O de BASIC Stamp es una entrada el circuito se comporta como si no existieran ni el pin I/O ni la resistencia de 220Ω. La Figura 6-6 muestra un circuito equivalente al recién construído en la tableta cuando el pin I/O se establece como entrada. Con Vdd (5 V) arriba y tierra (0 V) abajo en el circuito, 5 V de presión eléctrica (voltaje) hacen que los electrones en la batería del Boe-Bot quieran fluir a través de el.

Conectado en Serie. Cuando 2 o mas elementos son conectados punta a punta, están conectados en serie. El fototransistor y la resistencia (Figura 6-6) están en serie.

Un umbral lógico es un voltaje que establece una distinción entre un 1 y un 0 binarios. Para un pin I/O BASIC Stamp establecido como entrada, ese umbra les 1.4 V.

La razón por la que el voltaje en P6 cambia con la luz es porque el fototransistor deja pasar más corriente con más luz o menos corriente con menos luz. Esa corriente I (Figura 6-6), también tiene que pasar a través de la resistencia. Cuando pasa más corriente a través de una resistencia, el voltaje a través de ella será mayor y viceversa.. Puesto que una de las terminals de la resistencia está unida a Vss = 0 V, el voltaje en la Terminal V_{P6} sube con mas corriente y baja con menos corriente.

Si cambia la resistencia de 2 kΩ por una de 1 kΩ, V_{P6} será menor para las mismas corrientes. De hecho, será necesaria el doble de corriente para hacer que V_{P6} cruce el umbral lógico de 1.4 V del pin I/O de BASIC Stamp; es decir, la luz tendrá que ser el doble de brillante para que **`IN6`** guarde un 1. Asi pues, una resistencia menor en serie con el fototransistor hará al circuito menos sensible a la luz. Si ahora reemplaza la resistencia de 2 kΩ con una de 10 kΩ, V_{P6} será 5 veces mayor con la misma corriente y solo tomará 1/5 de luz para generar 1/5 de corriente e incrementar V_{P6} por encima de 1.4 V y hacer que **`IN6`** guarde un 1. Asi, una resistencia mayor hara al circuito más sensible a la luz.

La Ley de Ohm para calcular Voltaje, Corriente y Resistencia

Dos propiedades afectan el voltaje V_{P6}: corriente y resistencia, y la Ley de Ohm explica como. La Ley de Ohm establece que el voltaje (V) a través de una resistencia es igual a la corriente (I) que pasa por ella multiplicado por su resistencia (R). Entonces, si conoce dos de estos valores puede usar la ecuación de la Ley de Ohm para calcular el tercero:

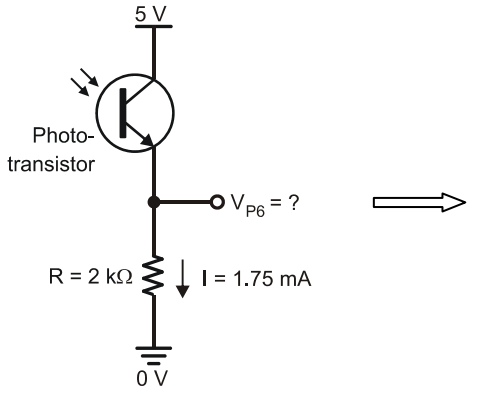
$$\mathbf{V = I \times R}$$

En algunos libros de texto, verá E = I × R. E indica potencial eléctrico.

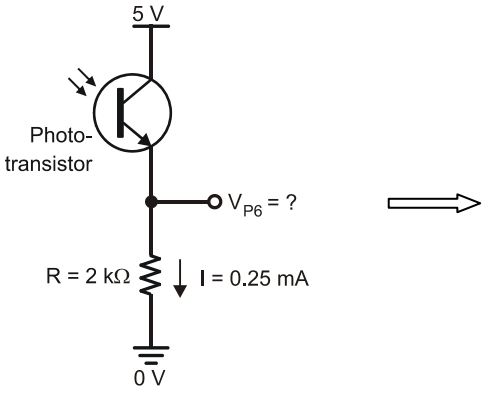
El Voltaje V es medido en unidades llamadas volts, que se abrevian con una V mayúscula. La Corriente I es medida en amperes, que se abrevian A, y la resistencia R es medida en ohms que se abrevian con la letra griega omega (Ω). Los niveles de corriente que verá en este circuito estan en el orden de milliamps (mA). La m minúscula indica que es una medición de milésimas de amps. De manera semejante, la k minúscula en kΩ indica que la medición es el miles de ohms.

Usemos la ley de Ohm para calcular V_{P6} permitiendo con the fototransistor fluir dos cantidades de corriente diferentes a través del circuito: 1.75 mA, que podría pasar como resultado de una luz bastante brillante, y 0.25 mA, que podría ocurrir con luz menos brillante. La Figura 6-8 muestra las condiciones y sus soluciones. Cuando intente estos calculus, recuerde que milli (m) es milésimas y kilo (k) es miles cuando sustituya los números en la Ley de Ohm.

Figura 6-7: V_{P6} Cálculos para 2 corrientes Diferentes del arreglo Fototransistor-Resistencia



$$
\begin{aligned}
V_{P6} &= I \times R \\
&= 1.75\,mA \times 2\,k\Omega \\
&= \frac{1.75}{1000}\ A \times 2000\ \Omega \\
&= 1.75\,A \times 2\,\Omega \\
&= 3.5\,A\Omega \\
&= 3.5V
\end{aligned}
$$



$$
\begin{aligned}
V_{P6} &= I \times R \\
&= 0.25\,mA \times 2\,k\Omega \\
&= \frac{0.25}{1000}\ A \times 2000\ \Omega \\
&= 0.25\,A \times 2\,\Omega \\
&= 0.5\,A\Omega \\
&= 0.5V
\end{aligned}
$$

Su Turno – Ley de Ohm y Ajustes en la Resistencia

Digamos que ahora la luz en su cuarto es el doble de brillante que la que en el cuarto que resultó en Vo = 3.5 V para luz brillante y 0.5 V para oscuro. Otra situación que podría causar una mayor corriente es si la luz ahora es una fuente intensa de infrarrojo. En cualquier caso, el fototransistor puede permitir el doble de corriente fluir por el circuito, lo que podría llevar a dificultades de medición.

> **Pregunta:** ¿Qué podría hacer para regresar la respuesta del circuito a 3.5 V para luz brillante y 0.5 V para atenuada?
>
> **Respuesta:** Reducir a la mitad la resistencia; hacerla de 1 kΩ en vez de 2 kΩ.

Vuelva a repetir los cálculos con la Ley de Ohm para R = 1 kΩ, y corriente brillante I = 3.5 mA y corriente atenuada I = 0.5 mA. ¿Regresa Vo a 3.5 V para luz brillante y 0.5 V para luz tenue con el doble de corriente? (debiera, si no es así, revise sus cálculos.)

ACTIVIDAD #2: MIDIENDO NIVELES DE LUZ CON FOTOTRANSISTORES

Esta actividad introduce un circuito que el BASIC Stamp puede usar para medir la brillantez de la luz incidiento en la base del fototransistor. Los valores de las mediciones pueden variar de números pequeños, indicando luz brillante, a números grandes, indicando luz baja.

Binario vs. Analógico y Digital

A sensor binary puede transmitir 2 estados diferentes, típicamente para indicar la presencia o ausencia de algo. Por ejemplo, un filamento envía una señal alta si no está presionado o una señal baja si esta presionado.

Un sensor analógico envía un rango continuo de valores que corresponden a un rango continuo de mediciones. Los circuitos de fototransistor en esta Actividad son ejemplos de sensores analógicos que proveen rangos continuos de valores que corresponden con rangos continuos de niveles de luz.

Un valor digital es un número expresado por dígitos. Las computadoras y los microcontroladores gruardan las mediciones analógicas como valores digitales. El proceso de medir un sensor analógico y guardar esa medición como un valor digital es llamado conversión analógica a digital. La medición es llamada medición digitalizada. Los documentos resultantes de la conversión analógica a digital también son llamados mediciones cuantizadas.

Lista de partes

En esta actividad, necesitará 2 fototransistores y 2 capacitores de 0.1 µF. La Figura 6-8 muestra dibujos de ambos.

- ✓ Véa cuidadosamente la Figura 6-8 y note la diferencia entre un fototransistor y un LED infrarrojo.

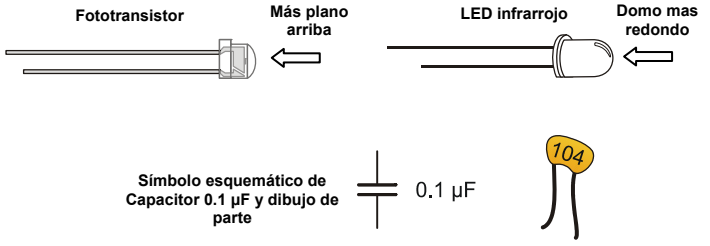


Figura 6-8
Diferencias entre Fototransistores y LEDs infrarrojos; Identificando el Capacitor de 0.1 µF

- ✓ Reúna las partes listadas a continuación usando la Figura 6-8 como guía para encontrar los fototransistores y capacitores de 0.1 µF en su kit de partes.

(2) Fototransistores
(2) Capacitores, 0.1 µF (104)
(2) Resistencias, 1 kΩ (cafe-negro-rojo)
(2) Cables de conexión

Presentando al Capacitor

Un capacitor es un dispositivo que guarda carga y es un bloque de construcción fundamental para muchos circuitos. Las baterías también son dispositivos que guardan carga y, para los fines de estas actividades, será conveniente pensar en capacitares como baterías muy pequeñas que pueden ser cargados, descargados y recargados.

La cantidad de carga que un capacitor tiende a guardar se mide en farads (F). Un farad es un valor muy grande e impráctico con estos circuitos del Boe-Bot. Los capacitores que usará en esta Actividad guardan fracciones de millonésimas de farads. Una millonésima de un farad es llamada microfarad, y se abrevia µF. El capacitor que usará en este ejercicio guarda la décima de una millonésima de un farad. Esto es, 0.1 µF.

Algunas mediciones communes de capacitancia son:

Microfarads:	(millonésima de un farad), abreviado µF	1 µF = 1×10^{-6} F
Nanofarads:	(billonésima* de un farad), abreviado nF	1 nF = 1×10^{-9} F
Picofarads:	(trillonésima* de un farad), abreviado pF	1 pF = 1×10^{-12} F

El 104 en el capacitor de 0.1 µF es una medición de picofarads o (pF). En este sistema de etiquetado, 104 representa un 10 con cuatro ceros, entonces el capacitor es de 100,000 pF, o sea 0.1 µF.

$(100{,}000) \times (1 \times 10^{-12})$ F = $(100 \times 10^{3}) \times (1 \times 10^{-12})$ F

$= 100 \times 10^{-9}$ F = 0.1×10^{-6} F

= 0.1 µF.

*para el sistema inglés (N. del T.)

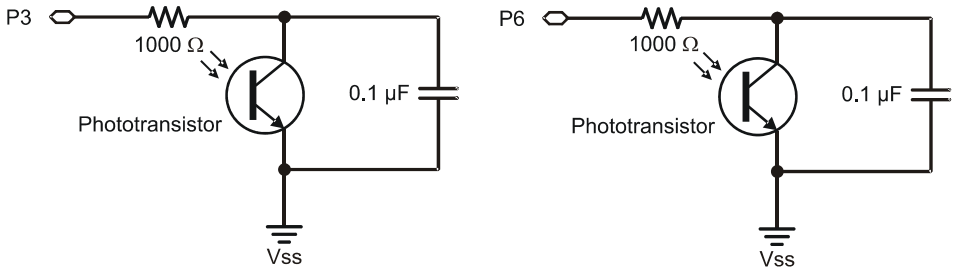
6

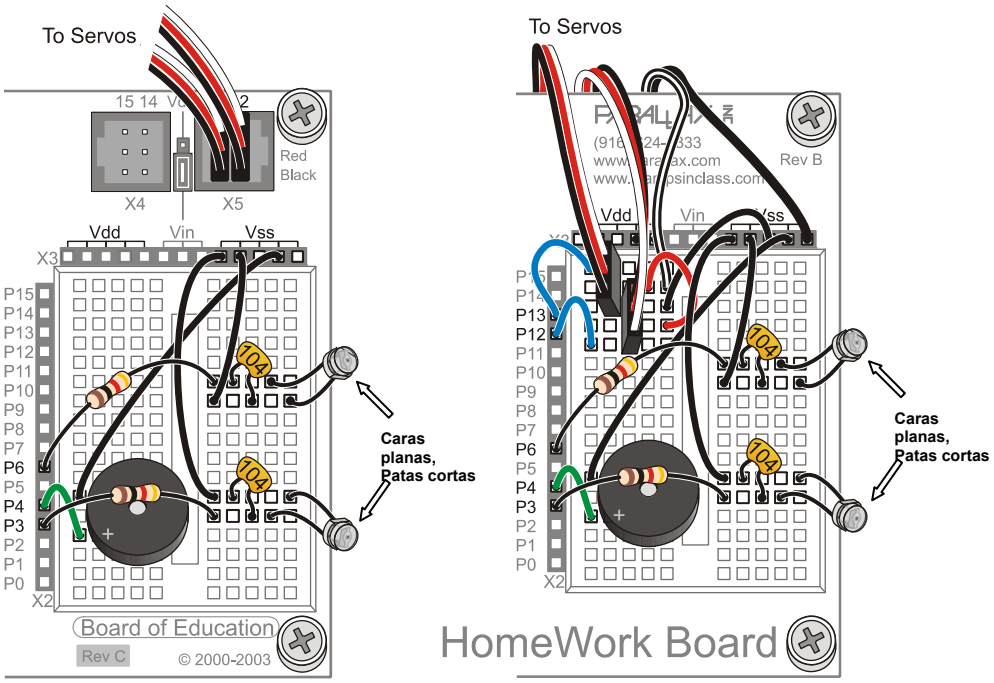
Construyendo los ojos fotosensibles

El BASIC Stamp puede usar los circuitos en la Figura 6-9 para medir la cantidad de luz incidiendo en cada base del fototransistor. Un fototransistor apuntará al frente y a la izquierda y el otro al frente y a la derecha. Ambos apuntarán hacia arriba unos 45°. Puesto que están apuntando en diferentes direcciones, el BASIC Stamp sera capaz de usarlos para determinar si una luz es más brillante a la derecha o izquierda del Boe-Bot.

- ✓ Desconecte la energía de su tarjeta y servos.
- ✓ Remueva todas las partes de su circuito de voltaje de salida de fototransistor de la Figura 6-4, incluyendo el cable que conecta la Terminal colector del fototransistor a Vdd.
- ✓ Construya los circuitos mostrados en la Figura 6-9.
- ✓ Vuelva a revisar sus circuitos contra el diagrama de alambrado para asegurarse que sus fototransistores no están conectados al revés. Use los indicadores “pata mas corta” y “cara plana” como guía.

Figura 6-9: Esquemáticos y Diagrama de alambrado del circuito Analógico fototransistor

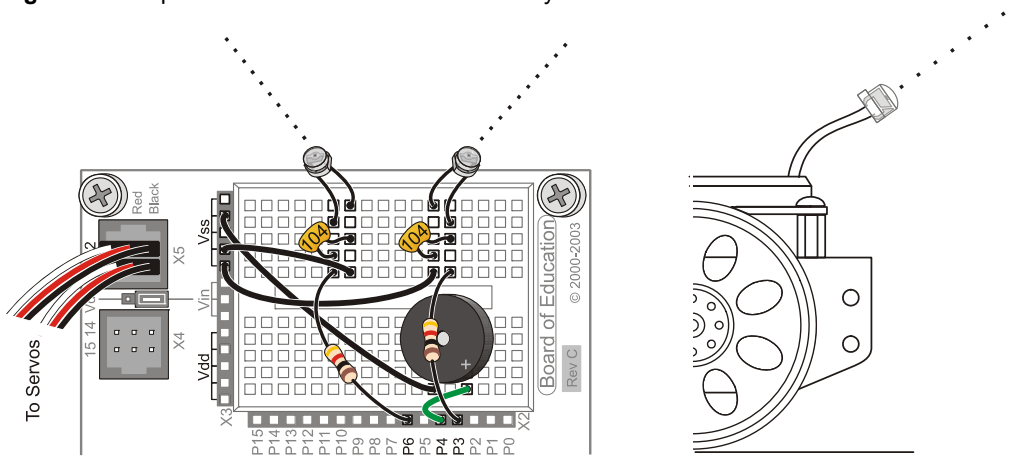




Los ejemplos de ubicación en este Capítulo dependerán de que los fototransistores sean apuntados hacia arriba y afuera para detectar las diferencias en los niveles de luz incidentes en diferentes direcciones.

- ✓ Asegúrese de que sus fototransistores están apuntando arriba y afuera como se muestra en la Figura 6-10.

Figura 6-10: Apuntando los fototransistores arriba y afuera



Acerca la transferencia de carga y el circuito fototransistor

Cada circuito fototransistor/capacitor es llamado circuito de transferencia de carga. El BASIC Stamp medirá la rapidéz con la que cada capacitor pierde su carga a través de su fototransistor midiendo cuánto tiempo le toma abatirse el voltaje del capacitor. El tiempo de abatimiento corresponde a la brillantez de la luz incidiendo en la base de cada fototransistor. Un abatimiento rápido significa más luz, un abatimiento lento significa menos luz.

Circuito QT: Una abreviación común para transferencia de carga es QT. La letra Q se refiere a la carga eléctrica (una acumulación de electrones), y T abrevia transferencia.

Visualice los capacitores en el circuito de la Figura 6-11 como pequeñas baterías recargables y los fototransistores como válvulas de corriente controladas por luz. Cada capacitor puede ser cargado a 5 V y luego permiten perder su carga a través de su fototransistor. La rapidez con la que el capacitor pierde su carga depende de cuanta corriente permite pasar el fototransistor (válvula de corriente), que a su vez depende de la brillantez de la luz sobre la base del fototransistor. Nuevamente, la luz más brillante resulta en más corriente, las sombras resultan en menos corriente.

Conectado en Paralelo

El fototransistor y el capacitor mostrados en la Figura 6-11 están conectados en paralelo. Para que 2 componentes estén conectados en paralelo, cada una de sus terminales debe estar conectada a terminales comunes (también llamados nodos). El fototransistor y el capacitor tienen cada uno una pata conectada a Vss. También tiene cada uno conectada una pata a la misma pata de la resistencia. Entonces, están conectados en paralelo.

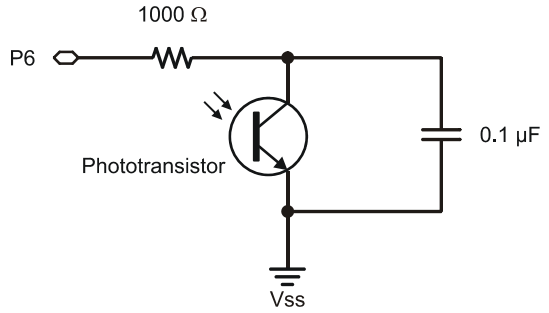


Figura 6-11
Circuito QT Conectado al Pin I/O P6

El BASIC Stamp ejecuta los siguientes pasos para medir un nivel de luz con el circuito fototransistor de transferencia de carga de la Figura 6-11:

1. Usa el comando **`HIGH`** para aplicar 5 V al circuito y carga el capacitor (batería pequeña).
2. Usa el comando **`PAUSE`** para esperar que el capacitor cargue.
3. Usa el comando **`RCTIME`** para establecer el pin I/O como entrada y medir el tiempo que toma al capacitor abatir su voltaje hasta 1.4 V al ir perdiendo su carga a través del fototransitor.

Una medición mayor de tiempo de abatimiento en el paso 3 significa menos luz; un tiempo menor significa más luz.

El comando **`RCTIME`** cambia la dirección ***Pin*** de salida a entrada y luego espera que estado del pin I/O cambie, lo que ocurre cuando el voltaje que el circuito aplica al pin alcanza el umbral lógico de 1.4 V. El comando **`RCTIME`** guarda en ***Variable*** la medición de tiempo resultante. Con el BASIC Stamp 2, este resultado es un número múltiplo de 2 µs.

RCTIME *Pin*, *State*, *Variable*

Si el argumento ***State*** es establecido en 1, **`RCTIME`** esperará a que cambie a 0 indicando que el voltaje se abatió hasta 1.4 V. Si **`State`** es establecido en 0, **`RCTIME`** esperará a que el voltaje suba a 1.4 V. En cada caso, el comando guarda en el argumento ***Variable*** la medición de tiempo resultante, que es típicamente una variable de tamaño word.

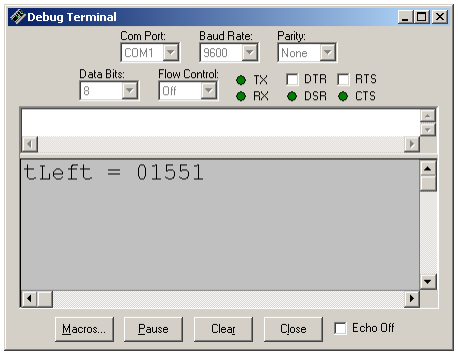
Cuando el comando `RCTIME` cambia la dirección pin de salida a entrada, deja de cargar el capacitor y se vuelve invisible al circuito. Tan pronto como esto ocurre, la carga del capacitor empieza a drenarse a través del fototransistor. Como una entrada, el pin I/O puede sensar si el voltaje del circuito está arriba o abajo de 1.4 V.

RC en `RCTIME` significa resistencia-capacitor y el uso más común del comando **`RCTIME`** es con sensores que varían ya sea con resistencia o capacitancia. Para un ejemplo que usa este comando para medir la posición de un indicador que controla resistencia, Véa *¿Qué es un Microcontrolador*?, Capítulo 5. Es una descarga gratuita en www.parallax.com/go/WAM.

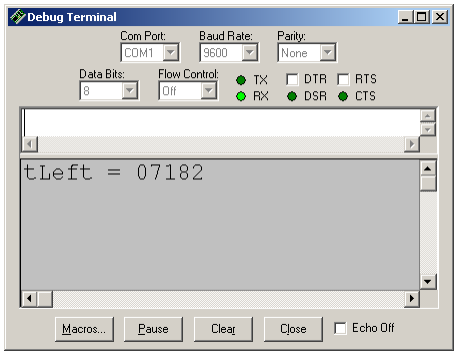
Pruebe el Circuito Fototransistor

El programa ejemplo TestP6LightSense.bs2 ejecuta los 3 pasos en el circuito QT conectado a P6 en la Figura 6-11 y despliegua mediciones de tiempo que representan el nivel de luz incidente en la terminal base del fototransistor. El circuito QT conectado a P6 es el sensor de luz izquierdo del Boe-Bot, y la Terminal de Depuración desplegará el tiempo de abatimiento del voltaje como **`tLeft`**, que es el nombre de la variable que guardará el resultado pero también una abreviación de tiempo-izquierda. El valor que despliega es el tiempo de abatimiento medido en incrementos de 2 µs. Este valor disminuirá con luz más brillante e incrementará con luz menos brillante, como indica la Figura 6-12.

Figura 6-12: Dos niveles de luz diferentes medidos por el sensor de luz izquierdo del Boe-Bot



Luz de interior Normal



Sombra sobre el Sensor

¿Está `tLeft` atorado en 0 o 1? Un 0 pudiera significar que está demasiado oscuro y un 1 pudiera significar que está demasiado brillante. Cualquiera de ellos también pudiera indicar un error en el cableado, verifique sus circuitos.

- ✓ Estos circuitos sensores están diseñados para iluminación en interiores. Asegúrese de que no hay luz de día por las ventanas. Si es así, cierre sus persianas.
- ✓ Introduzca y corra TestP6LightSense.bs2.
- ✓ Anote of el valor desplegado en la Terminal de Depuración.
- ✓ Use su mano o un libro para provocar una sombra sobre el fototransistor en el circuito conectado a P6.
- ✓ Revise nuevamente la medición en la Terminal de Depuración. El valor debe ser mayor que el primero. Anótelo también.
- ✓ Mueva el objeto que provoca la sombra más cerca a la parte superior del fototransistor. Intente hacer la sombra el doble de oscura y anote la medición.
- ✓ Experimente con sombras progresívamente más oscuras, incluso poniendo su mano sobre el fototransistor. (Si el nivel de luz se hace suficientemente bajo, el comando **`RCTIME`** puede exceder su máximo de 65535, en cuyo caso el comando guardará un 0 en la variable **`tLeft`**, y la Terminal de Depuración desplegará "tLeft = 00000.")

```
' Robotica con el Boe-Bot - TestP6LightSense.bs2
' Prueba el circuito fotoresistor izquierdo del Boe-Bot.

' {$STAMP BS2}                     ' Modulo = BASIC Stamp 2
' {$PBASIC 2.5}                    ' Lenguaje = PBASIC 2.5

tLeft          VAR     Word        ' Guarda tiempo de disminuc del sensor izq

PAUSE 1000                         ' Espera 1 s antes de cualquier DEBUG

DO                                 ' Ciclo Principal

  HIGH 6                           ' 1 Establece P6 alto para empezar carga
  PAUSE 1                          ' 2 Espera a que cargue el capacitor
  RCTIME 6, 1, tLeft               ' 3 P6->entrada, mide tiempo de abatimiento

  DEBUG HOME, "tLeft = ", DEC5 tLeft  ' Despliega resultado
  PAUSE 100                           ' Espera 0.1 segundos

LOOP                                  ' Repite Ciclo principal
```

Su Turno – Prueba el otro Circuito Fototransistor

El otro circuito fototransistor del Boe-Bot esta conectado a P3. Antes de modificar su programa para probar el otro circuito, siempre es mejor primero salvar el programa que trabaja tal y como esta.

- ✓ Salve TestP6LightSense.bs2, luego salve una copia como TestP3LightSense.bs2.
- ✓ Cambie el argumento ***Pin*** de 6 a 3 en los comandos **HIGH** y **RCTIME**.
- ✓ Cambie el nombre de la variable de **tLeft** a **tRight** en la declaración **VAR** y en los comandos **RCTIME** y **DEBUG**.
- ✓ Pruebe y corrija errores tipográficos o de otro tipo.
- ✓ Actualice los comentarios al principio del programa.
- ✓ Salve su programa modificado y córralo.

También sería bueno tener un tercer programa que pruebe ambos circuitos fototransistores. Como antes, salve uno de los programas que ya funciona y luego salve una copia bajo un nuevo nombre, por ejemplo TestP6P3LightSense.bs2. Este programa necesitará 2 declaraciones de variables, y 2 juegos de comandos **HIGH-PAUSE-RCTIME** en su ciclo principal. Los comandos **DEBUG** pueden ser condensados en uno. Podría verse así:

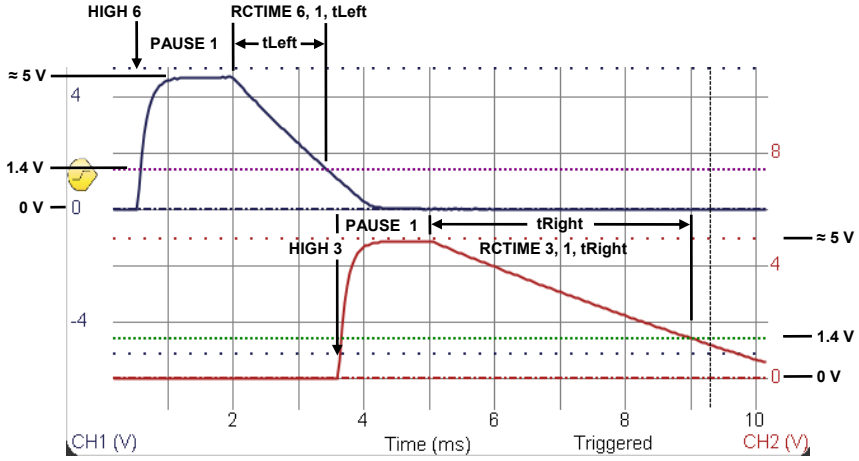
```
DEBUG HOME, "tLeft = ", DEC5 tLeft, "  ", "tRight = ", DEC5 tRight
```

- ✓ Pruebe TestP6P3LightSense.bs2. Esta en LightSensorExamples.zip, que es una descarga gratuita en www.parallax.com/go/Boe-Bot.
- ✓ Véa si puede rotar el Boe-Bot y detectar cuál está apuntando hacia la fuente más brillante en el cuarto (valor más bajo) y cual está apuntando lejos (valor más alto).

Tema Optional Avanzado: Gráficas de Reducción de Voltaje

La Figura 6-13 muestra las respuestas del circuito QT izquierdo y derecho del robot Boe-Bot al estarse ejecutando TestP6P3LightSense.bs2. El dispositivo que mide y grafica estas respuestas de voltaje en el tiempo es llamado osciloscopio. Las 2 líneas que grafican las 2 señales de voltaje se llaman trazas. La escala de voltaje para la traza superior está a la izquierda y la escala de voltaje para la traza inferior está a la derecha. La escala de tiempo para ambas trazas está abajo. Las etiquetas muestran cuando cada comando en TestP6P3LightSense.bs2 es ejecutado para que pueda ver como responden las señales de voltaje.

Figura 6-13: Vista del Osciloscopio de los tiempos de Abatimiento



La traza superior en la Figura 6-13 grafica el voltaje del capacitor en el circuito QT conectado a P6; es el circuito sensor de luz izquierdo del Boe-Bot. En respuesta a **`HIGH 6`**, el voltaje sube de 0 V a casi 5 V entre alrededor 0.5 ms y 1 ms. La señal se mantiene a aproximádamente 5 V por la duración de **`PAUSE 1`**. Luego, **`RCTIME`** causa que inicie el abatimiento que dura aproximádamente 2 ms en la gráfica. El comando **`RCTIME`** mide el tiempo que le toma al voltaje abatirse hasta 1.4 V y lo guarda en la variable **`tLeft`**. En la gráfica, el abatimiento hasta 1.4 V toma aproximádamente 1.5 ms, por lo que la variable **`tLeft`** debe guardar algo alrededor de 750 puesto que 750 × 2 µs = 1.5 ms.

La traza inferior en la Figura 6-13 grafica el otro voltaje del capacitor del circuito QT—the sensor P3 en el lado derecho del Boe-Bot. Esta medición empieza después de la medición de P6 en el lado izquierdo ha sido completada. El voltaje varía de modo similar a la traza superior, excepto que el tiempo de abatimiento es bastante más largo, aproximadamente 5 ms, y esperaríamos ver que **`tRight`** guarde un valor en las cercanías de 2500. Este valor más grande corresponde a un abatimiento más lento, lo que a su vez corresponde a un nivel de luz más bajo.

Tome sus propias mediciones de osciloscopio. Puede medir y aprender más acerca de las señales en este Capítulo con el kit y libro *Entendiendo Señales con el PropScope*. Para averiguar más, vaya a www.parallax.com/go/PropScope.

ACTIVIDAD #3: AJUSTE DE SENSIBILIDAD A LA LUZ

Si estas mediciones de luz con **RCTIME** van a ser usadas mientras que el Boe-Bot esta avanzando, tendrán que compartir el tiempo de procesamiento del BASIC Stamp con los comandos **PULSOUT** para el control de los servos. Hay una ventana de tiempo de 20 ms entre cada par de comandos **PULSOUT** para los comandos **RCTIME**. Aún cuando retrasos de 25 o 30 ms entre pulsos de servos pudieran no causar ninguna diferencia detectable, retrasos de más de 50 ms causarán problemas bien detectables y retrasos más largos causarán que los servos simplemente se convulsionen periódicamente en vez de rotar.

Un par de mediciones de fototransistor en un area realmente obscura podrían medir 50,000 cada una. Para ambas mediciones, esto sería 100,000 × 2 µs = 400 ms. Todo lo que los servos harían con este retraso entre pulsos de control sería convulsionarse cada 0.4 segundos.

En esta Actividad, intentará una técnica que puede ser usada para reducir las mediciones del tiempo de abatimiento en cuartos más oscuros. También probará los efectos de la luz reducida en el desempeño de los servos usando ambas técnicas de medición.

Corrija el problema cargando el Capacitor a un voltaje más bajo con PWM

¿Cómo puede un programa hacer que las mediciones tomen menos tiempo? Cargando los capacitores con voltajes más bajos antes de iniciar las mediciones de abatimiento, el programa puede reducer el tiempo que le toma al abatimiento llegar a 1.4 V. El lenguaje PBASIC tiene un comando llamado **PWM** que puede usar para que el BASIC Stamp establezca el voltaje inicial en el capacitor a un valor más bajo. Este comando le da 256 niveles de voltaje a escoger en el rango de 0 a 4.98 V. La sintaxis del comando **PWM** es:

PWM ***Pin***, ***Duty***, ***Duration***

El comando **PWM** aplica una secuencia rápida de señales alto/bajo al ***Pin*** I/O por una cierta duración ***Duration*** medida en ms. El argumento ***Duty*** es el número de 256avos de tiempo que la señal está en nivel alto y determina el número de 256avos de 5 V al que el capacitor queda cargado. Por ejemplo, el comando **PWM 6, 128, 1** manda una secuencia rápida de señales alto/bajo por 1 ms. Están en nivel alto por 128/256avos del tiempo—esto es, la mitad del tiempo. Entonces, carga al capacitor a 128/256avos de 5 V. Esto es la mitad de 5 V, que es 2.5 V.

PWM significa Modulación de Ancho de Pulso (Pulse Width Modulation). En el Capítulo 2, estudió la modulación de ancho de pulso para control de servo usando **PULSOUT**. El comando **PWM** hace al BASIC Stamp crear otra forma de modulación de ancho de pulso. Esta señal es una secuencia más rápida de pulsos que es especialmente últil para establecer un voltaje en un capacitor por medio de una resistencia. La relación de tiempo en nivel alto respecto al tiempo de ciclo (tiempo alto + tiempo bajo) es lo que controla el voltaje del capacitor y es llamado ciclo de trabajo. El argumento ***Duty*** del comando **PWM** controla el ciclo de trabajo de la señal PWM.

Dado un comando **PWM**, puede calcular el voltaje que establece en el capacitor multiplicando 5 V por el argumento ***Duty*** del comando y luego dividiendo entre 256:

Vcap = 5 V × Duty ÷ 256

He aquí dos ejemplos:

```
PWM 6, 128, 1                   '  5 V × 128 ÷ 256 = 2.5 V.
PWM 6, 96, 1                    '  5 V × 96 ÷ 256 = 1.875 V.
```

Digamos que quiere saber qué valor ***Duty*** usar para un voltaje en particular. Solo divida ambos lados de la ecuación entre 5 V y multiplique ambos lados por 256. El resultado es:

Duty = Vcap × 256 ÷ 5 V

Un valor útil de Duty sería el necesario para que Vcap = 1.4 V. Valores menores que ese no serían últiles para los fines de medición de tiempo de abatimiento de voltaje.

Duty = 1.4 V × 256 ÷ 5 V = 71.68

Luego, un valor de 72 sería el menor argumento ***Duty*** útil en **PWM 6, 72, 1**.

¿Por qué el argumento *Duration* del comando PWM siempre es 1 en estos ejemplos?

Porque 1 ms es suficiente para cargar un capacitor de 0.1 µF a través de una resistencia de 1 kΩ. La regla general es que necesita al menos 5×R×C segundos para cargar un capacitor. Con una resistencia de 1 kΩ y un capacitor de 0.1 µF, el mínimo sería 5 × 1000 × 0.0000001 = 0.0005 s = 0.5 ms. Entonces, un tiempo de carga de 1 ms es más que suficiente.

Si la resistencia o el capacitor fueran más grandes, el argumento ***Duration*** del comando **PWM** pudiera haber sido mayor. Por ejemplo, si se usara un capacitor de 1 µF, el argumento ***Duration*** tendría que haber sido al menos 5 para un tiempo de carga de 5 ms porque 5×R×C = 5 × 1000 × 0.000001 = 0.005 s = 5 ms. R×C es llamada la constante de tiempo RC y frecuentemente se abrevia con la letra Griega tau τ.

Empecemos por reducir los tiempos de abatimiento a la mitad de los valores que midió en la actividad previa. En la práctica, su programa necesitará reducirlos más para navegar en niveles de luz más bajos, pero aquí se muestra el primer paso. Para reducir el tiempo de abatimiento a la mitad tendrá que usar el comando **PWM** para cargar los capacitores al valor intermedio entre 1.4 V y 5 V. Esto corresponde a un valor ***Duty*** de **PWM** que esté a la mitad entre 72 y 256, que es (72 + 256) ÷ 2 = 184. Luego, puede reemplazar **HIGH 6** y **PAUSE 1** con **PWM 6, 184, 1** para reducir los tiempos de abatimiento a la mitad de los valores. Otra forma de visualizar esto es que está usando el comando **PWM** para hacer a los sensores la mitad de sensibles a la luz porque las mediciones de tiempo de abatimiento tomarán la mitad de tiempo para la mitad de los valores medidos.

- ✓ Introduzca, salve y corra HalfLightSensitivity.bs2.
- ✓ Intente que la luz ambiente sea igual al nivel que tenía en la actividad previa.
- ✓ Verifique que los niveles de luz son cerca a la mitad de lo que eran con TestP6LightSense.bs2 de la actividad previa. La precisión no es importante aquí. No se preocupe sis us mediciones no son exactamente la mitad de lo que eran, basta con que cumplan en estar en el orden de la mitad.
- ✓ Intente cambiar el the argumento ***Duty*** del comando **PWM** a 128 y verifique que las mediciones ahora están en las cercanías de un cuarto de los valores de TestP6LightSense.bs2. Nuevamente, no se preocupe por la precisión.

```
' Robotica con el Boe-Bot – HalfLightSensitivity.bs2
' Prueba circuitos fotoresistivos del Boe-Bot con el comando PWM cortando
' la sensibilidad a la luz de los fototransistores a la mitad.
' {$STAMP BS2}                              ' Modulo = BASIC Stamp 2
' {$PBASIC 2.5}                             ' Lenguaje = PBASIC 2.5

tLeft       VAR     Word         ' Guarda tiempo de abatimiento del sensor izq
tRight      VAR     Word         ' Guarda tiempo de abatimiento del sensor der

PAUSE 1000                       ' Espera 1 s antes de cualquier DEBUG

DO                               ' Ciclo Principal

  PWM 6, 184, 1                  ' Carga capacitor a 3.59 V
  RCTIME 6, 1,tLeft              ' P6->entrada, mide tiempo de abatimiento

  PWM 3, 184, 1                  ' Carga capacitor a 3.59 V
  RCTIME 3, 1,tRight             ' P6->entrada, mide tiempo de abatimiento

  DEBUG HOME, "tLeft = ", DEC5 tLeft, CR,    ' Despliega resultados
              "tRight = ", DEC5 tRight
  PAUSE 100                                  ' Espera 0.1 segundos

LOOP                                         ' Repite Ciclo Principal
```

Un Sensor de Luz, Dos Mediciones Diferentes

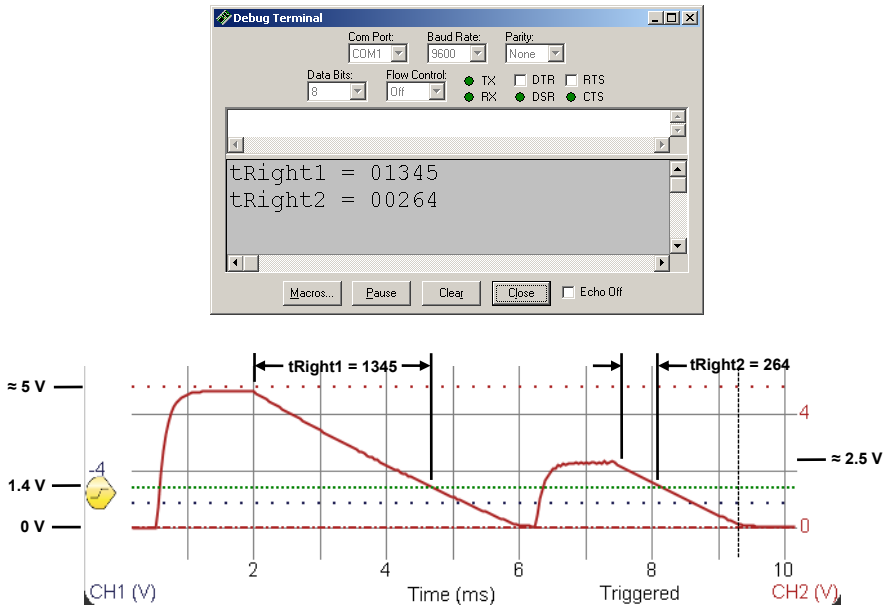
HighVsPwmInRctime.bs2 demuestra como las mediciones de abatimiento del sensor toman menos tiempo cuando **PWM** lo carga a un valor más bajo.

- ✓ Introduzca y corra HighVsPwmInRctime.bs2, y observe las 2 mediciones del mismo nivel de luz en la Terminal de Depuración.
- ✓ Intente variar el nivel de luz, la medición **tRight2** debe ser siempre significatívamente menor que **tRight1**.

```
' Robotica con el Boe-Bot - HighVsPwmInRctime.bs2
' Dos mediciones de abatimiento en fila.  La 1a usa la tecnica HIGH, PAUSE,
' RCTIME, y la 2a carga el capacitor a 2.5 V con PWM antes
' RCTIME.

' {$STAMP BS2}                     ' Modulo = BASIC Stamp 2
' {$PBASIC 2.5}                    ' Lenguaje = PBASIC 2.5

tRight1      VAR     Word          ' Tiempo de abatimiento del 1er sensor der
tRight2      VAR     Word          ' Tiempo de abatimiento del 2do sensor der

PAUSE 1000                         ' Espera 1 s antes de cualquier DEBUG

DO                                 ' Ciclo Principal

  HIGH 3                           ' 1 Establece P3 alto para empezar carga
  PAUSE 1                          ' 2 Espera a que cargue el capacitor
  RCTIME 3, 1, tRight1             ' 3 P3->entrada, mide tiempo de abatimiento

  PAUSE 1                          ' Separa mediciones por 1 ms

  PWM 3, 128, 1                    ' Carga capacitor P3 a 2.5 V
  RCTIME 3, 1,tRight2              ' P3->entrada, mide tiempo de abatimiento

  DEBUG HOME, "tRight1 = ", DEC5 tright1,    ' Despliega resultados
          CR, "tRight2 = ", DEC5 tRight2

  PAUSE 100                                  ' Espera 0.1 segundos

LOOP                                         ' Repite el ciclo principal
```

La Figura 6-14 muestra la Terminal de Depuración y una medición del osciloscopio de los dos abatimientos de HighVsPwmInRctime.bs2. En la traza del osciloscopio, el primer abatimiento inicia en 5 V debido a **HIGH 3** y **PAUSE 1** antes de **RCTIME 3, 1, tRight1**. El segundo abatimiento inicia en 2.5 V debido a **PWM 3, 128, 1** antes de **RCTIME 3, 1, tRight2**.

Figura 6-14: Terminal de Depuración y osciloscopio para HighVsPwmInRctime.bs2



¿Porqué `tRight2` es mas como 1/5 de `tRight1`? ¿No se supone sería 1/4? En el primer abatimiento, el pin de I/O era salida-nivel alto hasta que el tiempo **`RCTIME`** lo cambia a entrada. En el Segundo abatimiento, **`PWM`** cambia el pin I/O a entrada cuando termina y entonces hay un breve retraso entre el final del comando **`PWM`** y el inicio del comando **`RCTIME`**. El voltaje empieza a abatirse al final del comando **`PWM`**, entonces es un poco mas bajo que 2.5 V para el momento **`RCTIME`** empieza a medir el tiempo de abatimiento.

Esta reducción en el valor medido puede ser corregida con algunas pruebas, pero no importa porque será el mismo para ambos sensores, izquierdo y derecho. Cuando los programas del Boe-Bot comparen los dos valores de sensores para determinar cuál lado es más brillante o mas atenuado, ambas mediciones serán más bajas en una cantidad pequeña y constante. Entonces, si un sensor detecta menos luz que el otro, su medición aún sera mayor y eso es lo que el programa necesita para las decisiones de la navegación.

Su Turno – Mida el impacto del tiempo de Medición sobre el Servo Control

Puede agregar un par de comandos **`PULSOUT`** para que el Boe-Bot vaya a velocidad plena al frente y probar el efecto del tiempo de medición en el control del servo. Lo primero sería una prueba para averiguar qué tan bajo tiene que ser el nivel de luz antes de que los

servos dejen de funcionar adecuadamente con la solución **HIGH-PAUSE-RCTIME**. Luego, usar los commandos **PWM** en lugar de **HIGH** y **PAUSE**, con un ***Duty*** de 80, y probar para ver qué tan oscuro debe ser antes de que el servo deje de funcionar adecuadamente.

- ✓ Corra TestMaxDarkWithHighPause.bs2.
- ✓ Incremente gradualmente la sombra hasta que los servos empiecen a vibrar. (Una caja de zapatos debe trabajar bien para esto.)
- ✓ Repita con TestMaxDarkWithPwm.bs2. Los servos aún deben vibrar en cierto punto, pero debe ser más oscuro que antes.

Figura 6-15 El programa a la derecha debe permitir que los Servos trabajen en luz más baja

```
' TestMaxDarkWithHighPause.bs2

' {$STAMP BS2}
' {$PBASIC 2.5}

tLeft       VAR     Word
tRight      VAR     Word

PAUSE 1000
DEBUG "Program running... "

DO

  HIGH 6
  PAUSE 1
  RCTIME 6, 1,tLeft

  HIGH 3
  PAUSE 1
  RCTIME 3, 1,tRight

  PULSOUT 13, 850
  PULSOUT 12, 650

LOOP
```

```
' TestMaxDarkWithPwm.bs2

' {$STAMP BS2}
' {$PBASIC 2.5}

tLeft       VAR     Word
tRight      VAR     Word

PAUSE 1000
DEBUG "Program running... "

DO

  PWM 6, 80, 1
  RCTIME 6, 1,tLeft

  PWM 3, 80, 1
  RCTIME 3, 1,tRight

  PULSOUT 13, 850
  PULSOUT 12, 650

LOOP
```

ACTIVIDAD #4: MEDICIONES DE LUZ PARA NAVEGACIÓN

Esta Actividad presenta un programa que automáticamente ajusta a las condiciones de luz en el cuarto y provee información respecto a:

- Cuánta brillantez hay en el cuarto
- Cuál de los dos sensors de luz vé mas sombra
- Qué tanto es el contraste luz/oscuro entre los dos sensores

El Boe-Bot sera capaz de usar esta información para tareas como navegar hacia o lejos de la luz.

El primer programa ejemplo se ve bastante largo pero no se preocupe. Lo podrá descargar de www.parallax.com/go/Boe-Bot en lugar de teclearlo.

Pruebe LightSensorValues.bs2

LightSensorValues.bs2 usa varias subrutinas que condensan las mediciones de luz en dos valores: **`light`**, y **`ndShade`**. La variable luz guarda el nivel de luz ambiental detectado por el Boe-Bot. La variable **`ndShade`** guarda una medición de sombra normalizada y diferencial. Normalizada significa que las mediciones fueron ajustadas a una escala determinada, -500 a 500 en el caso de **`ndShade`**. Diferencial significa que el número corresponde a una diferencia entre las dos mediciones de los sensores. En el caso de **`ndShade`**, el valor indica la diferencia entre el nivel de sombra detectada por cada sensor.

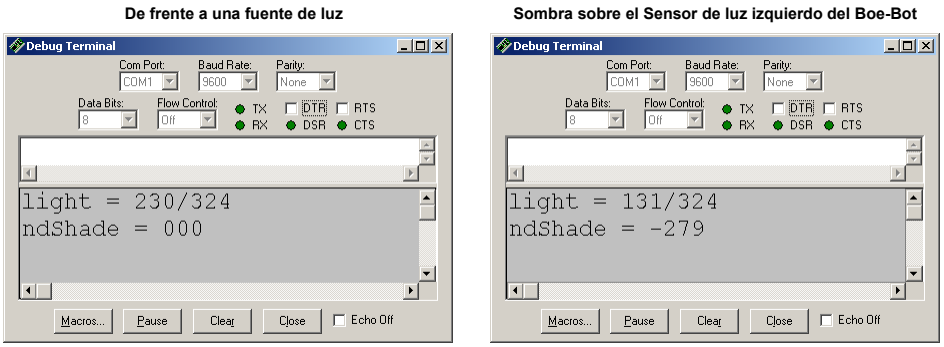
La variable luz es útil para detectar los niveles generales de luz. La escala de la variable es 1 a 324, con 1 siendo la condición más oscura que el sistema puede medir y reportar y 324 la más brillante. Esta medición es útil si una meta en un concurso incluye detectar cuando el robot pasa bajo una luz brillante. Aqui, la variable luz puede guardar un valor mas grande a cualquier otro valor en el curso del robot, y el programa puede usar una declaración **`IF…THEN`** para detectar esa condición y tomar acción.

La variable **`ndShade`** del programa indica cuánta mas sombra detecta un sensor de luz por encima del otro. La escala de la variable es -500 (sombra más oscura a la izquierda) a 500 (sombra más oscura a la derecha). Si el valor de **`ndShade`** es 0, quiere decir que los niveles de luz son parecidos en ambos fototransistores. La medición puede ser útil para un código que haga que el Boe-Bot viaje hacia o lejos de fuentes de luz. Por ejemplo, para hacer que el Boe-Bot viaje hacia la luz, una rutina simplemente tiene que hacer que el Boe-Bot gire si detecta sombra en un lado u otro.

La Figura 6-16 muestra ejemplos de dos condiciones diferentes medidas con LightSensorsValues.bs2. La Terminal de Depuración en la izquierda es un ejemplo del Boe-Bot frente a la principal fuente de luz en un cuarto. La variable luz reporta 230/324, lo cual está en el rango normal de una luz en interiores. La variable **`ndShade`** reporta 0, lo que significa que ambos fototransistores detectan niveles de luz que están muy cercanos uno del otro. La Terminal de Depuración al lado derecho de la Figura muestra

una medición con una sombra cubiendo el sensor de luz izquierdo. El valor de **ndShade** es -279, lo cual indica sombra sobre el sensor izquierdo y que el valor de luz ha caído porque una sombra cubriendo un sensor también reduce la medición de luz total.

Figura 6-16: ejemplo de prueba de sombras con LightSeekingDisplay.bs2



- ✓ Baje LightSensorExamples.zip de www.parallax.com/go/Boe-Bot.
- ✓ Asegúrse de qu no haya luz de sol directa en las ventanas cercanas. La luz de interiores esta bién, pero la luz solar directa cegará a los sensores.
- ✓ Descomprima en un directorio y luego abra LightSensorValues.bs2.
- ✓ Abra el programa con su Editor BASIC Stamp y cárgelo en el BASIC Stamp.
- ✓ Para mejores resultados, ajuste la luz ambiente en el cuarto para que la variable luz esté en el rango de 125 a 275 sin sombra sobre los sensores.
- ✓ Verifique que cuando pone sombra sobre el sensor izquierdo del Boe-Bot, resulte en valores negativos, con sombra más oscura resulte en valores más negativos.
- ✓ Verifique que cuando coloque sombra sobre el sensor derecho del Boe-Bot resulte en valores positivos, con sombra más oscura resulte en valores más postiivos.
- ✓ Verifique que cuando ambos sensores vea aproximadamente el mismo nivel de luz o sombra, que **ndShade** reporta valores cercanos a 0.
- ✓ Verifique que la variable luz caiga con incrementos en sombra y se eleve con más luz.

```
'-----[ Titulo ]-------------------------------------------------------------
' Robotica con el Boe-Bot - LightSensorValues.bs2
' Indica niveles de luz ambiental acondicionada y sombra diferencial en una
' escala de -500 a 500.

' {$STAMP BS2}                              ' Directiva Stamp.
' {$PBASIC 2.5}                             ' Directiva PBASIC.

'-----[ Constantes/Variables ]-----------------------------------------------

Negative       CON     1                   ' Para numeros negativos

' Variables de la Aplicacion
light          VAR     Word                ' indicator Brillo/oscuridad
ndShade        VAR     Word                ' Sombra diferencial Normalizada

' Variables de Subrutina
tLeft          VAR     Word                ' Guarda medicion RCTIME izq
tRight         VAR     Word                ' Guarda medicion RCTIME der
n              VAR     Word                ' Numerador
d              VAR     Word                ' Denominador
q              VAR     Word                ' Cociente
sumDiff        VAR     Word                ' Para calculo de sumas y restas
duty           VAR     Byte                ' variable del argumento PWM duty
i              VAR     Nib                 ' variable de cuenta de Indice
temp           VAR     Nib                 ' Almacen temporal de calculos
sign           VAR     Bit                 ' Var.BIT15 = 1 si neg, 0 si pos

'-----[ Inicializacion ]-----------------------------------------------------

FREQOUT 4, 2000, 3000                      ' Inicia beep

DEBUG CLS                                  ' Limpia la Terminal de Depurac

'-----[ Rutina principal ]---------------------------------------------------

DO
                                           ' Ciclo Principal.
  GOSUB Light_Shade_Info                   ' Obtiene luz & ndShade

  DEBUG HOME, "light = ", DEC3 light,      ' Despliega luz & ndShade
    "/324", CLREOL, CR,
    "ndShade = ", SDEC3  ndShade, CLREOL

  PAUSE 100                                ' Retraso de 0.1 segundos

LOOP                                       ' Repite el ciclo principal

'-----[ Subrutina - Light_Shade_Info ]----------------------------------------
' Usa tLeft y tRight (mediciones RCTIME) y pwm var para calcular:
'   o luz   - nivel de luz Ambiental en una escala de 0 a 324
'   o ndShade – sombra diferencial Normalizada en una escala de -500 a + 500
```

```
'               (-500 -> penumbra a la izq, 0 -> sombra uniforme,
'                +500 -> penumbra a la derecha)

Light_Shade_Info:                           ' Etiqueta de la Subrutina
  GOSUB Light_Sensors                       ' Obtiene medicion gruesa de luz RC
  sumdiff = (tLeft + tRight) MAX 65535      ' Inicia nivel de luz con sum
  IF duty <= 70 THEN                        ' Si duty esta al min
    light=duty-(sumdiff/905) MIN 1          ' Averigua cuanto mas oscuro
    IF sumdiff = 0 THEN luz = 0            ' Si tiempo agoto, max oscuridad
  ELSEIF duty = 255 THEN                    ' Si duty es max
    light=duty+((1800-(sumdiff))/26)        ' Averigua cuanto mas brillante
  ELSE                                      ' Si duty en rango
    luz = duty                             ' luz = duty
  ENDIF                                     ' Termina con nivel de luz
  GOSUB Duty_Auto_Adjust                    ' Ajusta ciclo PWM para sig ciclo
  n = tLeft                                 ' Establece tLeft/(tLeft+tRight)
  d = tLeft + tRight
  GOSUB Fraction_Thousandths                ' Divide (regresa milesimas)
  ndShade = 500-q                           ' Sombra diferencial Normalizada
  RETURN                                    ' Regreso de la subrutina

'-----[ Subrutina - Light_Sensors ]-------------------------------------------
' Mide circuitos sensors de luz P6 y P3. Variable Duty debe estar en 70...255.
' Guarda resultados en tLeft y tRight.

Light_Sensors:                              ' Etiqueta de la Subrutina
  PWM 6, duty, 1                            ' Carga cap en circuito P6
  RCTIME 6, 1, tLeft                        ' Mide abatimiento en P6
  PWM 3, duty, 1                            ' Carga cap en circuito P3
  RCTIME 3, 1, tRight                       ' Mide abatimiento
  RETURN                                    ' Regreso de la subrutina

'-----[ Subrutina - Duty_Auto_Adjust ]----------------------------------------
' Ajusta la variable duty para mantener tLeft + tRight entre 1800 a 2200.
' Requiere la variable sumdiff de tamaño word para los calculos.

Duty_Auto_Adjust:                           ' Etiqueta de la Subrutina
  sumDiff = (tLeft + tRight) MAX 4000       ' Limita el valor max ambiente
  IF sumDiff = 0 THEN sumDiff = 4000        ' Si 0 (timeout) entonces 4000
  IF (sumDiff<=1800) OR (sumDiff>=2200) THEN ' Si fuera de 1800 a 2200
    sumDiff = 2000 - sumDiff                ' Encuentra desviac del objetivo
    sign = sumDiff.BIT15                    ' Pos/neg si .BIT15 = 0/1
    sumDiff = ABS(sumDiff) / 10             ' Max sumDiff sera +/- 10
    sumDiff = sumDiff MAX ((duty-68)/2)     ' Reduce incrementos de ajuste
    sumDiff = sumDiff MAX ((257-duty)/2)    ' fin proximo del rango
    IF sign=NEGATIVE THEN sumDiff=-sumDiff  ' Reestablece signo
    duty = duty + sumDiff MIN 70 MAX 255    ' Limita ciclo a: 70 a 255
  ENDIF                                     ' Fin del if fuera de 1800 a 2200
  RETURN                                    ' Regreso de la subrutina
```

```
'-----[ Subrutina - Fraction_Thousandths ]-----------------------------------
' Calcula q = n/d como un numero de milesimas.
' n y d no deben tener signo y n < d.  Requiere variables Nib para temp & i.

Fraction_Thousandths:                     ' Etiqueta de la Subrutina
  q = 0                                   ' Limpia el cociente
  IF n > 6500 THEN                        ' If n > 6500
    temp = n / 6500                       ' escala n dentro de 0..6500
    n = n / temp
    d = d / temp                          ' escala d con n
  ENDIF
  FOR i = 0 TO 3                      ' division grande en decenas de milesimas
    n = n // d * 10                       ' Multiplica residuo por 10
    q = q * 10 + (n/d)                    ' Agrega siguiente digito al cociente
  NEXT
  IF q//10>=5 THEN q=q/10+1 ELSE q=q/10   ' Redondea q a milesima mas cercana
  RETURN                                  ' Regreso de la subrutina
```


Opcional: Como trabaja LightSensorValues.bs2

Los circuitos fototransistores y las mediciones **RCTIME** plantean 2 problemas. Primero, una medición **RCTIME** para una sombra en un cuarto más oscuro tendrá un valor mayor que el mismo objeto proyectando la misma sombra en un cuarto más brillante. Segundo, en cuartos más oscuros, las mediciones **RCTIME** pueden al final tomar más de los 20 ms que como tiempo libre tiene un programa entre los pulsos de servos.

La Rutina Principal en LightSensorValues.bs2 no tiene que preocuparse de alguno de estos problemas porque la subrutina **Light_Shade_Info** los resuelve. La Rutina Principal hace una sola llamada a la subrutina **Light_Shade_Info** y luego checa los valores de las variables luz y **ndShade** para los 2 valores que necesita para la navegación con un par de sensores de luz. De nuevo, la variable luz indica el nivel general de luz en una escala de 0 a 324, y la variable **ndShade** indica la diferencia luz/sombra entre los sensores en una escala de -500 a 500.

Mas detalle de estas subrutinas: esta sección solo se enfoca en lo que hacen las subrutinas, no en cómo lo hacen. Algunas actividades más avanzadas que narran el desarrollo de las subrutinas están disponibles como descarga en www.parallax.com/go/Boe-Bot. Busque la sección de sensado avanzado de luz.

La subrutina **Light_Shade_Info** llama a la subrutina **Light_Sensors** para obtener las variables **tLeft** y **tRight** que guardan las mediciones **RCTIME**. Note que los comandos **PWM** en la subrutina **Light_Sensors** se apoyan en una variable llamada **duty** para fijar su sensibilidad, la que a su vez controla cuánto le toma a los comandos obtener sus

mediciones de luz. El programa tiene una subrutina llamada **Duty_Auto_Adjust** que automáticamente ajusta la variable **duty** para ayudar a prevenir que cuartos demasiado oscuros deshabiliten los servos del Boe-Bot y que cuartos brillantes ceguen los sensores.

Luego de llamar a la subrutina **Light_Sensors**, la sbrutina **Light_Shade_Info** hace unas cuentas con **tLeft**, **tRight**, y la variable **duty** para calcular el valor de la variable **light**, la cual nuevamente inca el nivel de luz general. Después, llama a la subrutina **Duty_Auto_Adjust**, que ajusta a la variable **duty** para tartar de mantener la suma de las mediciones **RCTIME** en el rango de 1800 a 2200. Cuarto muy oscuros aún causarán que las llantas de los servos vibren en vez que girar y la luz solar directa aún cegará al Boe-Bot, pero **Duty_Auto_Adjust** extenderá significativamente el rango de condiciones de luz que el Boe-Bot podrá ajustar automáticamente y en las que podrá navegar.

Después, la subrutina **Light_Shade_Info** normaliza la diferencia entre los 2 sensores calculando cuánto de la luz total (medida por ambos sensors)solo sensor. Lo hace resolviendo esta ecuación:

$$ndShade = 500 - \left(1000 \times \frac{tLeft}{tLeft + tRight} \right)$$

Esta ecuación resuelve el problema de una sombra teniendo valores diferentes en cuartos con diferentes niveles de luz. Simplemente divide una medición entre la suma de ambas mediciones con un resultado fraccional que va entre 0 a 1. Luego multiplica este por 1000 para un resultado que va entre 0 a 1000. Entonces lo resta a 500, para la variable **ndShade**, que va entre -500 a 500.

Digamos que **tLeft** es 1500 y **tRight** es 500. Esto significa que hay sombra sobre el sensor de luz izquierdo. Si pone los valores en la ecuación, el resultado será -250. Ahora, en un cuarto más oscuro, la misma condición de sombra podría causar que **tLeft** fuera 3600 y **tRight** fuera 1200. Esos valores aún resultarán en un valor **ndShade** de -250.

- ✓ Use la ecuación **ndShade** para calcular ambos valores discutidos en el parrafo anterior.

Quizá también haya notado una nueva y diferente característica en la sección de Constantes/Variables: **Negative CON 1**. Esta is una declaración de una constant y le permite usar un nombre en lugar de un número en su programa. En vez de usar el número 1 en un cierto punto en el programa para revisar y averiguar si un número es negative, el

programa usa la constant **`Nádfegative`**. Entonces, más adelante en la subrutina **`Duty_Auto_Adjust`**, la declaración **`IF sign=Negative THEN sumDiff = -sumDiff`** revisa para averiguar si la variable **`sign`** contiene un 1, indicando que un valor se probó como negativo anteriormente en la subrutina. Esta linea también trabajaría si fuese reescrita como **`IF sign=1 THEN sumDiff = -sumDiff`**.

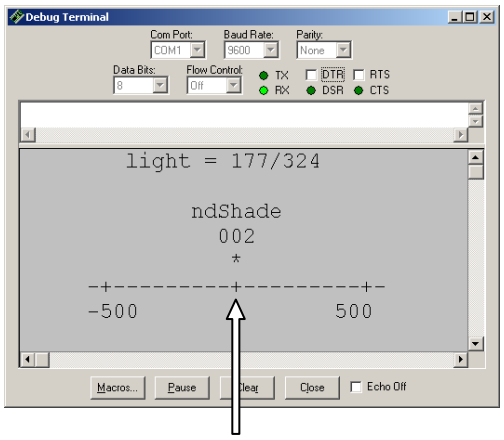
Las constantes pueden ser útiles para ayudar a que los comandos con números en ellas sean más explícitos y también si tiene un número que se usa en varios puntos en un programa. Al actualizar una directiva **`CON`**, todo el código que usa el nombre de la constante usará el valor actualizado. El Capítulo 8 utiliza esto para calibrar un programa que hace que el Boe-Bot siga objetos dentro de un cierto rango de sus sensores infrarrojos de objetos.

Desplegado Gráfico de Mediciones de Luz

La Figura 6-17 muestra un ejemplo de un desplegado gráfico de la variable **`ndShade`**. El asterísco estará en el centro de la escala de -500 a 500 si la luz o la sombra es igual sobre ambos sensores. Si la sombra es más oscura sobre el sensor izquierdo, el asterisco se posicionará a la izquierda de la escala. Si es más oscura sobre la derecha se posicionará hacia la derecha. Un mayor contraste de sombra/luz (como una sombra más oscura sobre uno de los sensores) resultará en el asterisco posicionándose más lejos del centro.

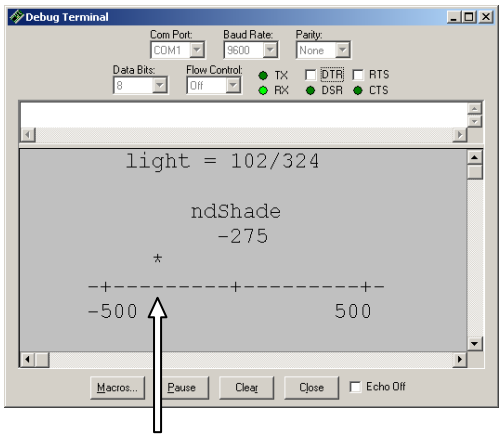
Figura 6-17: Desplegado Gráfico de la variable ndShade

Igual luz/sombra a ambos lados



Asterisco indicador al centro de la escala

Sombra más oscura sobre el lado izquierdo



Asterisco indicador próximo a la mitad entre el centro y la extrema izquierda

Todo lo que necesita para este display son algunas pequeñas modificaciones a las secciones de Inicialización y Rutina Principal de LightSensorValues.bs2. Abajo sigue un ejemplo. Hace uso de algunos nuevos formateadores **DEBUG**, com **REP** y **CRSRX**.

El formateador **REP** repite un character un cierto número de veces. Entonces **DEBUG CLS, REP CR\5** limpia la pantalla y luego escribre 5 retornos de carro, lo que manda el cursos 5 líneas abajo.

El formateador **CRSRX** posiciona el cursor un cierto número de espacios a la derecha del margen izquierdo de la Terminal de Depuración. Por ejemplo, **DEBUG HOME, CRSRX 8** manda el cursor a la posición extrema superior-izquierda de la Terminal de Depuración, luego mueve el cursor ocho espacios a la derecha.

CLREOL es otro nuevo formateador que borra todo a la derecha del cursor en una linea determinada. Esto puede ser útil cuando no necesariamente sabe cuántos dígitos serán desplegados. Si una medición despliega menos dígitos que la anterior, el formateador **CLREOL** borra cualesquiera dígitos fantsmas que pudiesen quedar a la derecha.

```
' Extracto de LightSensorDisplay.bs2

'-----[ Inicializacion ]---------------------------------------------------
FREQOUT 4, 2000, 3000                       ' Inicia sonido

DEBUG CLS, REP CR\5,
      "     -+---------+---------+-",       ' Vista grafica de nivel de sombra
  CR, "     -500               500"

'-----[ Rutina Principal ]-------------------------------------------------
DO                                          ' Ciclo principal.

  GOSUB Light_Shade_Info                    ' Obtiene luz & ndShade
                                            ' Despliega
  DEBUG HOME, CRSRX, 8, "light = ",         ' nombre de variable luz en x=8
      DEC3 light, "/324",CR, CR,            '  valor de la variable light
      CRSRX, 13, "ndShade", CR,             '  titulo sombra en x = 13
      CRSRX,15, SDEC3 ndShade, CLREOL, CR,  '  valor de ndShade en x = 15
      CLREOL, CRSRX,                        '  despliega asterisco en ndShade
      6+((ndShade+500)/50), "*"             '  posicion-x

  PAUSE 100                                 ' Retraso de 0.1 segundos

  LOOP                                      ' Repite ciclo principal
```

- ✓ LightSensorDisplay.bs2 fue otro ejemplo en LightSensorExamples.zip. Ábralo con el Editor BASIC Stamp.
- ✓ Si prefiere salvar LightSensorValues.bs2 como LightSensorDisplay.bs2 y teclear los cambios, asegúrese dejar 5 espacios entre las comillas y los primeros caracteres en cada escala. Por ejemplo, hay 5 espacios entre las comillas y el primer guión en **`CR, "     -500…`**
- ✓ Recuerde, para mejores resultados, asegúrese ajustar el area iluminada para que la Terminal de Depuración despliege valores en el rango entre 125 a 175 sin sombras sobre los fototransistores.
- ✓ Corra el programa y proyecte diferentes niveles de sombra sobre cada sensor de luz y observe cómo responde el asterisco de la Figura 6-17. Recuerde que si proyecta la misma sombra sobre ambos sensores, el asterisco debe permanecer en medio, solo indica cuál sensor ve mas sombra si hay diferencia entre ellos.

ACTIVIDAD #5: RUTINA PARA VIAJAR HACIA LA LUZ

Una forma de hacer que el Boe-Bot viaje hacia fuentes de luz es hacer que se aleje de la sombra. Incluso puede usar la variable **`ndShade`** para hacer que gire más o menos cuando el contraste entre la luz detectada a cada lado sea mas o menos. Primero, necesitamos un par de variables para guardar variables de duración de pulso para los servos.

```
' aplicación Variables
pulseLeft      VAR     Word
pulseRight     VAR     Word
```

Después, necesitamos código para establecer esos valores de pulso. El código que sigue parametriza **`pulseLeft`** y **`pulseRight`** para mantener la rueda que este en sombra a velocidad plena y reducir la velocidad o girar en reversa la otra. Cuando el contraste entre las mediciones de luz y sombra es pequeño, la rueda que no está bajo sombra solo reduce su velocidad para lograr una vuelta gradual. Cuando el contraste es mayor, la rueda al otro lado del lado oscuro puede reducir su velocidad aún más, o incluso empezar a girar en reversa para que el Boe-Bot ejecute una vuelta más cerrada para alejarse de la sombra.

```
' Rutina de navegacion
IF (ndShade + 500) > 500 THEN
  pulseLeft = 900 - ndShade MIN 650 MAX 850
  pulseRight = 650
ELSE
  pulseLeft = 850
  pulseRight= 600 - ndShade MIN 650 MAX 850
ENDIF
```

La rutina que establece los valores de **`pulseLeft y pulseRight`** empieza por decider si la sombra esta sobre el sensor izquierdo o derecho, comparando (**`ndShade + 500`**) contra **`500`**. Los operadores > (mayor que), >= (mayor o igual que), < (menor que), y <= (menor o igual que) solo compara dos números positivos. Puesto que el menor valor posible de **`ndShade`** es -500, el código en la condición **`IF`** agrega 500 a **`ndShade`** y luego lo compara contra 500. Es el equivalente PBASIC a **`IF ndShade > 0`**.

Digamos que **`ndShade`** es 125, lo que significa que definitivamente hay sombra sobre el sensor de luz derecho. **`IF ndShade + 500 > 500 THEN`** checa si 625 es mayor que 500, que lo es. La Figura 6-18 muestra lo que pasa después cuando el código reduce la velocidad de la rueda izquierda con **`pulseLeft = 900 – ndShade MIN 650 MAX 850`**, y establece la rueda derecha a velocidad plena al frente con **`pulseRight = 650`**. Puesto que **`ndShade`** es 125 en este ejemplo, 900 –125 = 775, lo que causaría un comando **`PULSOUT`** para reducir la velocidad de la rueda izquierda.

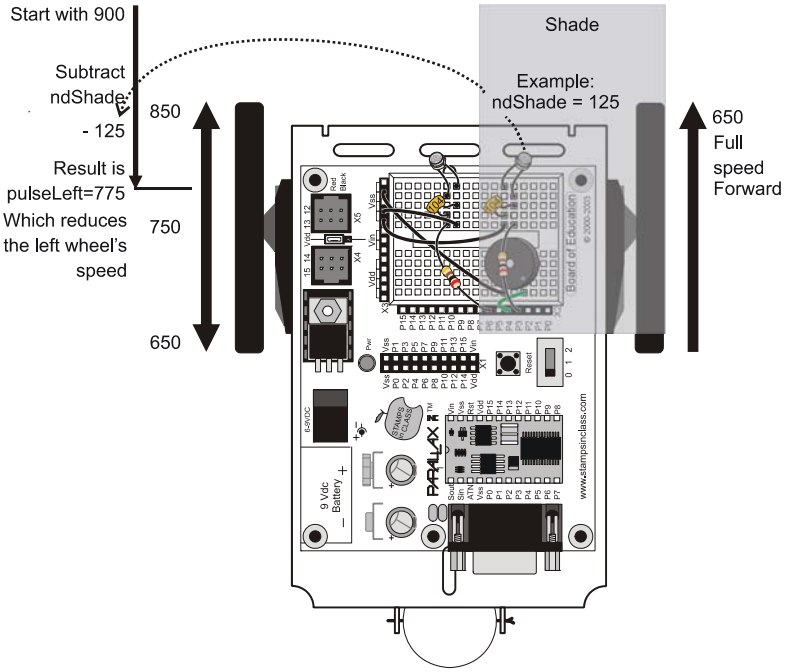


Figura 6-18
Reacción de LightSeekingBoe-Bot.bs2's a la sombra a la derecha

En este ejemplo, una medición ndShade de 125 se sustrae de 900, y the 775 resultante reduce la velocidad del servo izquierdo.

Si **ndShade** es mayor, como por ejemplo 190, lo que significa que la sombra sobre el sensor derecho es más oscura, **pulseLeft** termina con un valor de 710, lo que hará que la rueda izquierda gire en reversa para una vuelta much más cerrada. Para valores de **ndShade** mayores que 250, la expresión **900 - ndShade** puede resultar en valores menores que 650. De modo semejante, para valores de **ndShade** entre 1 y 49, la expresión puede resultar en valores por arriba de 850. Luego, el código usa los operadores **MIN** y **MAX** para mantener el resultado en el rango de 650 a 850 aún cuando **900 - ndShade** pueda tener resultados intermedios fuera del rango.

El operador **MIN** toma un resultado por debajo del valor especificado y lo incrementa a ese valor, pero deja intactos resultados por encima del valor **MIN**. Entonces, si el resultado de **600 - ndShade** es cualquier cosa por debajo de 650, el operador **MIN** guarda 650 en **pulseLeft**.

Por ejemplo, si **ndShade** fuese 350, el resultado intermedio de **900 - ndShade** sería 550, pero **MIN 650** lo cambiaría por 650. Similarmente, el operador **MAX** toma un resultado que este por encima del valor especificado y lo decrement a ese valor, pero deja intactos resultados inferiores al valor **MAX**. Asi pues, aún cuando valores entre 0 a 49 entregarían resultados intermedios **900 - ndShade** en el rango de 900 a 851, **MAX 850** establece en 850 cualquier resultado en dicho rango.

Para valores **ndShade** cero o menores, esto quiere decir que la sombra esta sobre el sensor izquierdo y que la rueda derecha necesita reducir su velocidad. El código en el bloque **ELSE** lo hace al fijar la rueda izquierda a velocidad plena con **pulseLeft = 850** para hacer que la rueda izquierda del Boe-Bot vaya a velocidad plena al frente y **pulseRight = 600 - ndShade MIN 650 MAX 850** para reducir la velocidad o incluso revertir la dirección de la rueda derecha del Boe-Bot, dependiendo de que tan oscura es la sombra sobre el sensor de luz izquierda.

Rutina de navegación de prueba con la Terminal de Depuración

La Figura 6-19 muestra algunos ejemplos de desplegado de la Terminal de Depuración a partir del siguiente programa ejemplo, LightSeekingDisplay.bs2. Las instrucciones mas adelante le pedirán que corra el programa, pero primero dé un vistazo a los desplegados de la Terminal de Depuración en la figura.

Estas pantallas demuestran cómo la rutina de navegación ajusta las variables **`pulseLeft`** y **`pulseRight`** en respuesta a valores **`ndShade`** diferentes. El programa hace que la Terminal de Depuración despliegue una vista superior del Boe-Bot con etiquetas **`pulseLeft`** y **`pulseRight`** y sus valores cerca a cada rueda. El programa también posiciona > a la izquierda y < a la derecha como indicadores de velocidad de las ruedas para mostrar qué tan rápido y en qué dirección gira cada rueda.

Por ejemplo, en la Terminal de Depuración superior izquierda, ambos indicadores de velocidad de rueda están a la par con una etiqueta “forward”, lo que significa que el Boe-Bot estaría yendo a velocidad plena hacia al frente.

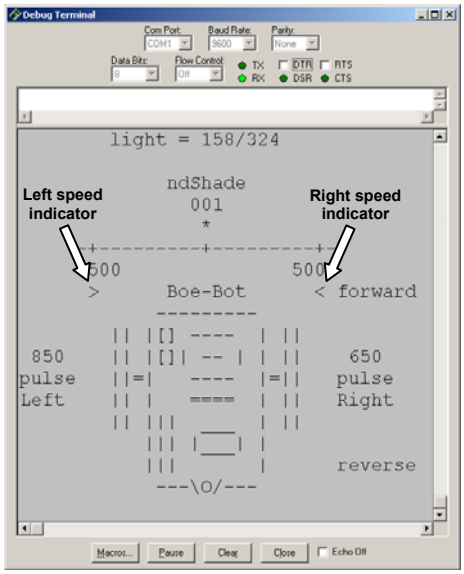
En la Terminal de Depuración superior derecha, el indicador de velocidad de rueda derecha está a la mitad entre las etiquetas “forward” y “backward” lo que significa que dicha rueda se detendrá.

En la Terminal de Depuración inferior izquierda, el indicador de velocidad de rueda izquierda está a la par con la etiqueta “reverse”, lo que significa que la izquierda estaría girando a velocidad plena en reversa.

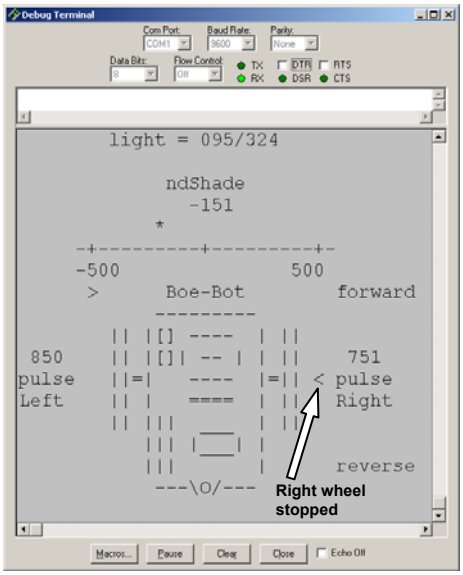
En la inferior derecha, la variable luz es menor. Puesto que **`ndShade`** esta cercana a cero, el nivel de sombra es aproximadamente el mismo y debe haber sombra sobre ambos sensors. Puesto que el Boe-Bot solo responde a diferencias en sombras, la misma sombra sobre ambos sensores mantendrá girando ambas ruedas a velocidad plena al frente nuevamente.

Figura 6-19: Ejemplos de indicadores de sombra y velocidad de rueda

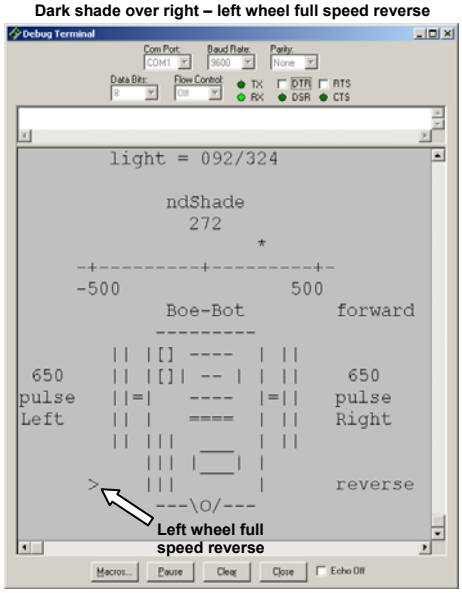
Equal luz – full speed forward



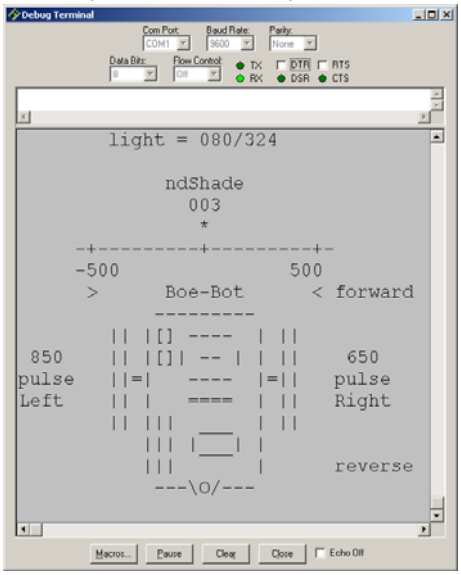
Shade over left – slow down right wheel



Dark shade over right – left wheel full speed reverse



Equal shade – back to full speed forward



6

LightSeekingDisplay.bs2 es otro ejemplo en el archivo LightSensorExamples.zip. Se expande en LightSensorDisplay.bs2 con las siguientes características:

- Declaraciones de variables de tamaño Word para **`pulseLeft`** y **`pulseRight`**
- Un comando **`DEBUG`** que despliega una vista superior del Boe-Bot
- El bloque de código **`IF…THEN…ELSE…ENDIF`** de la rutina de búsqueda de luz
- Comandos Debug que despliegan los valores de las variables **`pulseLeft`** y **`pulseRight`** cerca a cada rueda.
- Comandos **`DEBUG`** que posicionan los indicadores de velocidad de rueda izquierda > y derecha < para mostrar la velocidad y dirección de cada rueda.

LightSeekingDisplay.bs2 es un excelente programa para observer las respuestas de las variables **`pulseLeft`** y **`pulseRight`** a sombras sobre cada sensor y como afectan esos valores a la velocidad de rueda.

- ✓ Abra LightSeekingDisplay.bs2 con el Editor BASIC Stamp y descárguelo al BASIC Stamp.
- ✓ Nuevamente, para mejores resultados, ajuste la luz ambiente en el cuarto para que la Terminal de Depuración despliegue un valor de la variable luz en el rango de 125 a 275 sin sombra sobre los sensores.
- ✓ Experimente con mas y menos sombra sobre cada sensor y ponga especial atención en cómo afecta al valor **`ndShade`**, que a su vez afecta a las variables **`pulseLeft`** y **`pulseRight`** y las velocidades de rueda que establecerían si fuesen usadas en comandos **`PULSOUT`**.

```
' Extractos de LightSeekingDisplay.bs2

'   ...  (los puntos suspensivos indican codigo omitido)

' Variables de la Aplicación
pulseLeft      VAR     Word                 ' duracion de pulso servo izq
pulseRight     VAR     Word                 ' duracion de pulso servo der
servo pulse duration

'   ...
```

```
'-----[ Inicializacion ]-----------------------------------------------------
FREQOUT 4, 2000, 3000                        ' Inicia beep

DEBUG CLS, REP CR\5,
      "     -+---------+---------+-",        ' Vista grafica nivel de sombra
  CR, "     -500               500",
  CR, "             Boe-Bot        forward",
  CR, "            ---------       ",        ' vista superior del Boe-Bot con
  CR, "        || |[] ----  | ||   ",        ' etiquetas de pulso
  CR, "        || |[]| -- | | ||   ",
  CR, "pulse   ||=|   ----  |=||   pulse ",
  CR, "Left    || |   ====  | ||   Right ",
  CR, "        || |||  ___  | ||   ",
  CR, "           ||| |___| |      ",
  CR, "           |||       |      reverse",
  CR, "            ---\O/---       "

'      0123456789    +10   0123456789    +30 ' posiciones de cursor como guia
'          +0    0123456789   +20    01234567'  si teclea el codigo
'-----[ Rutina Principal ]---------------------------------------------------
DO                                           ' Ciclo Principal.

  GOSUB Light_Shade_Info                     ' Obtiene luz & ndShade
                                             ' Despliega
  DEBUG HOME, CRSRX, 8, "light = ",          '  nombre variable de luz en x=8
      DEC3 light, "/324",CR, CR,             '  valor variable de luz
      CRSRX, 13, "ndShade", CR,              '  titulo sombra en x = 13
      CRSRX,15, SDEC3 ndShade, CLREOL, CR,   '  valor ndShade en x = 15
      CLREOL, CRSRX,                         '  despliega asterisco en ndShade
      6+((ndShade+500)/50), "*"              '   posicion-x

  ' Rutina de navegacion
  IF (ndShade + 500) > 500 THEN              ' Si hay mas sombra a la der...
    pulseLeft = 900 - ndShade MIN 650 MAX 850' Alenta rueda izq c/sombra der
    pulseRight = 650                         ' Rueda der vel plena al frente
  ELSE                                       ' Si hay mas sombra a la izq...
    pulseLeft = 850                          ' Rueda izq vel plena al frente
    pulseRight= 600 - ndShade MIN 650 MAX 850' Alenta rueda der c/sombra izq
  ENDIF

  DEBUG CRSRXY, 1, 10, DEC3 pulseLeft,  ' Muestra valores de variable de pulso
        CRSRX, 29, DEC3 pulseRight           ' arriba de nombres de variables

  FOR i = 7 TO 15                            ' Limpia areas donde indicadores
    DEBUG CRSRXY, 6, i, " ", CRSRX, 26, " "  ' > y < de vel y direcc de rueda
  NEXT                                       ' pudieran estar colocados

  DEBUG CRSRXY,6,15-((pulseLeft-650)/25),">" ' Coloca nuevos indicadores de
  DEBUG CRSRXY,26,7+((pulseRight-650)/25),"<"' vel y direcc de rueda

LOOP                                         ' Repite ciclo principal
' ...
```

LightSeekingDisplay.bs2 utiliza otro formateador **DEBUG**, **CRSRXY**, para posicionar el cursor y desplegar cada indicador de velocidad de rueda. **CRXRXY** debe ser seguido por 2 números. Por ejemplo, **DEBUG CRXRXY, 6, 11, ">"** deplegaría el caracter > a 6 espacios del margen izquierdo de la Terminal de Depuración y a 11 retornos de carro hacia abajo desde su parte más alta. El programa de hecho usa una expresion para establecer el número de retornos de carro para positicionar el cursor. Así, el comando:

```
DEBUG CRSRXY,6,15-((pulseLeft-650)/25),">"
```

...posiciona el cursor 6 espacios de la extrema izquierda de la Terminal de Depuración, pero usa una expresion para escoger el número de retornos de carro a partir de la línea más alta de la Terminal de Depuración. Digamos que **pulseLeft** es 750. Luego, la posición y sería 11 porque 15 – ((750 – 650)/25 = 15 – 4 = 11. En ese caso, **CRSRXY** posiciona el cursor en 6, 11, y luego escribe el carácter ">".

Su Turno – Ahorre gran cantidad de RAM

Si quisiera agregar otras características a su Boe-Bot además de la búsqueda de luz podría ser difícil si su aplicación sobrepasa la RAM tan solo al intentar declarar agunas cuantas variables mas. El lado izquierdo de la Figura 6-20 muestra el problema, a la aplicación le restan ligéramente menos de 2 words. El lado derecho de la Figura muestra cuanto espacio puede ahorrar usando una simple técnica llamada asignación de alias a variables.

- ✓ Para ver el Mapa RAM para LightSeekingDisplay.bs2, haga click en el botón Memory Map; está justo a la izquierda del botón Run. También puede desplegarlo haciendo click en Run → Memory Map, o presionando CTRL + M. Su Mapa RAM debe semejarse al de la izquierda de la Figura 6-20.

De acuerdo con la ayuda de BASIC Stamp, un alias es “un nombre alternativo para una variable existente.”

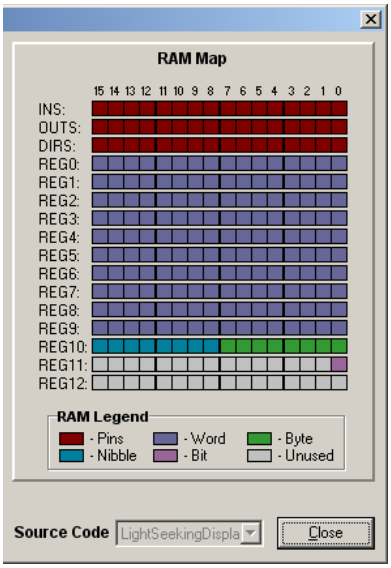
- ✓ En el Editor BASIC Stamp, haga click en Help y seleccione BASIC Stamp Help.
- ✓ Haga click en PBASIC Language referencia y luego en Variables para desplegar la página de Variables. Encuentre la explicación de alias, léala y examine el ejemplo de declaración de variable que usa alias.

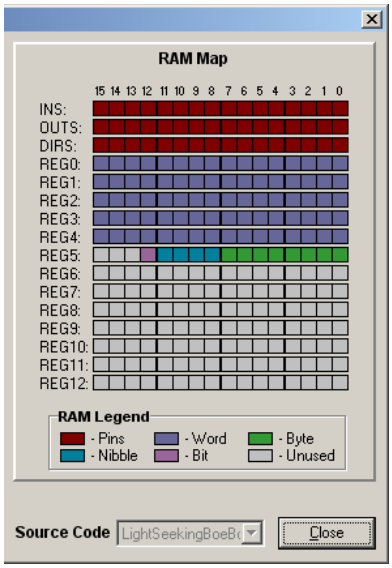
No todas las variables en LightSeekingDisplay.bs2 son usadas al miosmo tiempo. Por ejemplo, el programa termina de usar **tLeft** y **tRight** luego de que la subrutina **Light_Shade_Info** ha terminado. Más aún, nunca usa aquellas 2 variables al mismo

tiempo que usa **`pulseLeft`** y **`pulseRight`**. Entonces **`tLeft`** puede ser declarada como un alias de **`pulseLeft`** y **`tRight`** puede ser declarada como un alias de **`pulseRight`**. Ahora, **`pulseLeft`** y **`tLeft`** usan el mismo espacio de memoria, lo mismo que **`pulseRight`** y **`tRight`**, y su aplicación recuperó dos words de RAM.

Con las modificaciones en LightSeekingDisplayBetterRAM.bs2, el programa reduce el uso de RAM de "casi todo" a "menos de la mitad."

Figura 6-20: La Asignación de Alias a variables ahorra casi la mitad de la RAM del BASIC Stamp





- ✓ Salve LightSeekingDisplay.bs2 como LightSeekingDisplayBetterRAM.bs2.
- ✓ Actualice las declaraciones de variables en LightSeekingDisplay.bs2 para que igualen el lado derecho de la Figura 6-21.
- ✓ Revise nuevamente su Mapa de Memoria. Debe semejarse al de la derecha de la Figura 6-20.

Figura 6-21 Ahorrando espacio con la asignación de Alias a Variables

```
' Variables de la Aplicacion
pulseLeft       VAR     Word
pulseRight      VAR     Word
light           VAR     Word
ndShade         VAR     Word

' Variables de Subrutina
tLeft           VAR     Word
tRight          VAR     Word
n               VAR     Word
d               VAR     Word
q               VAR     Word
sumDiff         VAR     Word
duty            VAR     Byte
i               VAR     Nib
temp            VAR     Nib
sign            VAR     Bit
```

```
' Variables de la aplicacion
pulseLeft       VAR     Word
pulseRight      VAR     Word
light           VAR     Word
ndShade         VAR     Word

' Variables de Subrutina
tLeft           VAR     pulseLeft
tRight          VAR     pulseRight
n               VAR     tLeft
d               VAR     Word
q               VAR     ndShade
sumDiff         VAR     d
duty            VAR     Byte
i               VAR     Nib
temp            VAR     i
sign            VAR     Bit
```

CUIDADO: Sea cuidadoso en cómo usa la asignación de alias a variables. Si el programa necesita 2 variables al mismo tiempo, una variable no puede ser un alias de la otra. De modo semejante, si el programa necesita checar el valor previos de una variable en la siguiente eteración de un ciclo, asignar un alias y usarlo para otro porpósito borraría un valor que su programa necesitará más tarde.

Por ejemplo, si intentase hacer a **`pulseLeft`** un alias de **`pulseRight`**, ambas velocidades de sus servos serían iguales siempre. No podrían ser los dos valores independientes que su código necesita como servo control.

ACTIVIDAD #6: RUTINA DE PRUEBA DE NAVEGACIÓN CON EL BOE-BOT

El siguiente extracto de código de LightSeekingBoeBot.bs2 tiene una versión de solo-navegación de las rutinas de Inicialización y Principal de LightSeekingDisplayBetterRAM.bs2. Todos los comandos **`DEBUG`** han sido removidos junto con el comando **`100 ms PAUSE`**. Todos fueron remplazados con comandos **`PULSOUT`** que usan las variables **`pulseLeft`** y **`pulseRight`** para controlar los servos.

```
'-----[ Inicializacion ]-------------------------------------------------------
FREQOUT 4, 2000, 3000                       ' Inicia beep
DEBUG "Program running..."                  ' Mensaje de programa corriendo

'-----[ Rutina Principal ]------------------------------------------------------
DO                                          ' Ciclo Principal.

  GOSUB Light_Shade_Info                    ' Obtiene luz & ndShade

   ' Rutina de navegacion
  IF (ndShade + 500) > 500 THEN             ' Si mas sombra a la derecha...
    pulseLeft = 900 - ndShade MIN 650 MAX 850' Alenta rueda izq c/sombra der
    pulseRight = 650                        ' Rueda der vel plena al frente
  ELSE                                      ' Si mas sombra a la izquier...
    pulseLeft = 850                         ' Rueda izq vel plena al frente
    pulseRight= 600 - ndShade MIN 650 MAX 850' Alenta rueda der c/sombra izq
  ENDIF

  PULSOUT 13, pulseLeft                     ' Pulso de control de servo izq
  PULSOUT 12, pulseRight                    ' Pulso de control de servo der

LOOP                                        ' Repite ciclo principal
```

- ✓ Abra LightSeekingBoeBot.bs2 con el Editor BASIC Stamp y cárguleo a su BASIC Stamp.
- ✓ O salve LightSeekingDisplayBetterRAM.bs2 como LightSeekingBoeBot.bs2 y actualice las Rutinas Principal y de Inicialización para que concuerden con el extracto de arriba.
- ✓ Corra el programa.
- ✓ Si tiene un Board of Education, coloque el switch de 3-posiciones en 2 después de haber desconectado el Boe-Bot de su cable de programación y póngalo donde quiere que empiece a viajar.
- ✓ Su Boe-Bot puede ahora viajar hacia la luz.
- ✓ Intente proyectar sombras sobre los sensores de luz de su Boe-Bot conforme viaja. Debe alejarse de las sombras.
- ✓ Intente enviar su Boe-Bot hacia una sombra oscura proyectada por un escritorio. Asegúrese de que se aproxime en ángulo y no en dirección de frente a la sombra. Debe bordearla.
- ✓ Intente llevar el Boe-Bot en un cuarto con baja iluminación y con una luz brillante en la puerta. ¿Puede encontrar la salida?

Corrección de Problemas

Si el Boe-Bot parece un poco menos sensible a la luz en un lado, intente corregirlo siguiendo las Instrucciones en la siguiente sección (Su Turno – Ajustes de Sensibilidad de Luz/Sombra). Lo mismo aplica si quiere que el Boe-Bot sea mas o menos sensible a la sombra.

Si el Boe-Bot no responde a las sombras alejándose de ellas, o si gira en su lugar en lugar de viajar, siga los siguientes pasos:

- Si tecleó su código, pruebe el código ejemplo LightSeekingBoeBot.bs2 en LightSensorExamples.zip, una descarga gratis en www.parallax.com/go/Boe-Bot. Esto cancelará los errores de código antes de examiner su circuito.
- Si el código del sitio web de Parallax no corrige el problema, tendrá que revisar el circuito. Comience por verificar cuidadosamente todas las conexiones de su circuito según el esquemático y el diagrama de alambrado en la Actividad #2. Vuelva a revisar el código de color de la Resistencia (café-negro-rojo), numerous en el capacitor (104), la longitud de patas del fototransistor, y asegúrese de que todas las terminales están conectadas como se muestra en el diagrama de alambrado. También asegúrese de veriricar que seleccionó los fototransistores y no los LEDs infrarrojos con la ayuda de la Figura 6-8 en la página 180. Verifique que los fototransistores apuntan arriba y afuera como en la Figura 6-10 en la página 183. A veces, apuntarlos ligeramente hacia afuera mejora la respuesta a la sombra. Al ajustar la dirección de sus fototransistores asegúrese de que sus patas no se toquen una a la otra. Repita las pruebas de la Terminal de Depuración y asegúrese de hacerlo para ambos sensores. Cada uno debe responder similarmente a la luz y a la sombra.
- Después, repita las pruebas en la Actividad #5. Use la Terminal de Depuración para verificar que los niveles de luz están en el rango de 125 a 275. También, proyectar una sombra sobre un sensor dado deberá desacelerar el servo en el lado contrario. Una sombra similar sobre el otro sensor deberá resultar en un ajuste de velocidad de motor similar en la otra rueda. También verifique que el mismo nivel de sombra o luz sobre ambos sensores resulte en velociad al frente plena.
- Si la Terminal de Depuración despliega un valor de **ndShade** muy lejano a cero, aún cuando los sensores vean aproximadamente el mismo nivel de luz, hay algunas pruebas adicionales que puede probar en la Actividad de chequeo de fototransistor. Es parte de las actividades avanzado de sensado de luz disponibles en www.parallax.com/go/Boe-Bot. Si la Terminal de Depuración indica que los sensores y servos están ambos respondiendo a la luz y sombra correctamente, intente LightSeekingBoeBot.bs2 nuevamente. Si aún no responden correctamente a las sombras, es momento de revisar sus servo motores. Repita la Actividad #6 del Capítulo 2. Para mayor profundidad, haga las gráficas de la Actividad #4 Capítulo 3, para ambos servos.

For mas ayuda en la corrección de problemas, pruebe los recursos en www.parallax.com/support.

Su Turno – Ajustes de Sensibilidad de Luz/Sombra

Puede cambiar el 900 y el 600 en estas lineas para hacer al Boe-Bot mas o menos sensible a la sombra:

```
pulseLeft = 900 - ndShade MIN 650 MAX 850
```

...y:

```
pulseRight= 600 - ndShade MIN 650 MAX 850
```

Por ejemplo, puede incrementar la sensibilidad a la luz/sombra del Boe-Bot cambiando el 900 a 875 y cambiando 600 por 625. Puede hacerlo mas sensible cambiando el 900 a 850 y el 600 por 650. De modo semejante, puede hacer todo el sistema menos sensible cambiando el 900 a 925, y el 600 por 575, etc…

- ✓ Inténtelo.

También puede ajustar uno de los valores para hacer uno u otro sensors izquierdo o derecho mas sensible. Cambiando el 900 por otro valor cambiará la sensibilidad del Boe-Bot a la sombra en la izquierda, mientras que cambiar el 600 a otro valor cambiará la sensibilidad del Boe-Bot a la sombra en la derecha.

- ✓ Inténtelo también.

Otras cosas que puede hacer con ajustes mínimos a la Rutina Principal son:

- Seguir la sombra en vez de la luz con **`ndShade = -ndShade`** justo después de la llamada de la subrutina **`Light_Shade_Info`** en la Rutina Principal.
- Finalizar el desplazamiento bajo una luz brillante o en un area oscura detectando condiciones de luz o sombra extrema con la variable luz con una declaración **`IF…THEN`**.
- Funcionar como una brújula de luz mateniéndose estacionario y girando sobre sí hacia fuentes brillantes de luz.
- Incorporar filamentos en la navegación al frente hacia la luz para que el Boe-Bot pueda detectar y rodear objetos en su camino.

RESUMEN

Un fototransistor es una válvula de corriente controlada por la luz. Permite el paso de mas corriente con incidencia de luz más brillante y menos corriente con luz menos brillante. Este Capítulo utilizó 2 circuitos fototransistores diferentes para detectar luz: un circuito de voltaje de salida y un circuito de transferencia de carga.

The circuito fototransistor de voltaje de salida en este Capítulo fué conectado a un pin I/O configurado como entrada para dar un valor binario que indicaba brillo o luz ambiental. Cuando el fototransistor permite mayor paso de corriente, el voltaje a través de la resistencia es menor. Escogiendo la rsistencia adecuada para las condiciones de luz, el circuito puede ser monitoreado por un pin I/O porque su voltaje será superior a 1.4 V en luz brillante y menor que 1.4 V en luz ambiente. El registro de entrada del pin I/O guarda un 1 cuando el voltaje es mayor que 1.4 V y un 0 cuando es menor que 1.4 V.

Un par de circuitos de transferencia de carga fueron usados para medir diferencias en la intensidad de luz entre los fototransistores izquierdo y derecho, y el Boe-Bot fue programado para detectar y actuar según las diferencias. El circuito de transferencia de carg consistió de un capacitor y un fototransistor conectados en paralelo a un pin I/O con una resisencia. En este circuito, el BASIC Stamp usó un pin I/O para cargar el capacitor. Luego, cambió de modo al pin I/O de salida a entrada y midió el tiempo que le tomó al voltaje del capacitor abatir su carga a través del fototransistor. Esta medición del tiempo resulta ser menor con luz brillante y mayor en la sombra.

Un comando **`HIGH`** seguido por una **`PAUSE`** pueden cargar el capacitor y luego el comando **`RCTIME`** cambia el pin I/O a entrada y mide el tiempo que le toma al voltaje del capacitor abatirse a 1.4 V al perder su carga a través del fototransistor. La medición de tiempo puede ser reducida usando el comando **`PWM`** en lugar de **`HIGH`** y **`PAUSE`**. El comando **`PWM`** puede cargar el capacitor a valores menores a 5 V antes que el comando **`RCTIME`** y entonces el capacitor tiene menos volts que abatir antes de alcanzar 1.4 V, y esto toma menos tiempo. La reduccion de tiempo ayuda a prevenir que el retraso entre comandos **`PULSOUT`** se haga tan largo que haga a los servos vibrar en vez de girar.

Las Actividad #4 a Actividad #6 utilizan una colección de subrutinas que entrega a la Rutina Principal valores de los niveles de luz general junto con el contraste de luz/sombra entre los 2 sensores. Este contraste de luz/sombra entre los sensores es llamado una medición diferencial, y las subrutinas también normalizan la medición a una escala de -500 a 500. Las mediciones de abatimiento reales pueden variar dependiento de los niveles de luz ambiental, entonces los valores normalizados mantienen estas mediciones en una escala que es útil para el servo control ypara la navegación del Boe-Bot hacia las fuentes de luz.

Preguntas

1. ¿Qué es lo que regula un transistor?
2. ¿Cuáles son las terminales del fototransistor que tienen patas?
3. ¿Cómo puede usar la cara plana del encapsulado plástico del fototransistor para identificar sus terminales?
4. ¿A qué color sera más sensible el fototransistor: rojo o verde?
5. ¿Cómo respond V_{P6} en la Figura 6-6 si la luz se hace más brillante?
6. ¿Qué es lo que hace el fototransistor en la Figura 6-6 que causa que V_{P6} se incremente o decremente?
7. ¿Cómo puede ser modificado el circuito en la Figura 6-6 para hacerlo más sensible a la luz?
8. ¿Qué pasa cuando el voltaje aplicado a un pin I/O que ha sido configurado como entrada esta arriba o abajo del voltaje de umbral?
9. Si la cantidad de carga que un capacitor guarda se decrementa, ¿qué pasa con el voltaje en sus terminales?

Ejercicios

1. Encuentre V_{P6} si I = 1 mA en la Figura 6-6.
2. Calcule la corriente en la Resistencia si V_{P6} es 4.5 V en la Figura 6-6.
3. Asuma que el umbral entre luz y sombra que necesita su aplicación ocurre cuando V_{P6} = 2.8 V. Calcule el valor de Resistencia que necesita para que el BASIC Stamp detecte este umbral.
4. Calcule el valor de un capacitor marcado con 105.
5. Escriba un comando **`RCTIME`** que mida el tiempo de abatimiento con el pin I/O P7 y que guarde el resultado en una variable llamada **`tDecay`**.
6. Escriba un **`comando PWM`** que carge el capacitor en la Figura 6-11 hasta aproximadamente 1.625 V para preparar el circuito para una medición de abatimiento.

7. Calcule cuál sería la medición de **ndShade** si el BASIC Stamp mide valores de abatimiento de 1001 en ambos lados.
8. Escriba un **comando DEBUG** que despliege 50 caracteres del signo igual.
9. Escriba un **comando DEBUG** que posicione el caracter "#" ocho espacios a la derecha del margen izquierdo de la Terminal de Depuración y 10 retornos de carro hacia abajo a partir de la extrema superior.

Proyectos

1. En la Actividad #1, el circuito junto al código ejemplo en la sección Su Turno hace que el Boe-Bot se detenga bajo una luz al final de la ruta. ¿Qué pasaría si tuviera una cantidad limitada de tiempo para recorrer el tramo y no conociera las condiciones de iluminación con anterioridad? Quizá tendría que calibrar su Boe-Bot allí mismo. Un programa que haga que el piezoparlante suene repetídamente cuando el Boe-Bot detecte luz brillante y se quede callado cuando detecte luz ambiental podría ser útil para esta tarea. Escriba y pruebe el programa que trabajo con el circuito en la Figura 6-4 en la página 173.
2. Desarrolle una aplicación que haga que el Boe-Bot navegue y busque oscuridad en vez de luz. Esta aplicación debe utilizar los circuitos de transferencia de carga en la Figura 6-9 en la página 181.
3. Desarrolle una aplicación que haga que el Boe-Bot navegue hacia una lámpara de escritorio incandescente brillante en un cuarto donde las únicas otras fuentes de luz son lámparas fluorescentes de techo. El Boe-Bot debe ser capaz de navegar hacia la lámpara de escritorio y tocar un tono cuando esté bajo ella. Esta aplicación debe utilizar los circuitos de transferencia de carga de la Figura 6-9 en la página 181.

Soluciones

Q1. La cantidad de corriente que permite entrar en su colector y salir por su base.
Q2. Las terminales colectora y emisora del fototransistor están conectadas a las patas.
Q3. La pata que está más cerca a la cara plana es el emisor. La pata más lejana a la cara plana es el colector.
Q4. La longitude de onda del rojo está más cercana a la longitude de onda del infrarrojo, entonces debería ser más sensible al rojo.
Q5. V_{P6} incrementa con más luz.
Q6. Suministra más o menos corriente a la resistencia.
Q7. Cambie la Resistencia de 2 kΩ por una de mayor valor.

Q8. Si el voltaje aplicado es mayor que el voltaje de umbral, el bit de registro de entrada para ese pin guarda un 1. Si está debajo del voltaje de umbra, el bit de registro de entrada guarda un 0.
Q9. El voltaje se decrementa.

E1. $V = I \times R = 0.001\ A \times 2000\ \Omega = 2\ V$.
E2. $V = I \times R \rightarrow I = V \div R = 4.5 \div 2000 = 0.00225\ A = 2.25\ mA$.
E3. El voltaje de umbral del BASIC Stamp es 1.4 V, pero el umbral de luz ocurre a 2.8 V. Entonces, el fototransistor entrega una cierta corriente que resulta en una medición de 2.8 V, regida por $V = I \times R$, o sea $2.8\ V = I \times 2000\ \Omega$. Necesitamos encontrar la resistencia que haga que el voltaje sea 1.4 V para esa misma corriente, es decir $1.4\ V = I \times R$. Para encontrar R, rearreglamos la primera ecuación para determinar I; es decir $I = 2.8\ V \div 2000\ \Omega$. Luego, sustituimos $2.8\ V \div 2000\ \Omega$ en lugar de I en la segunda ecuación y resolver R. Esto es $1.4\ V = I \times R \rightarrow 1.4\ V = (2.8\ V \div 2000\ \Omega) \times R \rightarrow R = 1.4\ V \div (2.8\ V \div 2000\ \Omega) = 2000\ \Omega \times (1.4\ V \div 2.8\ V) = 1000\ \Omega = 1\ k\Omega$.
E4. 105 → 10 con 5 ceros y multiplicados por 1 pF. $1{,}000{,}000 \times 1\ pF = (1 \times 10^{6}) \times (1 \times 10^{-12})\ F = 1 \times 10^{-6}\ F = 1\ \mu F$.
E5. **`RCTIME 7, 1, tDecay`**
E6. 1.625 × 256 ÷ 5 = 83.2, tome 83. Respuesta: **`PWM 6, 83, 1`**
E7. **`ndShade`** = 500 – (1000 × **`tLeft`** ÷ (**`tLeft`** + **`tRight`**)) = 500 – (1000 × 1001 ÷ (1001 + 1001)) = 500 – 1000/2 = 500 – 500 = 0.
E8. **`DEBUG REP "="\50`**
E9. **`DEBUG CRSRXY 8, 10, "#"`**

P1.

```
' Robotica con el Boe-Bot - CH6P1.bs2
' Suna periodicamente con luz brillante. De lo contrario, silencio.
' {$STAMP BS2}
' {$PBASIC 2.5}

PAUSE 1000
DEBUG "Program running..."

DO
  IF IN6 = 1 THEN FREQOUT 4, 20, 4000
  PAUSE 100
LOOP
```

P2. La solución de esto es hacer una copia de LightSeekingBoeBot.bs2, y agregar un comando a la Rutina Principal: **`ndShade = -ndShade`**. Agréguelo justo después

de la llamada de la subrutina **`Light_Shade_Info`**. Luego, en vez de indicar alejarse de la sombra, indicará alejarse de la luz.

P3. A continuación, una Rutina Principal modificada de LightSeekingBoeBot.bs2 que navega hacia la luz y se detiene cuando llega bajo una lámpara incandescente. La clave para esto es muy simple porque LightSeekingBoeBot.bs2 tiene una variable luz que alcanza valores mas altos bajo la luz brillante. Con cada repeticion del **`DO…LOOP`**, la declaración **`IF…THEN`** busca valores mayores a 320. Para areas de luz menores y lámparas más debiles, quizá tenga que ajustar **`IF luz > 320 THEN...`** para que compare la variable luz contra un valor menor, por ejemplo: **`IF luz > 250 THEN`**… Disminuir el valor contra el que la declaración **`IF…THEN`** compara la variable luz lo hará mas sensible; incrementarlo lo hará menos sensible. El valor 324 es el mayor valor posible valor por lo que no incremente su valor de comparación por arriba de 323.

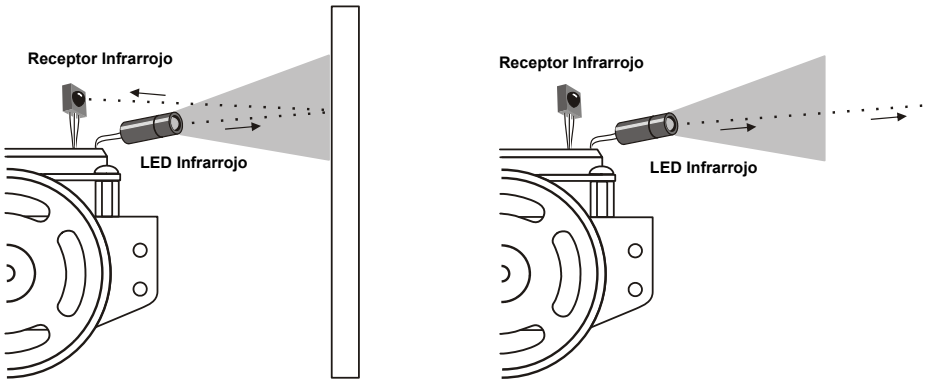
TIP: Use LightSensorValues.bs2 para probar y encontrar un valor que este entre la luz ambiental y el haz de la luz brillante.

```
DO
  GOSUB Light_Shade_Info

  IF luz > 320 THEN
    FREQOUT 4, 500, 3000
    PAUSE 500
    FREQOUT 4, 500, 3500
    PAUSE 500
    END
  ENDIF

  IF (ndShade + 500) > 500 THEN
    pulseLeft = 900 - ndShade MIN 650 MAX 850
    pulseRight = 650
  ELSE
    pulseLeft = 850
    pulseRight= 600 - ndShade MIN 650 MAX 850
  ENDIF

  PULSOUT 13, pulseLeft
  PULSOUT 12, pulseRight
LOOP
```

Capítulo 7: Navegando con Luces Frontales Infrarrojas

El Boe-Bot ya puede usar filamentos para rodear objetos que detecta cuando los golpea, pero ¿no sería mejor si pudiera "verlos" y decidir qué hacer? Es exactamente lo que puede hacer con luces frontales infrarrojas y ojos como los de la Figura 7-1. Las luces frontales infrarrojas es un LED infrarrojo dentro de un escudo de luz que dirige su luz al frente como una lámpara de mano. El ojo infrarrojo es un receptor infrarrojo que envía al BASIC Stamp señales de nivel alto/bajo para indicar si detecta la luz del LED infrarrojo reflejada por un objeto.

Figura 7-1: Detección Infrarroja de Objetos

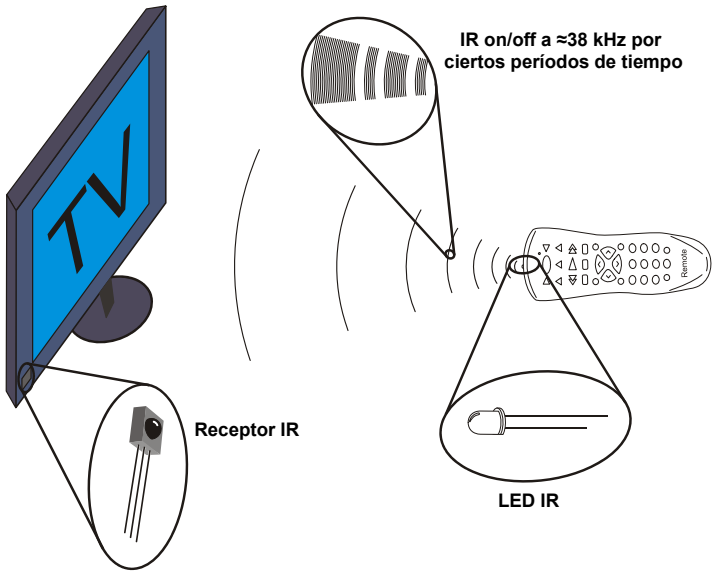


Izq: Infrarrojo reflejado, obstáculo detectado. Der: No refleja Infrarrojo, obstáculo no detectado.

LUZ INFRARROJA

Infrarrojo se abrevia IR y puede visualizarlo como una forma de luz que el ojo humano no puede detectar (revise la Figura 6-2 en la página 171). Los dispositivos como el LED IR presentado en este Capítulo emite luz infrarroja y los dispositivos como el fototransistor del Capítulo previo y el receptor infrarrojo de este Capítulo detectan luz IR. La Figura 7-2 muestra como el LED IR que el Boe-Bot usa como una pequeña lámpara es el mismo que puede encontrar en casi cualquier control remoto de TV. El control remoto de TV manda mensajes IR a su TV y el microcontrolador en su TV los recibe con un receptor IR como el que usará su Boe-Bot para detectar IR reflejado por objetos.

Figura 7-2: LED y Receptor IR en su casa



Note que el control remoto de la TV envía mensajes que describen qué botón presiona al hacer parpadear encendiendo/apagando su LED IR en las cercanías de 38 kHz (cerca de 38,000 veces por segundo). El receptor IR solo responde si está parpadeando en este nivel. Esto previene que el infrarrojo de fuentes como el sol y luces incandescentes sea malinterpetado como el remoto. Entonces, para enviar señales que el receptor IR pueda detector, su BASIC Stamp también tendrá que enviar señales en el rango de 38 kHz. El lenguaje PBASIC lo logra con poco esfuerzo, con solo una línea de código para transmitir la señal y una segunda línea para checar el receptor IR.

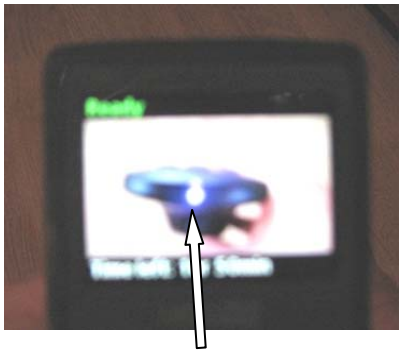
Algunas luces fluorescentes generan señales detectables por los receptores IR. Estas luces pueden causar problemas a las luces delanteras infrarrojas de su Boe-Bot. Una de las cosas que hará en este Capítulo es desarrollar un "olfateador" de interferencia IR que pueda usar para probar las luces fluorescentes cercanas a las rutas de su Boe-Bot.

Los sensores de color de luz en la mayoría de las cámaras digitales, celulares, teléfonos y cámaras web, todos pueden detectar luz infrarroja, lo que nos da una forma de "verla" aún cuando el ojo humano no lo haga. La Figura 7-3 muestra un ejemplo con una cámara

digital y un control remoto. Cuando mantiene presionado un botón en el remoto y apunta su LED IR al lente de la cámara digital, lo muestra parpadeando con luz blanca brillante.

Figura 7-3: LED IR en un Control Remoto de TV Visto con una Cámara Digital

Con un botón presionado, el LED IR no se ve diferente.



A través del display de una cámara digital, el LED IR aparece parpadeando con luz brillante blanca.

Los sensors de pixeles dentro de la cámara digital detectan niveles de luz roja, verde y el procesador suma esos nieles para determinar el color y brillo de cada pixel. Sin importer si un senxor de pixel detecta rojo, verde o azul, detecta infrarrojo. Puesto que los 3 sensores de color de piexeles detectan infrarrojo, la cámara mezcla los colores juntos, lo que resulta en blanco.

***Infra* significa *abajo*, luego *infrarrojo* significa *abajo del rojo*.** El nombre se refiere a que la frecuencia de las ondas de luz infrarroja es menor que la de las ondas de luz roja.

Longitudes de ondaIR y sus usos: La longitud de onda que transmite nuestro LED IR es 980 nm, y es la misma que detecta nuestro receptor. Esta longitud de onda esta en el rango del infrarrojo cercano, que es de 2000 a 10,000 nm, y algunas longitudes de onda en este rango se usan para los visores de visión noctura y sensado de temperatura por IR.

ACTIVIDAD #1: CONSTRUYENDO Y PROBANDO LOS DETECTORES IR DE OBJETOS

En esta Actividad, construirá y probará detectores IR de objetos para el robot Boe-Bot.

- ✓ Reúna las partes en la Lista de partes usando la Figura 7-4 para ayudarle a identificar los receptores infrarrojos, LEDs y las partes de ensamble del escudo.

Lista de partes:

(2) receptores IR
(2) LEDs IR (transparentes)
(2) Ensambles de escudos para LEDs IR
(2) Resistencias, 220 Ω (rojo-rojo-cafe)
(2) Resistencias, 1 kΩ (cafe-negro-rojo)

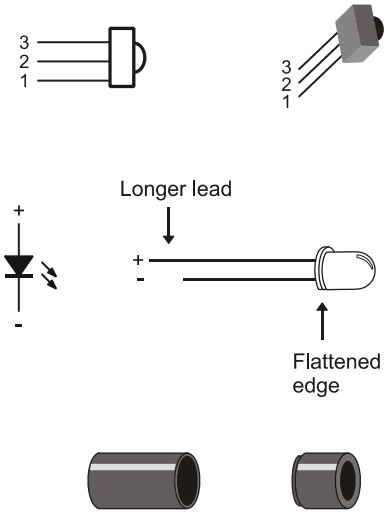


Figura 7-4
Partes Nuevas Usadas en este Capítulo

Receptor IR (arriba)

LED IR (enmedio)

Ensamble de escudo IR LED (abajo)

✓ Revise la Figura 7-5 para asegurarse que ha seleccionado los LEDs infrarrojos y no los fototransistores. El LED infrarrojo tiene un domo plastic más alto y mas redondeado y se muestra a la derecha de la Figura 7-5.

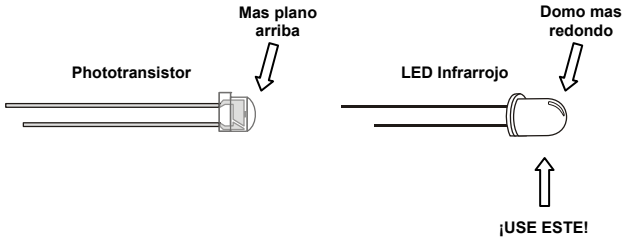


Figura 7-5
Distinguiendo los fototransistors de los LEDs Infrarrojos

Asegúrese de tener dos LEDs infrarrojos.

Construyendo las luces frontales IR

✓ Inserte el LED infrarrojo en el sujetador de LED (la pieza mas larga del ensamble de escudo) como se muestra en la Figura 7-6.
✓ Asegúrese que el LED IR se encaja en el sujetador de LED.
✓ Deslice la cubierta de LED (la pieza más pequeña del ensamble de escudo) sobre la cápsula plástica transparente del LED IR. El anillo a un extremo de la cubierta debe ajustar sobre el sujetador.
✓ Use un poco de cinta transparente para asegurar que las dos mitades del escudo no se separen al usarlo.

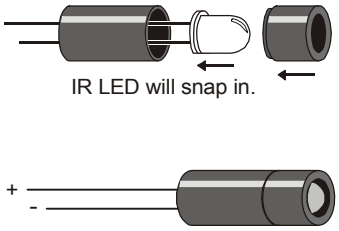


Figura 7-6
Encajando el LED IR en el ensamble de escudo

Circuito de Detección de Objetos IR

La Figura 7-7 muestra el esquemático del circuito de detección de objetos IR y la Figura 7-8 muestra un diagrama de conexiones del circuito. En el diagrama de conexiones, un detector IR de objetos (LED IR y receptor) es montado en cada esquina de la tablet lo mas próximo al frente del Boe-Bot.

- ✓ Desconecte la energía de su tarjeta y servos.
- ✓ Construya el circuito del esquemático de la Figura 7-7 usando el diagrama de conexiones de la Figura 7-8 como referencia para colocar las partes.

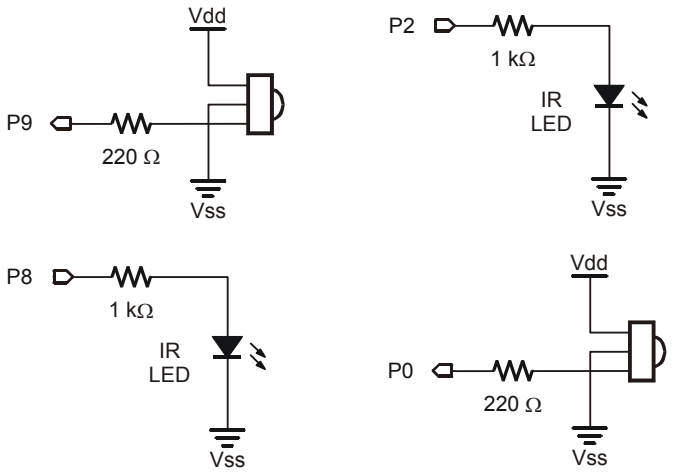


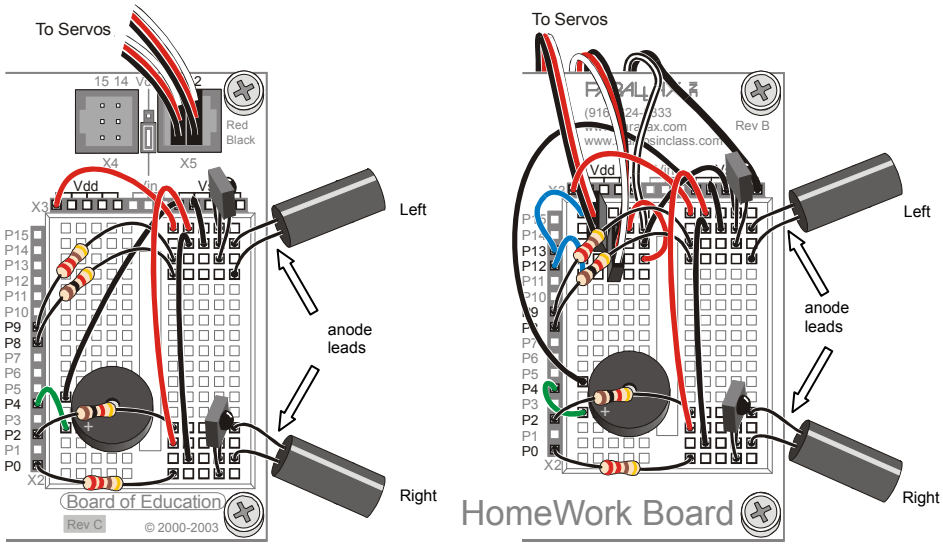
Figura 7-7
Detectores IR de objetos izquierdo y derecho

¡Ponga atención a los ánodos y cátodos del LED IR!

La terminal ánodo es la más larga en un LED IR por convención. La terminal cátodo es más corta y está montada más cerca a la cara plana del encapsulado plástico. Las mismas convenciones aplicaron para los LEDs rojos en el Capítulo 2.

En la Figura 7-8, la terminal ánodo de cada LED IR se conecta a una Resistencia de 1 kΩ. La terminal cátodo se conecta en la misma columna de la tableta que el pin central del detector IR, y esa columan se conecta a Vss con cable conector.

Figura 7-8: Diagramas de conexiones para circuitos infrarrojos Emisor y Receptor



Código de Prueba de Detección de Objetos

Los receptores infrarrojos de su Boe-Bot están diseñados para detector luz infrarroja con una longitude de onda de 980 nm que parpadea encendido/apagado o que varía en su brillantes a una tasa de alrededor de 38 kHz. El LED infrarrojo emite IR de 980 nm, eso entonces está resuelto. Todo lo que necesitamos es hacer que la brillantes del LED varíe, más brillante y luego atenúe, a una tasa de aproximadamente 38 kHz. Podemos hacer esto con el mismo comando que hemos venido usando para hacer que el parlante del Boe-Bot's toque un tono al inicio de cada programa—el comando **FREQOUT**.

Solo toma 2 líneas de código para probar la presencia o ausencia de un objeto usando un circuto de detección IR de objetos. He aquí un ejemplo para averiguar si un objeto está en frente del circuito de detección IR izquierdo del robot Boe-Bot.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9
```

El comando **`FREQOUT 8, 1, 38500`** hace que varíe la brillantez del LED IR, haciéndose más brillante y atenuándose 38500 veces por segundo. Lo hace durante 1 ms; luego, **`irDetectLeft = IN9`** guarda la salida de los receptores IR en una variable. La salida del detector sera alta si no detecta IR de 38.5 kHz reflejado por un objeto, o bajo si lo detecta. Entonces el valor de **`IN9`** que se copia en la variable **`irDetectLeft`** sera 1 si no se detecta objeto, 0 si se detecta.

Siempre use `irDetectLeft = IN9` justo después de `FREQOUT 8, 1, 38500`.

El BASIC Stamp tiene una ventana de tiempo breve para copiar la señal binaria que obtiene del receptor IR a una variable. El receptor IR manda una señal baja mientras que detecta IR de 38.5 kHz reflejado por un objeto, lo que causa que **`IN9`** guarde un 0. Cuando el BASIC Stamp termina de transmitir su señal **`FREQOUT`** y continua con el siguiente comando, deja de enviar esa señal de 38.5 kHz. Entonces el programa tiene que usar **`irDetectLeft = IN9`** para capturer ese valor 0 antes de que el receptor se percate de que la señal de 38.5 kHz se ha detenido. Solo toma una pequeña fracción de milisegundo al receptor IR para percatarse de ello y después de ello, su salida regresa a alto, y **`IN9`** guarda un 1 otra vez.

Programa Ejemplo: TestLeftIr.bs2

- ✓ Reconecte la energía a su tarjeta.
- ✓ Introduzca, salve y corraTestLeftIr.bs2.

```
' Robotica con el Boe-Bot - TestLeftIr.bs2
' Prueba circuitos de deteccion IR de objetos, LED IR en P8 y detector en P9.

' {$STAMP BS2}
' {$PBASIC 2.5}

irDetectLeft   VAR     Bit

DO
  FREQOUT 8, 1, 38500
  irDetectLeft = IN9

  DEBUG HOME, "irDetectLeft = ", BIN1 irDetectLeft
  PAUSE 100
LOOP
```

- ✓ Deje el Boe-Bot conectado a su cale de programación, ya que estará usando la Terminal de Depuración para probar su detector IR de objetos.
- ✓ Coloque un objeto, como su mano o una hoja de papel, a aproximadamente 1 pulgada del detector IR de objetos (vea la Figura 7-1 en la página 221).
- ✓ Verifique que la Terminal de Depuración despliegue un 0 cuando coloque un objeto a pocas pulgadas en frente del detector IR de objetos.
- ✓ Verifique que despliega un 1 cuando remueve el objeto.
- ✓ Si la Terminal de Depuración despliega los valores esperados para objeto no detectado (1) y objeto detectado (0), continúe con la sección Su Turno.
- ✓ Si la Terminal de Depuración no muestra los valores esperados, intente los pasos en la sección Corrección de Problemas.

Corrección de Problemas

Si la Terminal de Depuración no despliega los valores esperados, ntente esta lista:

- ✓ Revise el circuito y errores de tecleado de programa. Un error común es usar una Resistencia de 10 kΩ (café-nego-naranja) en vez de una de 1 kΩ (café-negro-rojo).
- ✓ Mantenga el Boe-Bot lejos de la luz directa del sol.
- ✓ Si siempre obtiene un 0, aún cuando un objeto no está colocado en frente del Boe-Bot, puede haber un objeto cercaqno que esté reflejando el infrarrojo. La superficie de la mesa en frente del Boe-Bot es un culpable común. Mueva el Boe-Bot para que el LED IR y el detector no puedan reflejar ningún objeto cercano.
- ✓ Si la lectura es 1 la mayoría del tiempo cuando no hay objetos en frente del Boe-Bot, pero parpadea a 0 ocasionalmente, quiere decir que tiene interferencia de una lámpara fluorescente cercana. Apague todas las lámparas fluorescents cercanas y repita sus pruebas. También pruebe cerrar las persianas si está cerca de una ventana.
- ✓ Si la lectura es 1 todo el tiempo, aún cuando un objeto es colocado en frente del Boe-Bot: aún cuando no es un error común, los fabricantes ocasionalmente hacen un lote de LEDs con las terminales corta y larga al revés. Si ya volvió a revisar su cableado y su programa, intente desconectar el LED IR e inverter su polaridad conectando la pata corta a la Resistencia de 1 kΩ y la pata larga conectada a Vss.
- ✓ Una última prueba que puede intentar es conectar su circuito LED IR a un pin I/O diferente y ajustar your programa. Empiece con una correcta orientación de ánodo/cátodo y si no trabaja, pruebe volteándolo otra vez.

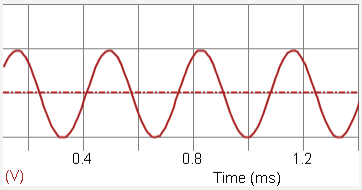
Su Turno – Pruebe el Detector IR de Objetos Derecho

- ✓ Salve TestLeftIr.bs2 como TestRightIr.bs2.
- ✓ Cambie el comando **`DEBUG`**, el título del programa y los comentarios para referirse al detector IR de objetos derecho.
- ✓ Cambie el nombre de la variable de **`irDetectLeft`** a **`irDetectRight`**. Necesitará hacerlo en 4 puntos del programa.
- ✓ Cambie el argumento ***Pin*** del comando **`FREQOUT`** de 8 a 2.
- ✓ Cambie el registro de entrada monitoreado por la variable **`irDetectRight`** de **`IN9`** a **`IN0`**.
- ✓ Repita los pasos de prueba en esta Actividad para el detector IR derecho.

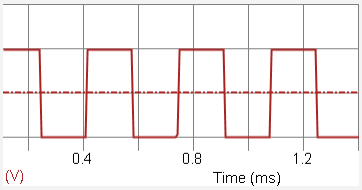
Ondas senoidales sintetizadas por FREQOUT

El comando **`FREQOUT`** transmite una secuencia rápida de señales on/off que digitalmente sintetizan voltajes para crear un patrón de onda senoidal, las cuales suenan más naturales que las ondas cuadradas cuando son ejecutadas por un parlante. Las ondas cuadradas hacen un ruido más como un zumbido.

Onda Senoidal



Onda Cuadrada



Una señal **`FREQOUT`** contiene dos componentes de onda senoida con dos frecuencias diferentes. Una componente de la frecuencia es **`Freq1`**. La segunda componente de la frecuencia es 65536 – **`Freq1`**.

Cuando el comando **`FREQOUT`** se usa para tocar tonos audibles, la segunda recuencia de la señal esta siempre muy por arriba de los 20 kHz, que es típicamente la más alta que el oído humano puede detectar.

Ejemplo 1: **`FREQOUT 4, 2000, 3000`** toca un tono de onda senoidal de 3 kHz en el piezoparlante porque **`Freq1`** es 3000. La señal contiene una segundo componente con una frecuencia de 65536 – 3000 = 62536 Hz, pero el oído humano no puede detectarla. Puesto que 65536 – 62536 = 3000, podría tocar el mismo tono con **`FREQOUT 4, 2000, 62536`**. Aún cuando **`Freq1`** está ahora muy afuera del rango del oído humano, la segunda señal es 3 kHz, entonces obtendrá el mismo tono en su piezoparlante.

Ejemplo 2: **`FREQOUT 8, 1, 38500`** hace que la brillantez del LED varíe a una tasa de 38500 Hz de tal forma que el receptor IR puede detectarla. La señal creada también contiene una segunda onda senoidal con una frecuencia de 65536 – 38500 = 27036 Hz, pero esa señal no tiene efecto en el receptor IR.

ACTIVIDAD #2: PRUEBA DE CAMPO DE DETECCIÓN DE OBJETOS E INTERFERENCIA INFARROJA

En esta Actividad, construirá y probará LEDs indicadores que le dirán si un objeto es detectado sin la ayuda de la Terminal de Depuración. Esto es útil si no está cercano a una PC o laptop y necesita corregir problemas en sus circuitos detectores de IR. También escribirá un programa para "olfatear" interferencia infrarroja de las lámparas fluorescentes. Algunas lámparas fluorescentes Mandan señales que semejan la señal enviada por sus LEDs infrarrojo. El dispositivo dentro de una lámpara fluorescente que controla su voltaje es llamado balastro. Algunos balastros operan en la mismo rango de frecuencia que su detector IR, 38.5 kHz, que a su vez causa que la lámpara emita una señal a esta frecuencia. Cuando integra detección IR de objetos con navegación, ¡esta interferencia puede causar algún comportamiento bizarro del Boe-Bot!

Reconstruyendo los Circuitos Indicadores LED

Estos son los mismos circuitos indicadores LED que usó con los filamentos.

Lista de partes:

(2) LEDs Rojos
(2) Resistencias, 220 Ω (rojo-rojo-café)

- ✓ Desconecte la energía de su tarjeta y servos.
- ✓ Construya el circuito mostrado en la Figura 7-9 usando la Figura 7-10 como referencia.

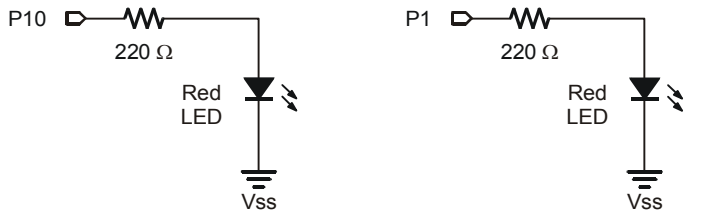
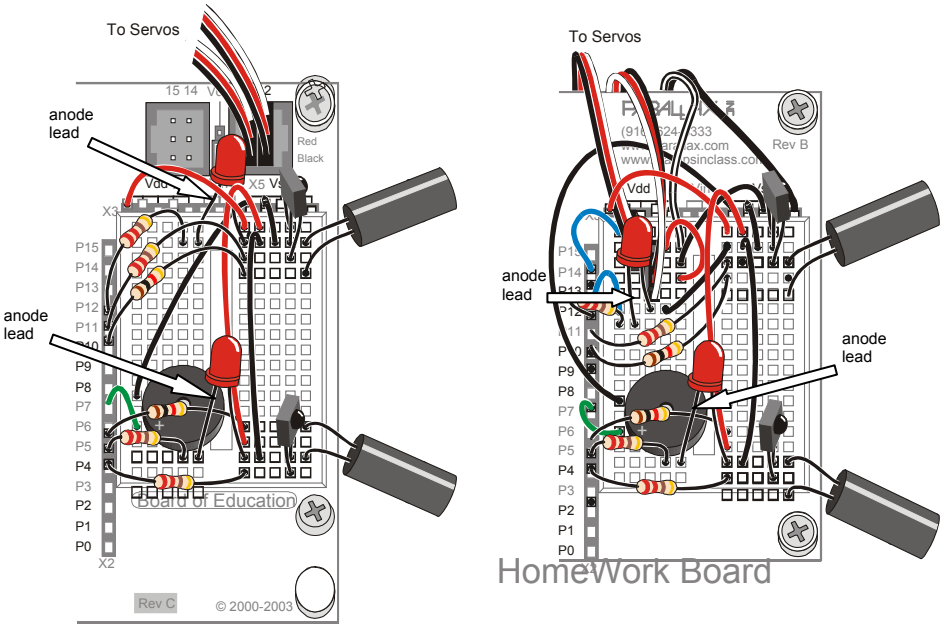


Figura 7-9
LEDs Indicadores izquierdo y derecho

Figura 7-10: Diagramas de conexiones para circuitos emisor y receptor Infrarrojos



Board of Education (izq) y HomeWork Board (der)

Probando el Sistema

Hay muy pocos componentes involucrados en este sistema, y esto incrementa la posibilidad de un error de conexiones. Por eso es importante tener un programa de prueba que le indique lo que los detectores infrarrojos están sensando. Puede usar este programa para verificar que todos los circuitos están trabajando antes de desconectar el Boe-Bot de su cable de programación y probar otros objetos.

Programa Ejemplo – TestBothIrAndIndicators.bs2

- ✓ Reconecte la energía a su tarjeta.
- ✓ Introduzca, salve y corraTestBothIrAndIndicators.bs2.
- ✓ Verifique que el parlante hace un tono claro y audible mientras que la Terminal de Depuración despliega “Testing piezospeaker…”
- ✓ Use la Terminal de Depuración para verificar que el BASIC Stamp aún recibe un cero de cada detector IR cuando un objeto es colocado en frente a el.

- ✓ Verifique que el LED cercano a cada detector emite luz cuando el detector detecta un objeto. Si uno o ambos LEDs parecen no trabajar, revise su cableado y su programa.

```
' Robotica con el Boe-Bot - TestBothIrAndIndicators.bs2
' Prueba circuitos IR de deteccion de objetos.

' {$STAMP BS2}                              ‘ Directiva Stamp.
' {$PBASIC 2.5}                             ‘ Directiva PBASIC.

' -----[ Variables ]------------------------------------------------------

irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit

' -----[ Inicializacion ]--------------------------------------------------

DEBUG "Testing piezospeaker..."
FREQOUT 4, 2000, 3000

DEBUG CLS,
      "IR DETECTORS", CR,
      "Left   Right", CR,
      "-----  -----"

' -----[ Rutina Principal ]------------------------------------------------

DO
  FREQOUT 8, 1, 38500
  irDetectLeft = IN9

  FREQOUT 2, 1, 38500
  irDetectRight = IN0

  IF (irDetectLeft = 0) THEN
    HIGH 10
  ELSE
    LOW 10
  ENDIF

  IF (irDetectRight = 0) THEN
    HIGH 1
  ELSE
    LOW 1
  ENDIF

  DEBUG CRSRXY, 2, 3, BIN1 irDetectLeft,
        CRSRXY, 9, 3, BIN1 irDetectRight

  PAUSE 100
LOOP
```

Su Turno – Prueba Remota y Prueba de Rango

Ahora puede usar sus detectores LED para tomar su Boe-Bot y probar sus detectores IR con objetos que pudieran no estar de otra forma al alcance del cable de programación de su computadora.

- ✓ Desconecte su Boe-Bot del cable de programación, y lleve su Boe-Bot frente a una variedad de objetos y pruebe el rango de los detectores IR.
- ✓ Pruebe el rango de detección de objetos de diferentes colores. ¿Qué color es detectado a mayor rango? ¿Qué color es detectado con el rango menor?

Olfateando Interferencia IR

Si ha notado que su Boe-Bot le hace saber que detectó algo aún cuando nada estaba en el rango, puede significar que una luz cercana está generando alguna luz IR a una frecuencia cercana a 38.5 kHz. Si intenta tener un concurso o demostración del Boe-Bot bajo una de estas luces, sus sistemas infrarrojos podrían tener una ejecución muy pobre. Lo ultimo que cualquiera quiere es que su robot no se desempeñe como se anunció en una demostración pública, asi es que asegúrese revisar cualquier prospecto de area de demostración con este programa "olfateador" de interferencia IR con antelación.

El concepto detrás de este programa es simple: no transmita ningún IR a través de los LEDs, solo monitoreé para ver si se detecta algún IR. Si es así suena la alarma usando el piezoparlante.

Puede usar un control remoto para generar interferencia IR sobre casi cualquier equipo. TVs, VCRs, CD/DVD y proyectores, todos usan el mismo tipo de detectores IR que tiene en su Boe-Bot ahora. Igualmente, los remotes que usa para controlar estos dispositivos todos usan la misma clase de LED IR que está en su Boe-Bot para transmitir mensajes al detector IR en su TV, VCR, CD/DVD, etc.

Programa Ejemplo – IrInterferenceSniffer.bs2

- ✓ Introduzca, salve y corra IrInterferenceSniffer.bs2.
- ✓ Pruebe para asegurarse que el Boe-Bot suena la alarma cuando detecta Interferencia IR. Si está en un salon de clases, puede hacerlo con otro Boe-Bot que esté corriendo TestBothIrAndIndicators.bs2. Si no tiene un segundo Boe-Bot, use un control remote de una TV, VCR, CD/DVD o proyector. Solo apunte el remote al Boe-Bot y presione un botón. Si el Boe-Bot responde sonando la alarma, sabrá que su olfateador de Interferencia IR está trabajando.

```
' Robotica con el Boe-Bot – IrInterferenceSniffer.bs2
' Pruebe luces fluorescentes, remotos infrarrojos y otras fuentes
' de 38.5 kHz buscando Interferencia IR.

' {$STAMP BS2}                              ‘ Directiva Stamp.
' {$PBASIC 2.5}                             ‘ Directiva PBASIC.

counter        VAR     Nib

DEBUG "IR Interferencia not detected, yet...", CR

DO

  IF (IN0 = 0) OR (IN9 = 0) THEN
    DEBUG "IR Interferencia detected!!!", CR
    FOR counter = 1 TO 5
      HIGH 1
      HIGH 10
      FREQOUT 4, 50, 4000
      LOW 1
      LOW 10
      PAUSE 20
    NEXT
  ENDIF

LOOP
```

Su Turno – Probando luces Fluorescentes que Interfieren

- ✓ Desconecte su Boe-Bot de su cable de programación y apúntelo a cualquier luz fluorescente cercana a donde planea operarlo. Especialmente si obtiene alarmas frecuentes, apague esa luz fluorescente antes de intentar usar la detección IR de objetos bajo esa luz.

Siempre use este IrInterferenceSniffer.bs2 para asegurarse que cualquier area donde esté usando el Boe-Bot esta libre de interferencia infrarroja.

ACTIVIDAD #3: AJUSTES DE RANGO DE DETECCIÓN INFRARROJA

Quiza haya notado que se pueden usar luces frontales más brillantes (o una lámpara más brillante) en autos para ver objetos más lejanos cuando está oscuro. Al hacer el LED infrarrojo del Boe-Bot más brillante, también puede incrementar su rango de detección. Al reducir la oposición a la corriente eléctrica, una resistencia menor permite mayor corriente fluyendo a través de un LED. Más corriente a través de un LED es lo que causa que brille más. En esta Actividad, examinará el efecto de diferentes valores de resistencia con ambos LEDs, rojo e infrarrojo.

Lista de partes:

Necesitará algunas partes extra para esta Actividad.

(2) Resistencias, 470 Ω (amarillo-violeta-café)
(2) Resistencias, 220 Ω (rojo-rojo-café)
(2) Resistencias, 2 kΩ (rojo-negro-rojo)
(2) Resistencia, 4.7 kΩ (amarillo-violeta-rojo)

Resistencia en Serie y Brillo de LED

Primero, usemos uno de los LEDs rojos para "ver" la diferencia que una resistencia hace en la brillantez de un LED. Todo lo que necesitamos para probar el LED es un programa que envíe una señal alta al LED.

Programa Ejemplo – P1LedHigh.bs2

- ✓ Introduzca, salve y corra P1LedHigh.bs2.
- ✓ Corra el programa y verifique que el LED en el circuito conectado a P1 emite luz.

```
' Robotica con el Boe-Bot - P1LedHigh.bs2
' Configura P1 en alto para probar brillantez de LED, probando con cada uno de
' estos valores de resistencia en turno: 220 ohm , 470 ohm, 1 k ohm.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

HIGH 1

STOP
```

El comando **STOP** se usa aquí en vez de **END**, puesto que **END** pondría al BASIC Stamp en modo de baja energía.

Su Turno – Probando brillantez de LED

Recuerde desconectar la energía antes de hacer cambios en el circuito. Recuerde también que el mismo programa correrá de nuevo cuando reconecte la energía, asi es que podrá retomarlo donde lo dejó con cada prueba.

- ✓ Note cuán brillante relumbra el circuito P1 LED con la resistencia de 220 Ω.

- ✓ Remplace la resistencia de 220 Ω conectada a P1 y el cátodo derecho del LED con una resistencia de 470 Ω. Note ahora que tan brillante es el LED.
- ✓ Repita para una Resistencia de 2 kΩ.
- ✓ Repita una vez mas con una resistencia de 4.7 kΩ.
- ✓ Remplace la Resistencia de 4.7 kΩ con la resistencia de 220 Ω antes de continuar con la siguiente parte de esta Actividad.
- ✓ Explique en sus propias palabras la relación entre brillantez del LED y la resistencia en serie.

Resistencia en Series y Rango de detección IR

Ahora sabemos que una Resistencia serie menor hará que un LED reluzca más brillantemente. Una hipótesis razonable sería que un LED IR más brillante puede hacer posible detectar objetos mas lejanos.

- ✓ Abra y corraTestBothIrAndIndicators.bs2 (página 231).
- ✓ Verifique que ambos detectores trabajan adecuadamente.

Su Turno – Probando el Rango del LED IR

- ✓ Con una regla, mida la distancia más alejada del LED IR a la que una hoja de papel puede ser vista, usando una resistencia de 1 kΩ, y registre su dato en la Tabla 7-1.
- ✓ Remplace las resistencias de 1 kΩ que conectan P2 y P8 a los ánodos de los LED IR con resistencias de 4.7 kΩ.
- ✓ Determine la distancia más lejana a la que la misma hoja de papel es detectada y registre su dato.
- ✓ Repita con Resistencias de 2 kΩ, 470 Ω y 220 Ω.

Tabla 7-1: Distancias de detección vs. Resistencia

IRELD Resistencia Series (Ω)	Máxima Distancia de detección
4700	
2000	
1000	
470	
220	

- ✓ Antes de continuar con la siguiente Actividad, regrese sus detectores IR de objetos a sus configuraciones originales (con resistencias de 1 kΩ en serie con cada LED IR).

- ✓ También, antes de seguir, asegúrese probar este último cambio con TestBothIrAndIndicators.bs2 para verificar que los dos detectores IR de objetos trabajan adecuadamente.

ACTIVIDAD #4: DETECCIÓN DE OBJETOS Y RODEO

Una cosa interesante acerca de los detectores IR es que sus salidas son tal y como las de los filamentos. Cuando no se detecta algún objeto, la salida es alta; cuando un objeto es detectado, la salida es baja. En esta Actividad, se modifica RoamingWithWhiskers.bs2 de la página 156 para que trabaje con los detectores IR.

Convirtiendo el programa de Filamentos para Detection/Rodeo IR de Objetos

El siguiente programa ejemplo empezó como RoamingWithWhiskers.bs2. Además de ajustar el nombre y la descripción, se agregaron dos variables bit para guardar los estados de los detectores IR.


```
irDetectLeft  VAR     Bit
irDetectRight VAR     Bit
```

También se agregó una rutina para leer los detectores IR de objeto.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9

FREQOUT 2, 1, 38500
irDetectRight = IN0
```

Las declaraciones **IF…THEN** fuerno modificadas para que vean las variables que guardan las detecciones IR de objetos en vez de las entradas de los filamentos.

```
IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
  GOSUB Turn_Left
ELSEIF (irDetectLeft = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Right
ELSEIF (irDetectRight = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
ELSE
  GOSUB Forward_Pulse
ENDIF
```

Programa Ejemplo – RoamingWithIr.bs2

- ✓ Abra RoamingWithWhiskers.bs2
- ✓ Modifíquelo para que iguale el siguiente programa.
- ✓ Reconecte la energía a su tarjeta y a sus servos.
- ✓ Sálvelo y córralo.
- ✓ Verifique que, además del hecho de que no se requieren contactos, se comporta como RoamingWithWhiskers.bs2.

```
' -----[ Title ]-----------------------------------------------------------
' Robotica con el Boe-Bot - RoamingWithIr.bs2
' Adaptacion de RoamingWithWhiskers.bs2 para usar con detectores IR de objetos
' {$STAMP BS2}                      ‘ Directiva Stamp.
' {$PBASIC 2.5}                     ‘ Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Variables ]-------------------------------------------------------

  irDetectLeft  VAR     Bit
  irDetectRight VAR     Bit
  pulseCount    VAR     Byte

' -----[ Inicializacion ]---------------------------------------------------

FREQOUT 4, 2000, 3000               ' Señal de programa en inicio/reset.

' -----[ Rutina Principal ]-------------------------------------------------

DO
  FREQOUT 8, 1, 38500                   ' Guarda valores deteccion IR en
  irDetectLeft = IN9                    ' variables bit.

  FREQOUT 2, 1, 38500
  irDetectRight = IN0

  IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
    GOSUB Back_Up                       ' Ambos detectan obstaculo
    GOSUB Turn_Left                     ' Retrocede y vuelta en U(2 izqs.)
    GOSUB Turn_Left
  ELSEIF (irDetectLeft = 0) THEN        ' El izquierdo detecta
    GOSUB Back_Up                       ' Retrocede y vuelta a la derecha
    GOSUB Turn_Right
  ELSEIF (irDetectRight = 0) THEN       ' El derecho detecta
    GOSUB Back_Up                       ' Retrocede y vuelta a la izquierda
    GOSUB Turn_Left
  ELSE                                  ' Sin deteccion
    GOSUB Forward_Pulse                 ' Aplica un pulso al frente
  ENDIF                                 ' y checa otra vez
LOOP
```

```
' -----[ subrutinas ]-----------------------------------------------------

Forward_Pulse:                          ' Envia un solo pulso al frente.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                              ' Vuelta a la izq, aprox 90-grados.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20              ' Vuelta a la der, aprox 90-grados.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN

Back_Up:                                ' Retrocede.
  FOR pulseCount = 0 TO 40
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```


Su Turno

- ✓ Modifique RoamingWithIr.bs2 para que los detectores IR de objetos sean checados en una subrutina.

ACTIVIDAD #5: NAVEGACIÓN IR DE ALTO RENDIMIENTO

El estilo de maniobras pre-programadas que fueron usadas en la Actividad previa estaba bien para los filamentos, pero es innecesariamente lento al usar los LEDs y detectores IR. Puede mejorar importantemente el desempeño de navegación del Boe-Bot buscando obstáculos antes de entregar cada grupo de pulsos a los servos. El programa puede usar las entradas de los sensors para seleccionar la major maniobra para cada momento de navegación. De esta forma, el Boe-Bot nunca da más vuelta que la que necesita y puede puntualmente encontrar su camino rodeando obstáculos y navegar exitosamente rutas más complejas.

Muestreo entre cada Pulso para evitar Colisiones

Lo mejor acerca detectar un obstáculo antes de chocar con el es que le da al Boe-Bot algún espacio para navegar rodeándolo. El Boe-Bot puede aplicar un pulso para alejarse de un objeto, volver a revisar y si el objeto sigue allí, aplicar otro pulso para evitarlo. El Boe-Bot puede seguir aplicando pulsos y revisando hasta que complete el rodeo del obstáculo. Luego puede continuar con pulsos al frente. Después de experimentar con el siguiente programa ejemplo, estará de acuerdo que es una mucho mejor manera de navegar para el Boe-Bot.

Programa Ejemplo – FastIrRoaming.bs2

- ✓ Introduzca, salve y corra FastIrRoaming.bs2.

```
' Robotica con el Boe-Bot - FastIrRoaming.bs2
' El major rendimiento de la deteccion IR de objeto asiste la navegacion

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

irDetectLeft   VAR     Bit                   ' Declaracion de Variables
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word

FREQOUT 4, 2000, 3000                     ' Señal de programa en inicio/reset.

DO                                           ' Rutina Principal

  FREQOUT 8, 1, 38500                        ' Checa Detectores IR
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                             ' Decide como navegar.
  IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
    pulseLeft = 650
    pulseRight = 850
  ELSEIF (irDetectLeft = 0) THEN
    pulseLeft = 850
    pulseRight = 850
  ELSEIF (irDetectRight = 0) THEN
    pulseLeft = 650
    pulseRight = 650
  ELSE
    pulseLeft = 850
    pulseRight = 650
  ENDIF
```

```
  PULSOUT 13,pulseLeft                          ' Aplica el pulso.
  PULSOUT 12,pulseRight
  PAUSE 15

LOOP                                            ' Repite Rutina Principal
```

How FastIrRoaming.bs2 Works

This programa toma un camino ligeramente diferente para aplicar pulsos. Además de los dos bits usados para guardar las salidas del detector IR, usa dos variables word para establecer las duraciones de pulsos entregadas por el comando **PULSOUT**.

```
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word
```


Dentro del **DO…LOOP**, los comandos **FREQOUT** son usados para enviar una señal IR de 38.5 kHz a cada LED IR. Inmediatamente después de haberse enviado la señal IR de 1 ms, una variable bit guarda el estado de la salida IR. Esto es necesario, porque si espera un tiempo mayor al de un commando, el detector IR regresará a no detectado (estado de 1), sin importar si detectó un objeto.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9
FREQOUT 2, 1, 38500
irDetectRight = IN0
```

En las declaraciones **IF…THEN**, en vez de entregar pulsos o llamar rutinas de navegación, este programa establece valores de variable que serán usados en argumentos ***Duration*** de comandos **PULSOUT**.

```
IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
  pulseLeft = 650
  pulseRight = 850
ELSEIF (irDetectLeft = 0) THEN
  pulseLeft = 850
  pulseRight = 850
ELSEIF (irDetectRight = 0) THEN
  pulseLeft = 650
  pulseRight = 650
ELSE
  pulseLeft = 850
  pulseRight = 650
ENDIF
```

Antes de repetirse **DO…LOOP**, la última cosa por hacer es entregar pulsos a los servos. Note que el comando **PAUSE** ya no es 20. En vez de ello, es 15 puesto que alrededor de 5 ms se toman para checar los LEDs IR.

```
PULSOUT 13,pulseLeft                    ' Aplica el pulso.
PULSOUT 12,pulseRight
PAUSE 15
```

Su Turno

- ✓ Salve FastIrRoaming.bs2 como FastIrRoamingYourTurn.bs2.
- ✓ Use los LEDs para indicar que el Boe-Bot ha detectado un objeto.
- ✓ Intente modificar los valores a los que pulseLeft y pulseRight están establecidos para que el Boe-Bot haga todo a media velocidad.

ACTIVIDAD #6: EL DETECTOR DE CAÍDA

Hasta ahora, el Boe-Bot ha sido principalmente programado para tomar maniobras evasivas cuando un objeto es detectado. Hay también aplicaciones donde el Boe-Bot debe tomar acción evasiva cuando un objeto no es detectado. Por ejemplo, si el Boe-Bot esta navegando en una mesa, sus detectores IR podrían ver hacia la superficie como se muestra en la Figura 7-11. El programa debiera hacerle continuar al frente mientras que ambos detectores IR puedan "ver" la superficie de la mesa.

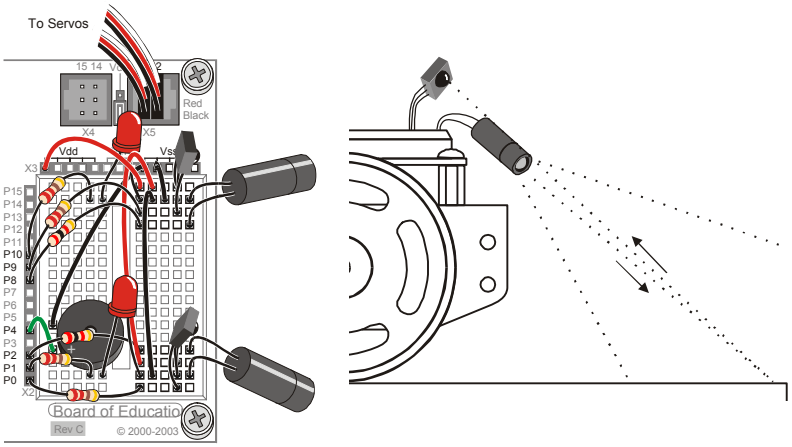


Figura 7-11
Detectores IR de objetos dirigidos abajo para buscar una caída

Vista superior (izquierda); vista lateral (derecha).

- ✓ Desconecte la energía de su tarjeta y servos.
- ✓ Apunte sus detectores IR de objetos hacia abajo y afuera como se muestra en la Figura 7-11.

Materiales Recomendados:

(1) Rollo de cinta eléctrica de aislar plástica negra, ¾″ (19 mm) de ancho.
(1) Hoja de cartulina 22 x 28 pulgadas (56 x 71 cm).

Simulando una Caída con la cinta Eléctrica

Una hoja de cartulina blanca con una orilla hecha de cinta de aislar eléctrica crea una forma manejable de simular la caída de la orilla de una mesa, con mucho menor riesgo para su Boe-Bot.

- ✓ Construya una ruta similar a la mostrada y delimitada con cinta eléctrica en la Figura 7-12. Use al menos 3 tiras de cinta eléctrica, de punta a punta sin papel visible entre ellas.
- ✓ Remplace sus resistencias de 1 kΩ con resistencias de 2 kΩ (rojo-negro-rojo) para conectar P2 a su LED IR y P8 a su LED IR. Queremos que el Boe-Bot sea observado de cerca para esta Actividad.
- ✓ Reconecte la energía a su tarjeta.
- ✓ Corra el programa IrInterferenceSniffer.bs2 (página 233) para asegurarse que las luces fluorescentes cercanas no interferirán con los detectores IR del Boe-Bot.
- ✓ Use TestBothIrAndIndicators.bs2 (página 232) para asegurarse que el Boe-Bot detecta la cartulina pero no detecta la cinta eléctrica.

Si el Boe-Bot aún "ve" la cinta eléctrica claramente, pruebe estos remedios:

- ✓ Intente ajustar los detectores IR y LEDs hacia abajo en varios ángulos.
- ✓ Intente una marca diferente de cinta eléctrica.
- ✓ Intente remplazar las resistencias de 2 kΩ con resistencias de 4.7 kΩ (amarillo-violeta-rojo) para hacer el Boe-Bot mas corto en alcance.
- ✓ Ajuste el comando **FREQOUT** con diferentes argumentos ***Freq1***. Algunos argumentos que cortarán en alcance del Boe-Bot son: 38250, 39500, 40500

Si está usando LEDs IR viejos, el Boe-Bot podría estar teniendo problemas con ser **muy corto en alcance.** Algunos remedios que incrementaran la sensibilidad del Boe-Bot a objetos y le darán mas profundidad de visión son:

- ✓ Intente resistencias de 1 kΩ (café-negro-rojo), 470 Ω (amarillo-violeta-café) o incluso de 220 Ω (rojo-rojo-café) en serie con los LEDs IR en vez de 2 kΩ.

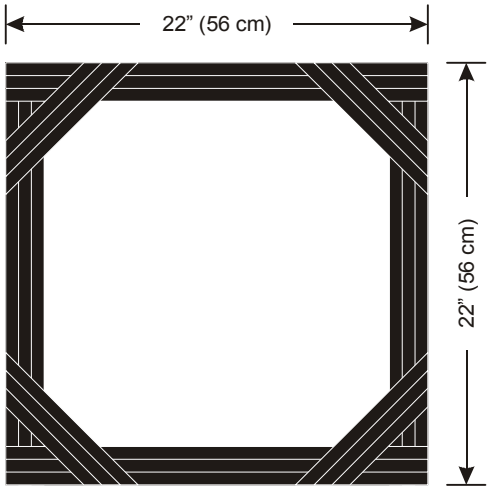


Figura 7-12
Orilla de cinta Eléctrica Simulando la orilla de la mesa

Si intenta una mesa después de tener éxito con la ruta de cinta eléctrica:

- ✓ ¡Recuerde seguir los mismos pasos que siguió antes de correr el Boe-Bot en la ruta de cinta eléctrica!

Asegúrese de ser el guardian de su Boe-Bot. Esté listo al navegar su Boe-Bot por la mesa:

- ✓ Siempre esté listo a recoger su Boe-Bot de arriba al acercarce a la orilla de la mesa que navegue. Si el Boe-Bot intent continuar fuera de la orilla, recógalo antes de que caiga. De lo contrario, ¡su Boe-Bot podría ser un No-Bot!
- ✓ Su Boe-Bot pudiera detectar si usted está en su linea de visión. Su programa actual no tiene forma de diferenciarlo de la mesa abajo y quizá intente continuar al frente y fuera de la orilla de la mesa. Entonces, manténgase fuera de la línea de vision de su detector.

Programando para Detección de condiciones de caídas

Por la mayor parte, programar su Boe-Bot para navegar alrededor de una mesa sin caer por la orilla es un asunto de ajustar las declaraciones **`IF...THEN`** de FastIrNavigation.bs2. El principal ajuste es que los servos deben ser dirigidos para hacer que el Boe-Bot vaya al frente cuando **`irDetectLeft`** y **`irDetectRight`** som ambos 0, indicando que un objeto (la superficie de la mesa) ha sido detectado. El Boe-Bot también tiene que alejarse de un detector que indica que no ha detectado un objeto. Por ejemplo, si **`irDetectLeft`** es 1, el Boe-Bot mejor debiera girar a la derecha.

Una segunda característica de un programa para alejarse de condiciones de caídas es una distancia ajustable. Quizá quiera que su Boe-Bot tome solo un pulso al frente entre

chequeo de los detectores, pero tan pronto como una condición de caída sea detectada, quizá quiera que su Boe-Bot tome varios pulsos para girar antes de checar los detectores de nuevo.

Solo porque esté haciendo pulsos multiples en una maniobra evasiva, no significa que tenga que regresar al estilo de navegación con filamentos. En vez de ello, puede agregar una variable **`pulseCount`** que puede usar para establecer el número de pulsos a entregar para una maniobra. El comando **`PULSOUT`** puede ser colocado dentro de un ciclo **`FOR…NEXT`** que ejecute **`FOR 1 TO pulseCount`** pulsos. Para 1 pulso al frente, **`pulseCount`** puede ser 1, para diez pulsos a la izquierda, **`pulseCount`** puede ser fijado en 10, etc.

Programa Ejemplo – AvoidTableEdge.bs2

- ✓ Abra FastIrNavigation.bs2 y sálvelo como AvoidTableEdge.bs2.
- ✓ Modifique el programa para que iguale el programa ejemplo. Esto involucrará agregar variables, modificar las declaraciones **`IF…THEN`**, y anidar los comandos **`PULSOUT`** dentro de un ciclo **`FOR…NEXT`**. Tenga cuidad y asegure que todos los valores de las variables **`pulseLeft`** y **`pulseRight`** dentro de la declaración **`IF…THEN`** sean adecuadamente ajustados. Sus valores son diferentes de los de FastIrNavigation.bs2 porque las reglas de ruta son diferentes.
- ✓ Reconecte su tarjeta y sus servos.
- ✓ Pruebe el programa en su ruta de cinta eléctrica.
- ✓ Si decide intentarlo sobre una mesa, recuerde seguir los tips de prueba y vigilancia discutidos anteriormente.

```
' Robotica con el Boe-Bot - AvoidTableEdge.bs2
' IR detecta la orilla de objeto y navega para evitar caída.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

irDetectLeft   VAR     Bit               ' Declaracion de Variables.
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word
loopCount      VAR     Byte
pulseCount     VAR     Byte

FREQOUT 4, 2000, 3000                    ' Señal de programa en inicio/reset.
```

```
DO                                        ' Rutina Principal.

  FREQOUT 8, 1, 38500                     ' Checa detectores IR.
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                          ' Decide navegacion.

  IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
    pulseCount = 1                        ' Ambos detectados,
    pulseLeft = 850                       ' un pulso al frente.
    pulseRight =  650
  ELSEIF (irDetectRight = 1) THEN         ' Derecha no detectado,
    pulseCount = 10                       ' 10 pulsos a la izquierda.
    pulseLeft = 650
    pulseRight =  650
  ELSEIF (irDetectLeft = 1) THEN          ' Izquierda no detectado,
    pulseCount = 10                       ' 10 pulsos a la derecha.
    pulseLeft = 850
    pulseRight =  850
  ELSE                                       ' Ninguno detectado,
    pulseCount = 15                          ' retrocede y vuelve a intentar.
    pulseLeft = 650
    pulseRight =  850
  ENDIF

  FOR loopCount = 1 TO pulseCount            ' Envía pulseCount pulsos
    PULSOUT 13,pulseLeft
    PULSOUT 12,pulseRight
    PAUSE 20
  NEXT

LOOP
```

Cómo trabaja AvoidTableEdge.bs2

Puesto que este programa is una versión modificada de FastIrRoaming.bs2, solo los cambios al programa son discutidos aquí.

Se agrega un ciclo **`FOR…NEXT`** al programa para controlar cuántos pulsos se entregan cada vez a través de la Rutina Perincipal (**`DO…LOOP`**). Dos variables son agregadas, **`loopCount`** funciona como un índice para un ciclo **`FOR…NEXT`** y **`pulseCount`** es usado como el argumento ***EndValue***.

```
      loopCount      VAR     Byte
      pulseCount     VAR     Byte
```

Los **IF…THEN** ahora establecen el valor de **pulseCount** así como **pulseRight** y **pulseLeft**. Si ambos detectores pueden ver la mesa, toma un precavido pulso al frente.

```
IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
    pulseCount = 1
    pulseLeft = 850
    pulseRight =  650
```

De lo contrario, si el detector IR derecho no ve la mesa, rota a la izquierda 10 pulsos.

```
ELSEIF (irDetectRight = 1) THEN
  pulseCount = 10
  pulseLeft = 650
  pulseRight =  650
```

De lo contrario, si el detector IR izquierdo no ve la mesa, rota a la derecha 10 pulsos.


```
ELSEIF (irDetectLeft = 1) THEN
  pulseCount = 10
  pulseLeft = 850
  pulseRight =  850
```

De lo contrario, si ningún detector puede ver la mesa, retrocede 15 pulsos y vuelve a intentar, esperando que uno de los detectores vea la condición de caída antes que el otro.

```
ELSE
  pulseCount = 15
  pulseLeft = 650
  pulseRight =  850
ENDIF
```

Ahora que los valores de **pulseCount**, **pulseLeft**, y **pulseRight** han sido establecidos, este ciclo **FOR…NEXT** entrega el número especificado de pulsos para la maniobra determinada por las variables **pulseLeft** y **pulseRight**.

```
FOR loopCount = 1 TO pulseCount
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 20
NEXT
```

Su Turno

Puede experimentar estableciendo diferentes valores **pulseLeft**, **pulseRight**, y **pulseCount** dentro de la declaración **IF…THEN**. Por ejemplo, si el Boe-Bot no gira lo

suficiente, puede de hecho seguir la orilla de la cinta eléctrica. Pivotear hacia atrás en vez de rotar en su sitio puede llevar a comportamientos interesantes.

- ✓ Modifique AvoidTableEdge.bs2 para que siga la orilla del curso delimitado por la cinta eléctrica ajustando los valores **`pulseCount`** para que el Boe-Bot no gire muy lejos de la orilla.
- ✓ Experimente con el pivoteo como una manera de hacer que el Boe-Bot viaje dentro del perímetro en vez de seguir la orilla.

RESUMEN

Este Capítulo cubrió una técnica única para la detección infrarroja de objetos que usa el LED infrarrojo encontrado en controles remotos comunes y el detector infrarrojo encontrado en TVs, CD/DVD y otros dispositivos controlados por estos remotos. Al incidir infrarrojo en el camino del Boe-Bot y buscando su reflejo, se puede lograr la detección de objetos sin entrar en contacto físico con el objeto. Se usan circuitos LED infrarrojo para enviar una señal de 38.5 kHz con la ayuda de una propiedad del comando **`FREQOUT`** llamada armónica, que es inherente a las señales sintetizadas digitalmente.

Un programa indicador de detección de infrarrojo fue presentado para prueba remota (no conectada a la PC) de los pares IR LED/detector. También se presentó un programa "olfateador" de interferencia infrarroja para ayudar a detectar interferencia que puede ser generada por algunas lámparas fluorescentes. Puesto que las señales enviadas por los detectores IR son muy similares a las señales enviadas por los filamentos, RoamingWithWhiskers.bs2 fue adaptado a los detectores infrarrojos. Se presentó un programa que checa los detectores IR entre cada pulso de servo para demostrar una forma de navegar de mayor desempeño sin colisionar con los objetos. Este programa fue entonces modificado para evitar la orilla de un área delimitada por cinta eléctrica. Puesto que la cinta eléctrica absorbe IR, enmarcar una hoja grande de cartulina emula la condición de caída que es vista en la orilla de una mesa sin peligro para el Boe-Bot.

Preguntas

1. ¿Cuál es la frecuencia de la señal enviada por **`FREQOUT 2, 1, 38500`**? ¿Cuál es el valor de la segunda frecuencia enviada por ese comando? ¿Qué tanto tiempo se envían estas señales? ¿A qué pin I/O debe ser conectado el circuito LED IR a fin de emitir esta señal?

2. ¿Qué comando tiene que seguir inmediatamente al comando **`FREQOUT`** a fin de determinar con precisión si un objeto ha sido detectado?
3. ¿Qué significa si el detector IR envía una señal baja? ¿Qué significa cuando el detector envía una señal alta?
4. ¿Qué pasa si cambia el valor de una resistencia en serie con un LED rojo? ¿Qué pasa si cambia el valor de una resistencia en serie con un LED infrarrojo?

Ejercicios

1. Modifique una línea de código en en IrInterferenceSniffer.bs2 para que monitoree solo uno de los detectores IR.
2. Explique la function de **`pulseCount`** en AvoidTableEdge.bs2.

Proyectos

1. Diseñe una aplicación Boe-Bot para que se quede quieto hasta que agite su mano enfrente de el, luego empiece a navegar.
2. Diseñe una aplicación Boe-Bot para que gire lentamente en su lugar hasta que detecte un objeto. Tan pronto lo detecte se fije a el y lo siga. Este es un clásico comportamiento SumoBot.
3. Diseñe una aplicación Boe-Bot para que navegue, pero si detecta interferencia infrarroja, suene la alarma brevemente, y luego continúe navegando. Esta alarma debe ser diferente a la alarma de batería baja.

Soluciones

Q1. 38.5 kHz es la frecuencia de la señal. La segunda frecuencia = 65536 – 38500 = 27036 Hz. Las señales son enviadas por 1 milisegundo, y el LED IR debe ser conectado al Pin I/O 2.

Q2. El comando que guarda la salida del detector en una variable. Por ejemplo, **`irDetectLeft = IN9`**.

Q3. Una señal baja significa que fué detectado IR a 38.5 kHz, por lo tanto, un objeto fue detectado. Una señal alta signidfica que no se detectó IR a 38.5kHz, entonces, no hay objeto.

Q4. Electrícamente hablando, para ambos LEDs rojo e infrarrojo, una resistencia más pequeña causará que el LED brille más intensamente. Una resistencia mayor resulta en LEDs atenuados. En términos de resultados, LEDs IR mas brillantes hacen posible detectar objetos que están más lejos.

E1. Cambie el **`IF…THEN`** para leer.

```
  IF (IN0 = 0) THEN
```

E2. El programa establece esta variable en 1 cuando está tomando un pulso al frente. De esa manera, al avanzar el Boe-Bot, checa si hay condición de caída entre cada pulso. Cuando la detecta, ejecuta una vuelta por un cierto número de pulsos, que también están determinados por el valor de la variable **`pulseCount`**.

P1. El programa FastIrRoaming.bs2 puede ser combinado con un ciclo **`DO…LOOP UNTIL`** que no hace nada hasta que detecta un objeto. Se muestra a continuación una solución ejemplo.

```
' -----[ Titulo ]-----------------------------------------------------
' Robotica con el Boe-Bot - MotionActivatedBoeBot.bs2
' Boe-Bot navega cuando agita su mano en frente de detectores IR.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]---------------------------------------------------

irDetectLeft   VAR     Bit                 ' Declaraciones de Variable
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word

' -----[ Inicializacion ]----------------------------------------------

DEBUG "Program Running!"
FREQOUT 4, 2000, 3000                       ' señal programa
start/reset.

' -----[ Rutina Principal ]--------------------------------------------

Main:
' Ciclo hasta que algo es detectado
  DO
    GOSUB Check_IRs
  LOOP UNTIL (irDetectLeft = 0) OR (irDetectRight = 0)
' Ahora empieza navegacion -- este codigo es de FastIrRoaming.bs2
  DO
    IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
      pulseLeft = 650                       ' Ambos detectan
      pulseRight = 850                      ' Retrocede
    ELSEIF (irDetectLeft = 0) THEN          ' Izquierda detecta
      pulseLeft = 850                       ' Vuelta a la derecha
      pulseRight = 850
    ELSEIF (irDetectRight = 0) THEN         ' Derecha detecta
      pulseLeft = 650                       ' Vuelta a la izquierda
      pulseRight = 650
```

```
    ELSE                                      ' Nada detectado
      pulseLeft = 850                         ' Al frente
      pulseRight = 650
    ENDIF

    PULSOUT 13, pulseLeft                     ' Aplica el pulso.
    PULSOUT 12, pulseRight
    PAUSE 15

    GOSUB Check_IRs                           ' Checa IRs otra vez
  LOOP

' -----[ subrutinas ] ------------------------------------------------

Check_IRs:
  FREQOUT 8, 1, 38500                         ' Checa Detectores IR
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  IrDetectRight = IN0
  RETURN
```


P2. Este comportamiento es en muchas formas lo opuesto al comportamiento de navegación cubierto en este Capítulo. En vez de evitar objetos, el Boe-Bot trata de ir hacia los objetos. Por esta razón, el código principal puede ser derivado de "FastIrRoaming.bs2", con un bit agregado que hace girar al Boe-Bot lentamente hasta que un objeto es detectado. En la siguiente solución, una vez que el Boe-Bot ha encontrado un objeto, continuará al frente aún si ambos detectores leen "no hay objeto" (1) por unos cuantos ciclos. Esto es porque, al manibrar el Boe-Bot hacia el objeto, algunas veces los detectores leen "no hay objeto" por breves momentos, pero esta no es razón suficiente para abandonar la persecución.

```
' Robotica con el Boe-Bot - SumoBoeBot.bs2
' Busca un objeto, lo fija y lo persigue.
' {$STAMP BS2}
' {$PBASIC 2.5}

irDetectLeft   VAR     Bit                  ' IR izq leyendo
irDetectRight  VAR     Bit                  ' IR der leyendo
pulseLeft      VAR     Word                 ' valores pulso para servos
pulseRight     VAR     Word

' -----[ Inicializacion ]---------------------------------------------

DEBUG "Program Running!"
FREQOUT 4, 2000, 3000                        ' señal de inicio/reset.

' -----[ Rutina Principal ]--------------------------------------------
```

```
Main:

' Gira lentamente hasta que un objeto es marcado
  DO
    PULSOUT 13, 790                         ' Gira lentamente
    PULSOUT 12, 790
    PAUSE 15                                ' 5 ms para detectores
    GOSUB Check_IRs                         ' Mientras busca objeto
  LOOP UNTIL (irDetectLeft = 0) OR (irDetectRight = 0)

' Ahora encuentra exactamente donde esta el objeto y lo sigue
  DO
  ' objeto en ambos detectores – va al frente
    IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
      pulseLeft =  850                      ' Al frente
      pulseRight = 650
  ' objeto a la izq – ir a la izq
    ELSEIF (irDetectLeft = 0) THEN
      pulseLeft =  650                      ' Izq hacia objeto
      pulseRight = 650
  ' objeto a la der – ir a la der
    ELSEIF (irDetectRight = 0) THEN
      pulseLeft =  850                      ' Der hacia objeto
      pulseRight = 850
  ' No objeto – al frente , porque los detectores momentaneamente
    ELSE                                    ' mostraran
      pulseLeft =  850                      ' "no object" mientras
      pulseRight = 650                      ' el Boe-Bot ajusta
    ENDIF                                   ' su posicion.

    PULSOUT 13,pulseLeft                    ' Aplica el pulso.
    PULSOUT 12,pulseRight
    PAUSE 15                                ' 5 ms para detectores

  ' Checa IRs nuevamente en caso de que un objeto este moviéndose
    GOSUB Check_IRs
  LOOP

' -----[ subrutinas ] -------------------------------------------------

Check_IRs:
  FREQOUT 8, 1, 38500                       ' Checa Detectores IR
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  IrDetectRight = IN0
  RETURN
```

P3. La clave para resolver este problema es combinar "FastIrRoaming.bs2" y "IrInterferenceSniffer.bs2" en un solo programa.

```
' -----[ Titulo ]-----------------------------------------------------
```

```
' Robotica con el Boe-Bot - RoamAndSniffBoeBot.bs2
' Boe-Bot viaja en circulo y suena una alarma cuando IR son detectados.

' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]-------------------------------------------------

irDetectLeft   VAR     Bit                  ' sensor IR izq leyendo
irDetectRight  VAR     Bit                  ' sensor IR der leyendo
pulseLeft      VAR     Word                 ' pulsos enviados a servos
pulseRight     VAR     Word
counter        VAR     Nib                  ' Contador de Ciclos

' -----[ Inicializacion ]--------------------------------------------

DEBUG "Program Running!"
FREQOUT 4, 2000, 3000                       ' señal programa
start/reset.

' -----[ Rutina Principal ]---------------------- --------------------

Main:
DO
  GOSUB Roam
  GOSUB Sniff
LOOP

' -----[ subrutinas ] ------------------------------------------------

Sniff:                              ' De IrInterferenceSniffer.bs2
  IF (IN0 = 0) OR (IN9 = 0) THEN
    FOR counter = 1 TO 5                    ' Beep 5 veces
      HIGH 1                                ' y flash LEDs
      HIGH 10
      FREQOUT 4, 50, 4000
      LOW 1
      LOW 10
      PAUSE 20
    NEXT
  ENDIF
  RETURN

Roam:                                       ' De FastIrRoaming.bs2
  FREQOUT 8, 1, 38500                       ' Checa Detectores IR
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                            ' Decide como navegar.
  IF (irDetectLeft = 0) y (irDetectRight = 0) THEN
    pulseLeft = 650
    pulseRight = 850
```

```
  ELSEIF (irDetectLeft = 0) THEN
    pulseLeft = 850
    pulseRight = 850
  ELSEIF (irDetectRight = 0) THEN
    pulseLeft = 650
    pulseRight = 650
  ELSE
    pulseLeft = 850
    pulseRight = 650
  ENDIF

  PULSOUT 13,pulseLeft                        ' Aplica el pulso.
  PULSOUT 12,pulseRight
  PAUSE 15
  RETURN
```

Capítulo 8: Control de Robot con Detección de Distancia

En el Capítulo 7, usamos los LEDs y receptores infrarrojos para detector cuando un objeto está en el camino del Boe-Bot sin tener que tocarlo. ¿No sería agradable también saber qué tan lejos está el objeto? Esta es usualmente una tarea para un sonar, que manda un pulso de sondeo y registra cuánto tiempo toma al eco regresar. Este tiempo puede ser entonces usado para calcular qué tan lejos está el objeto. Sin embargo, hay una forma de lograr la detección de distancia con el mismo circuito que usó en el Capítulo anterior. Con su Boe-Bot capaz de determinar la distancia de un objeto, puede ser programado para seguir un objeto en movimiento sin golpearlo. El Boe-Bot también puede ser programado para seguir marcas oscuras sobre un fondo blanco.

DETERMINANDO DISTANCIA CON EL MISMO CIRCUITO DETECTOR IR LED

Usará el mismo circuito del Capítulo anterior para detectar distancia.

- ✓ Si el circuito aún está armado en su Boe-Bot, asegúrese de que sus LEDs IR tienen resistencias de 1 kΩ en serie.
- ✓ Si ya desarmó el circuito del Capítulo previo, repita los pasos del Capítulo 7, Actividad #1, en la página 223.

Recommended Equipment y Materials:

(1) Ruler
(1) Sheet of paper

ACTIVIDAD #1: PROBANDO LA FRECUENCIA DE BARRIDO

La Figura 8-1 muestra un extracto de la hoja de especificación de un detector IR específico (Panasonic PNA4602M; en su kit puede haber una marca diferente). Este extracto es una gráfica que muestra cuánto menos sensible se hace este detector IR si la señal IR que receibe parpadea encendiendo/apagando a una frecuencia distinta a 38.5 kHz. Por ejemplo, si envía IR parpadeando on/off a 40 kHz, solo es 50% sensible de lo que sería a 38.5 kHz. Si la señal IR parpadea on/off a 42 kHz, el detector es solo 20% sensible. Especialmente cierto para frecuencias que hacen al detector menos sensible, el objeto tiene que estar más cercano para hacer que la luz IR reflejada sea más brillante para el detector y que pueda detectarlo.

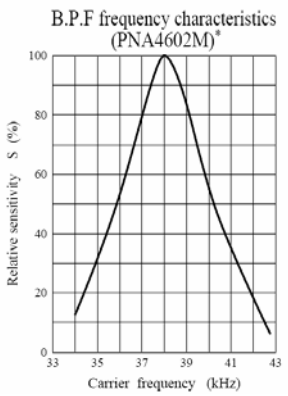


Figura 8-1
La sensibilidad del filtro depende de la frecuencia de la portadora

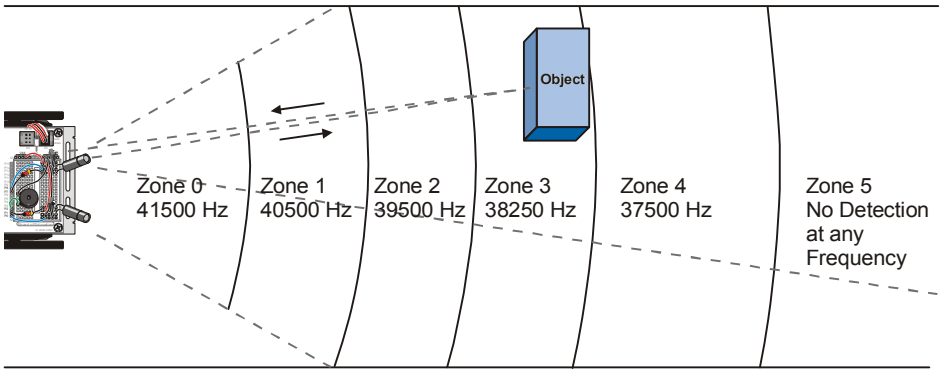
Otra forma de visualizarlo es que la frecuencia más sensible detectará los objetos que están más lejos, mientras que las frecuencias menos sensibles solo pueden ser usadas para detectar objetos cercanos. Esto simplifica la detección de distancia. Escoja 5 frecuencias, luego pruébelas de la más a la menos sensible. Intente primero a la frecuencia más sensible. Si un objeto es detectado, revise y véa si la siguiente frecuencia más sensible hace detección. Dependiendo de la frecuencia que haga que el infrarrojo reflejado deje de ser visible al detector IR, puede inferir la distancia.

Barrido de frecuencia es la técnica de probar la salida de un circuito usando una variedad de frecuencias de entrada.

Programando la frecuencia de barrido para Detección de distancia

La Figura 8-2 muestra un ejemplo de cómo el Boe-Bot puede probar distancia usando frequencia. En este ejemplo, el objeto esta en la Zona 3. Esto quiere decir que el objeto puede ser detectado cuando 37500 y 38250 Hz es transmitido, pero no puede ser detectado con 39500, 40500, y 41500 Hz. Si fuese a mover el objeto a la Zona 2, entonces el objeto puede ser detectado cuando son transmitidos 37500, 38250, y 39500 Hz, pero no cuando son transmitidos 40500 y 41500 Hz.

Figura 8-2 detección de distancia Frecuencias y Zonas para el Boe-Bot



Quizá se pregunte porqué el valor de la zona 4 es 37.5 kHz y no 38.5 kHz. La razón de porqué no son los valores que esperaría basado en la gráfica de % de sensibilidad es porque el comando **FREQOUT** transmite una señal lígeramente más poderoas a 37.5 kHz de lo que lo hace a 38.5 kHz. Las frecuencias listadas en la Figura 8-2 son frecuencias a las programará el BASIC Stamp para determinar la distancia a un objeto.

A fin de probar el detector IR a cada frecuencia, necesitará usar **FREQOUT** para enviar 5 frecuencias diferentes y probar cada una para averiguar si el detector IR puede ver el objeto. Los pasos entre cada frecuencia no son lo suficientemente parejos como para usar la opción **STEP** de un ciclo **FOR…NEXT**. Podría usar **DATA** y **READ**, pero sería poco manejable. Puede usar 5 comandos **FREQOUT** diferentes, pero eso sería un desperdicio de espacio de código. En vez de esto, el major camino para guardar una lista de valores corta que desee usar en secuencia es un comando llamado **LOOKUP**. La sintaxis del comando **LOOKUP** es:

LOOKUP ***Index*****, [*****Value0*****, *****Value1*****, …*****ValueN*****], *****Variable***

Si el argumento ***Index*** es 0, ***Value0*** de la lista dentro de los paréntesis cuadrados será colocado en ***Variable***. Si ***Index*** es 1, ***Value1*** de la lista será colocado en ***Variable***. Puede haber hasta 256 valores en la lista, pero para el siguiente programa ejemplo, solo necesitaremos 5. He aquí cómo será usado:

```
FOR freqSelect = 0 TO 4
  LOOKUP freqSelect,[37500,38250,39500,40500,41500],irFrequency
  FREQOUT 8,1, irFrequency
  irDetect = IN9
    ' Comandos not shown...
NEXT
```

La primera pasada a través del ciclo **`FOR…NEXT`**, **`freqSelect`** es 0, entonces el comando **`LOOKUP`** coloca el valor 37500 en la variable **`irFrequency`**. Puesto que **`irFrequency`** contiene 37500 después del commando **`LOOKUP`**, el comando **`FREQOUT`** envía esa frecuencia al LED IR conectado a P8. Como en el Capítulo previo, el valor de **`IN9`** es entonces guardado en la variable **`irDetect`**. La segunda vez a través del ciclo **`FOR…NEXT`**, el valor de **`freqSelect`** es ahora 1, lo que significa que el comando **`LOOKUP`** coloca 38250 en la variable **`irFrequency`**, y el proceso se repite para este mayor valor de frequencia. La tercera vez se repite de nuevo con 39500, etc. El resultado es notable, especialmente considerando que está usando partes que fueron diseñadas para un propósito completamente diferente, que es establecer comunicación IR entre un control remoto y una television.

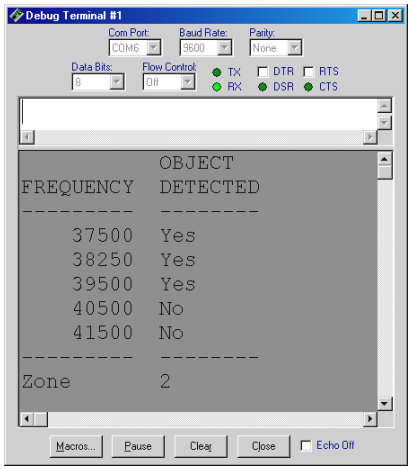
Programa Ejemplo – TestLeftFrequencySweep.bs2

TestLeftFrequencySweep.bs2 hace dos cosas. Primero, prueba el detector IR de objetos izquierdo (conectado a P8 y P9) para asegurarse que funciona adecuadamente para la detección de distancia. Sin embargo, también demuestra cómo se logra la frecuencia de barrido ilustrada en la Figura 8-2.

Cuando corra el programa, la Terminal de Depuración desplegará su zona de medición. Hay muchos posibles patrones si-no que pueden ser generados; se muestran dos en la Figura 8-3. Los patrones de prueba variarán según las características del filtro dentro del detector IR.

El programa determina en qué zona está el objeto detectado contando el número de ocurrencias “No”. Note que aún cuando los dos patrones de prueba de la Terminal de Depuración en la Figura 8-3 son diferentes, ambos tienen tres ocurrencias “Yes” y dos “No”. Por lo tanto, “Zone 2” es la ubicación del objeto detectado en ambos ejemplos.

- ✓ Introduzca, salve y corra TestLeftFrequencySweep.bs2.
- ✓ Use una hoja de papel or tarjeta viendo hacia el LED/detector IR para probar la detección de distancia.
- ✓ Empiece con la hoja muy cerca al LED IR, quizá a ¼ pulgada (o 1 cm) de distancia del LED IR. Su zona en la Terminal de Depuración debe ser 0 o 1.
- ✓ Mueva gradualmente la hoja de papel lejos del LED IR y tome nota de cada distancia que causa que el valor de zona se haga mayor.



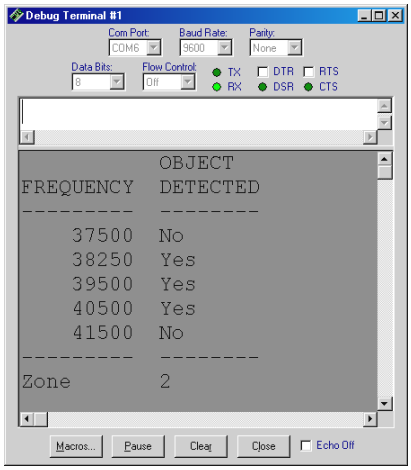


Figura 8-3
Ejemplos de salida al probar la detección de distancia

Recuerde que estas mediciones de distancia son relativas y no necesariamente precisar o equidistantes. Sin embargo, darán al Boe-Bot un buen sentido de distancia al objeto para seguir, rastrear y otras actividades.

Las zonas 1-4 typicamente caerán en el rango de 6 a 12 pulgadas (15 a 30 cm) para los LEDs en su escudo con una resistencia de 1 kΩ. Mientras que los objetos puedan ser detectadoa hasta 4 pulgadas (10 cm) de distancia, los experimentos en este Capítulo trabajarán. Si el rango del detector de distancia es menos que esto, intenten reducir su resistencia serie de 1 kΩ a 470 Ω o 220 Ω.

```
' -----[ Titulo ]-----------------------------------------------------------
' Robotica con el Boe-Bot - TestLeftFrequencySweep.bs2
' Prueba las respuestas del detector IR de distancia al barrido de frecuencia.

' {$STAMP BS2}                                ‘ Directiva Stamp.
' {$PBASIC 2.5}                               ‘ Directiva PBASIC.

' -----[ Variables ]-------------------------------------------------------

freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetect       VAR     Bit
distance       VAR     Nib

' -----[ Inicializacion ]--------------------------------------------------

DEBUG CLS,
      "           OBJECT", CR,
      "FREQUENCY  DETECTED", CR,
      "---------  --------"
```

```
' -----[ Rutina Principal ]-------------------------------------------------

DO

  distancia = 0

  FOR freqSelect = 0 TO 4

    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency
    FREQOUT 8,1, irFrequency
    irDetect = IN9
    distancia = distancia + irDetect

    DEBUG CRSRXY, 4, (freqSelect + 3), DEC5 irFrequency
    DEBUG CRSRXY, 11, freqSelect + 3

    IF (irDetect = 0) THEN DEBUG "Yes" ELSE DEBUG "No "

    PAUSE 100

  NEXT

  DEBUG CR,
        "---------  --------", CR,
        "Zone       ", DEC1 distance
LOOP
```

Su Turno – Probando el LED/Detector IR Derecho del Detector de Objetos

Aún cuando hay algo de etiquetado en cuestión, puede modificar este programa para probar el LED y detector IR derecho cambiando estas dos líneas:

```
FREQOUT 8,1, irFrequency
irDetect = IN9
```

...para que se lean:

```
FREQOUT 2,1, irFrequency
irDetect = IN0
```

- ✓ Modifique TestLeftFrequencySweep.bs2 para probar la medición de distancia del detector de objetos IR derecho.
- ✓ Corra el programa y verifique que puede medir una distancia similar.

Desplegando ambas Distancias

A veces es util tener un programa rápido con el que pueda probrar ambos detectores de distancia al mismo tiempo. Este programa esta organizado en subrutinas, las cuales pueden ser útiles para copiar y pegar en otros programas que requieran detección de distancia.

Programa Ejemplo – DisplayBothDistances.bs2

- ✓ Introduzca, salve y corra DisplayBothDistances.bs2.
- ✓ Repita el ejercicio de medición de distancia con una hoja de papel en cada LED, luego en ambos LEDs al mismo tiempo.

```
' -----[ Titulo ]-------------------------------------------------------------
' Robotica con el Boe-Bot - DisplayBothDistances.bs2
' Prueba respuestas del detector IR de distancia de ambos detectores IR de
' objetos al barrido de frecuencia.

' {$STAMP BS2}                              ‘ Directiva Stamp.
' {$PBASIC 2.5}                             ‘ Directiva PBASIC.

' -----[ Variables ]-----------------------------------------------------------

freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
distanceLeft   VAR     Nib
distanceRight  VAR     Nib

' -----[ Inicializacion ]-------------------------------------------------------

DEBUG CLS,
      "IR objeto ZONE", CR,
      "Left   Right", CR,
      "-----  -----"

' -----[ Rutina Principal ]-----------------------------------------------------

DO

  GOSUB Get_Distances
  GOSUB Display_Distances

LOOP
```

```
' -----[ Subrutina – Get_Distances ]-----------------------------------------

Get_Distances:

  distanceLeft = 0
  distanceRight = 0

  FOR freqSelect = 0 TO 4

    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency

    FREQOUT 8,1,irFrequency
    irDetectLeft = IN9
    distanceLeft = distanceLeft + irDetectLeft

    FREQOUT 2,1,irFrequency
    irDetectRight = IN0
    distanceRight = distanceRight + irDetectRight

    PAUSE 100

  NEXT

  RETURN

' -----[ Subrutina – Display_Distances ]-------------------------------------

Display_Distances:

  DEBUG CRSRXY,2,3, DEC1 distanceLeft,
        CRSRXY,9,3, DEC1 distanceRight
  RETURN
```

Su Turno – Mas Pruebas de Distancia

- ✓ Intente medir la distancia de diferentes objetos y averigue si el color y/o textura causan alguna diferencia a la medición de distancia.

ACTIVIDAD #2: VEHÍCULO SOMBRA BOE-BOT

Para que un Boe-Bot siga a otro, el Boe-Bot seguidor, conocido también como vehículo sombra, tiene que saber qué tan al frente está el vehículo líder. Si el vehículo sombra se rezaga, tiene que detectar esto y acelerar. Si el vehículo sombra está muy cerca del vehículo líder, tiene que detectar esto y desacelerar. Si es la distancia adecuada, puede esperar hasta que las mediciones indiquen que está muy lejos o muy cerca nuevamente.

La distancia es solo una clase de valor de la que los robots y otra maquinaria autómata es responsable. Cuando una máquina está diseñada para mantener automáticamente un

valor, como la distancia, presión o nivele de fluido, generalmente involuca un sistema de control. Algunas veces estos sistemas consisten en sensores y válvulas, o sensores y motores, o, en el caso del Boe-Bot, sensores y servos de rotación continua. También hay un tipo de procesador que tom alas mediciones de los sensors y las convierte en acciones mecánicas. El procesador tiene que ser programado para tomar decisiones basado en los datos de entrada de los sensores y luego controlar las salidas mecánicas de acuerdo a aquellas. En el caso del Boe-Bot, el procesador es el BASIC Stamp 2.

Un control de ciclo cerrado es un método común de mantener niveles y trabaja muy bien para ayudar al Boe-Bot a mantener su distancia a un objeto. Hay muchas clases diferentes de controles de ciclo cerrado. Algunas de las más comunes son control de histéresis, proporcional, integral, y derivativo. Todos estos tipos de control se presentan a detalle en el texto de Control de Proceso de Stamps in Class, listado en el Prefacio.

La mayoría de las técnicas de control pueden implementarse con pocas líneas de código en PBASIC. De hecho, la mayoría de los ciclos de control proporcional como el mostrado en la Figura 8-4 se reducen a solo una línea de código PBASIC. Este diagram es llamado diagram de bloques, y describe los pasos del proceso de control proporcional que el Boe-Bot usará para medir distancia con su LED y detector IR derecho y ajustar su posición para mantener la distancia con su servo derecho.

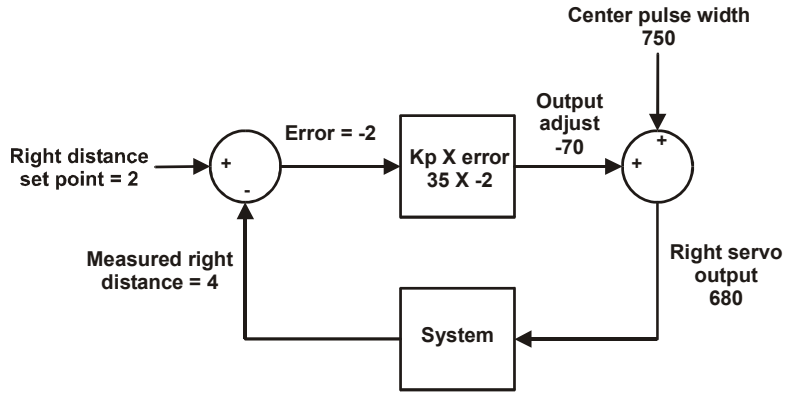


Figura 8-4
Diagrama de Bloques de Control Proporcional para Servo Derecho y Detector IR de objetos

Echemos un vistazo a los números en la Figura 8-4 para aprender cómo trabaja el control proporcional. Este ejemplo particular es para el LED/detector IR derecho y servo derecho. El valor de configuración es 2, lo que quiere decir qu queremos que el Boe-Bot mantenga una distancia de 2 entre sí mismo y cualquier objeto que detecte. La distancia medida es 4, lo cual está muy lejos. El error es el valor de configuración menos la

distancia medida, esto es 2 – 4 = -2. Esto se indica por los símbolos dentro del círculo a la izquierda. Este círculo es llamado nodo de suma. Después, el error alimenta a un bloque operador. Este bloque muestra que el error será multiplicado por un valor llmado constante proporcional (Kp). El valor de Kp is 35. La salida del bloque muestra un resultado de -2 × 35 = -70, que es llamado el ajuste de salida. Este ajuste de salida se introduce a otro nodo de suma, y esta vez se suma al ancho de pulso central del servo de 750. El resultado es un ancho de pulso de 680 que hará que el servo gire en sentido a la derecha a aproximadamente ¾ de su velocidad. Esto hace que la rueda derecha del Boe-Bot ruede al frente, hacia el objeto. Esta corrección se aplica al sistema en general, que consiste en el Boe-Bot y el objeto, que estaba a una distancia medida de 4.

La siguiente vez a través del ciclo la distancia medida puede cambiar, pero eso está bien porque, sin importar la distancia medida, este ciclo de control calculará un valor que causará que el servo se mueva para corregir cualquier error. La corrección siempre es proporcional al error, que es la diferencia entre el valor de configuración y las distancias medidas.

Un ciclo de control siempre tiene un conjunto de ecuaciones que gobiernan el sistema. El diagrama de bloques en la Figura 8-4 es una forma de describir visualmente este conjunto de ecuaciones. He aquí las ecuaciones que pueden desprenderse de este diagrama de bloques, junto con sus soluciones.

$$
\begin{aligned}
\textit{Error} &= \textit{Valor de configuración de distancia der} - \textit{Distancia medida derecha} \\
&= 2 - 4 \\
\textit{Ajuste de Salida} &= \textit{error} \times K_p \\
&= -2 \times 35 \\
&= -70 \\
\textit{Salida al servo der} &= \textit{Ajuste de Salida} + \textit{ancho de pulso central} \\
&= -70 + 750 \\
&= 680
\end{aligned}
$$

Haciendo algunas sustituciones, last res ecuaciones de arrib pueden ser reducidas a la siguiente, que le dará el mismo resultado.

$$
\begin{aligned}
\textit{Salida al servo der} &= (\textit{Valor de configuración de distancia der} - \textit{Distancia medida der}) \times Kp \\
&\quad + \textit{Ancho de pulso central}
\end{aligned}
$$

Sustituyendo los valores del Ejemplo, podemos ver que la ecuación aún trabaja:

$$
\begin{aligned}
&= ((2 - 4) \times 35) + 750 \\
&= 680
\end{aligned}
$$

El servo izquierdo y el detector IR de objetos tiene un algoritmo similar mostrado en la Figura 8-5. La diferencia es que Kp es -35 en vez de +35. Asumiendo el mismo valor medido que en el detector IR de objetos derecho, el resultado de ajuste de salida es un ancho de pulso de 820. He aquí la ecuación y cálculos para este diagrama de bloques:

Salida al servo izq = *(Valor de configuración de distancia izq– Distancia medida izq)× Kp + Ancho de pulso central*
= *((2 – 4) × –35) + 750*
= *820*

El resultado de este ciclo de control es un ancho de pulso que hace que el servo izquierdo gire en sentido a la izquierda aproximadamente ¾ de su velocidad. Esto también es un pulso al frente para la rueda izquierda. La idea de retroalimentar es que la salida del sistema vuelva a ser muestreada por el Boe-Bot sombra tomando otra medición de. Luego el ciclo de control se repite una y otra vez…aproximadamente 40 veces por segundo.

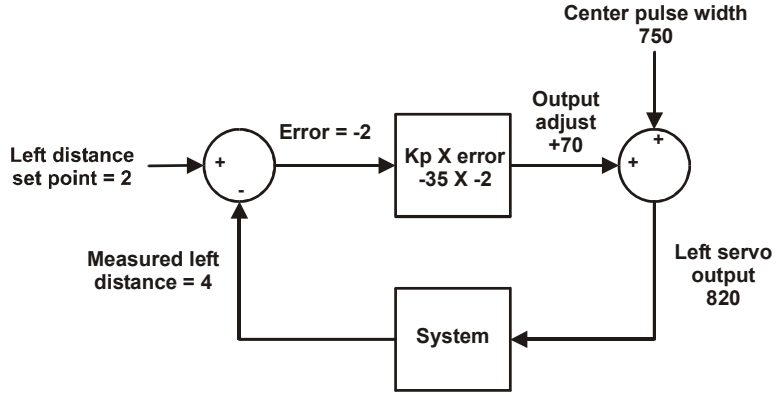


Figura 8-5
Diagrama de bloques de Control Proporcional para el Servo izquierdo y el Detector IR de objetos

Programando el Vehículo Boe-Bot Sombra

Recuerde que la ecuación para la salida al servo derecho fue:

Salida al servo der = *(Valor de configuración de distancia der–Distancia medida der) × Kp + Ancho de pulso central*

He aquí un ejemplo de solución de esta misma ecuación en PBASIC. El valor de configuración de distancia derecha es 2, la distancia medida es una variable llamada

distanceRight que guarde la medición IR de distancia, Kp es 35, y el ancho de pulso dentral es 750:

```
pulseRight = 2 - distanceRight * 35 + 750
```

Recuerde que en PBASIC las expresiones matemáticas se ejecutan de izquierda a derecha. Primero, **distanceRight** es restado a 2. El resultado de esta resta es luego multiplicado por **Kpr**, que es 35, y después de eso, el producto se suma al ancho de pulso central de 750.

Puede usar paréntesis para forzar un cálculo que está mas hacia la derecha en una línea de código de PBASIC para que se complete primero. Recuerde este ejemplo: puede reescribir esta línea de código PBASIC:

```
pulseRight = 2 - distanceRight * 35 + 750
```

...así:

```
pulseRight = 35 * (2 – distanceRight) + 750
```

En esta expresión, 35 es multiplicado por el resultado de (2 – distanceRight), luego el producto se suma a 750.

El servo izquierdo es diferente porque Kp para ese sistema es -35

```
pulseLeft = 2 - distanceLeft * (-35) + 750
```

Since the valores -35, 35, 2, y 750 all have names, it's definitely a good place for some constant declarations.

```
Kpl            CON     -35
Kpr            CON     35
SetPoint       CON     2
CenterPulse    CON     750
```

Con estas declaraciones de constantes en el programa, puede usar el nombre **Kpl** en lugar de -35, **Kpr** en lugar de 35, **SetPoint** en lugar de 2, y **CenterPulse** en lugar de 750. Después de estas declaraciones de constantes, los cálculos del control proportional ahora se ven así:

```
pulseLeft  = SetPoint - distanceLeft  * Kpl + CenterPulse
pulseRight = SetPoint - distanceRight * Kpr + CenterPulse
```

Lo conveniente de declarer constantes estos valores es que puede cambiarlos en un solo lugar, al principio del programa. Los cambios que hace al principio del programa se

reflejarán dondequiera que estas constantes sean usadas. Por ejemplo, cambiando la directive **`Kpl CON`** de -35 a -40, todas las instancias de **`Kpl`** en todo el programa cambian de -35 a -40. Esto es extremádamente útil para experimentar y ajustar los ciclos de control proporcional izquierdo y derecho.

Programa Ejemplo – FollowingBoeBot.bs2

FollowingBoeBot.bs2 repite el ciclo de control proporcional recién discutido con todos los pulsos de servo pulse. En othras palabras, antes de cada pulso la distancia es medida y la señal de error es determinada. Entonces el error es multiplicado por Kp, y el valor resultante es sumado/restado a/de los anchos de pulso que son enviados a los servos izquierdo/derecho.

- ✓ Introduzca, salve y corra FollowingBoeBot.bs2.
- ✓ Apunte el Boe-Bot a una hoja de papel tamaño carta (8 ½ x 11") sostenida al frente como si fuera un obstáculo-pared. El Boe-Bot debe mantener una distancia fija entre si mismo y la hoja de papel.
- ✓ Intente rotar la hoja de papel ligeramente. El Boe-Bot debe rotar con el.
- ✓ Intente usar la hoja de papel para guiar el Boe-Bot en las proximidades. El Boe-Bot debe seguirlo.
- ✓ Mueva la hoja de papel mucho mas cerca al Boe-Bot, debe retroceder alejándose del papel.


```
' -----[ Titulo ]-----------------------------------------------------------
' Robotica con el Boe-Bot - FollowingBoeBot.bs2
' Boe-Bot adjusta su posicion para mantener objetos detectados en zona 2.

' {$STAMP BS2}                              ‘ Directiva Stamp.
' {$PBASIC 2.5}                             ‘ Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Constantes ]-------------------------------------------------------

Kpl             CON     -35
Kpr             CON     35
SetPoint        CON     2
CenterPulse     CON     750

' -----[ Variables ]-------------------------------------------------------

freqSelect      VAR     Nib
irFrequency     VAR     Word
irDetectLeft    VAR     Bit
irDetectRight   VAR     Bit
```

```
distanceLeft   VAR     Nib
distanceRight  VAR     Nib
pulseLeft      VAR     Word
pulseRight     VAR     Word

' -----[ Inicializacion ]-------------------------------------------------

FREQOUT 4, 2000, 3000
' -----[ Rutina Principal ]-------------------------------------------------

DO

  GOSUB Get_Ir_Distances

  ' Calcula salida proporcional.

  pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse

  GOSUB Send_Pulse

LOOP

' -----[ Subrutina - Get IR Distances ]---------------------------------

Get_Ir_Distances:
  distanceLeft = 0
  distanceRight = 0
  FOR freqSelect = 0 TO 4
    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency

    FREQOUT 8,1,irFrequency
    irDetectLeft = IN9
    distanceLeft = distanceLeft + irDetectLeft

    FREQOUT 2,1,irFrequency
    irDetectRight = IN0
    distanceRight = distanceRight + irDetectRight
  NEXT
  RETURN

' -----[ Subrutina – Get Pulse ]-----------------------------------------

Send_Pulse:
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5
  RETURN
```

Como trabaja FollowingBoeBot.bs2

FollowingBoeBot.bs2 declara 4 constantes que usan la directiva **`CON`**: **`Kpr`**, **`Kpl`**, **`SetPoint`**, y **`CenterPulse`**. Dondequiera que vea **`SetPoint`** es de hecho el número 2 (una constante). De igual forma, dondequiera que vea **`Kpl`** es de hecho el número -35. **`Kpr`** es de hecho 35, y **`CenterPulse`** es 750.

```
Kpl            CON     -35
Kpr            CON     35
SetPoint       CON     2
CenterPulse    CON     750
```

Lo primero que la Rutina Principal hace es llamar a la subrutina **`Get_Ir_Distances`**. Después de que termina la subrutina **`Get_Ir_Distances`**, **`distanceLeft`** y **`distanceRight`** contienen cada una un número que corresponde a la zona en la que un objeto fue detectado para ambos detectores IR de objetos izquierdo y derecho.

```
DO

  GOSUB Get_Ir_Distances
```


Las siguientes dos líneas de código implementan los cálculos del control proporcional para cada servo.

```
  ' Calcula salida proporcional.

  pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse
```

Ahora que los cálculos de **`pulseLeft`** y **`pulseRight`** están hechos, la subrutina **`Send_Pulse`** puede ser llamada.

```
  GOSUB Send_Pulse
```

La porción **`LOOP`** del **`DO…LOOP`** manda al programa de regreso al comando inmediatamente siguiente al **`DO`** al principio del ciclo principal.

```
LOOP
```

Su Turno

La Figura 8-6 muestra un Boe-Bot líder seguido por un Boe-Bot sombra. El Boe-Bot líder está corriendo una version modificada de FastIrRoaming.bs2, y el Boe-Bot sombra está corriendo FollowingBoeBot.bs2. El contro proportional hace del Boe-Bot sombra un muy fiel seguidor. Un Boe-Bot líder puede encabezar una cadena de 6 o 7 Boe-Bots sombra. Solo agregue los paneles laterales y compuerta posterior del Boe-Bot al resto de los Boe-Bots sombra en la cadena.

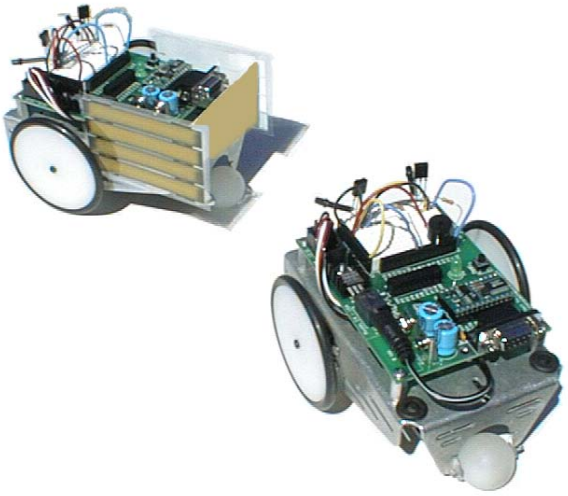


Figura 8-6
Boe-Bot líder (izq) y Boe-Bot sombra (der)

- ✓ Si es parte de una clase, monte paneles de papel en la cola y ambos lados del Boe-Bot líder como se muestra en la Figura 8-6.
- ✓ Si no es parte de una clase (y solo tiene un Boe-Bot) el vehículo sombra seguirá a un pedazo de papel o su mano tan bien como sigue al Boe-Bot.
- ✓ Remplace las resistencias de 1 kΩ que conectan a P2 y P8 del Boe-Bot líder a los LEDs IR con resistencias de 470 Ω o 220 Ω.
- ✓ Programe al Boe-Bot líder para evitar objetos usando una version modificada de FastIrRoaming.bs2, renombrado SlowerIrRoamingForLeadBoeBot.bs2.
- ✓ Haga estas modificaciones a SlowerIrRoamingForLeadBoeBot.bs2:
  - o Incremente todos los argumentos ***Duration*** de **PULSOUT** que ahora son 650 a 710.
  - o Reduzca todos los argumentos ***Duration*** de **PULSOUT** que ahora son 850 a 790.
- ✓ El Boe-Bot sombra debe corer FollowingBoeBot.bs2 sin modificaciones.

- ✓ Con ambos Boe-Bots corriendo sus respectivos programas, coloque el Boe-Bot sombra detrás del Boe-Bot líder. El Boe-Bot sombra debe seguir a una distancia fija, mientras que no sea distraído por otro objeto como una mano o una pared cercana.

Puede ajustar los valores de configuración y las constantes de proporcionalidad para cambiar el comportamiento del Boe-Bot sombra. Use su mano o un pedazo de papel para dirigir al Boe-Bot sombra al hacer estos ejercicios:

- ✓ Intente correr FollowingBoeBot.bs2 usando valores de **`Kpr`** y **`Kpl`** constantes, variando de 15 a 50. Note la diferencia en la respuesta del Boe-Bot cuando sigue un objeto.
- ✓ Intente ajustar el valor de la constante SetPoint. Intente valores de 0 a 4.

ACTIVIDAD #3: SIGUIENDO UNA TIRA

La Figura 8-7 muestra un ejemplo de un curso que puede construir y luego programar su Boe-Bot para que la siga. Cada tira en este curso esta hecha con 3 piezas largas de cinta eléctrica de vinil de ¾ de pulgada de ancho (19 mm) colocada de extremo a extremo sobre cartulina blanca. El papel no debe ser visible entre las tiras de cinta eléctrica.

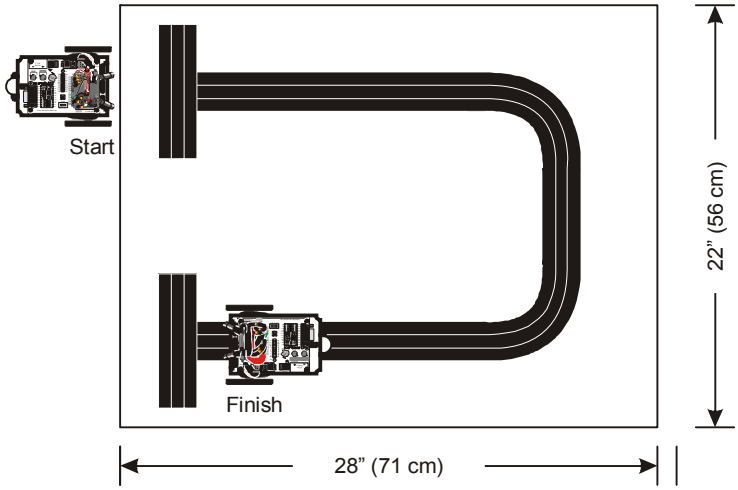


Figura 8-7
Curso de seguimiento de tira

Construyendo y probando el Curso

Para una navegación exitosa en este curso, se requieren hacer algunas pruebas y ajustes.

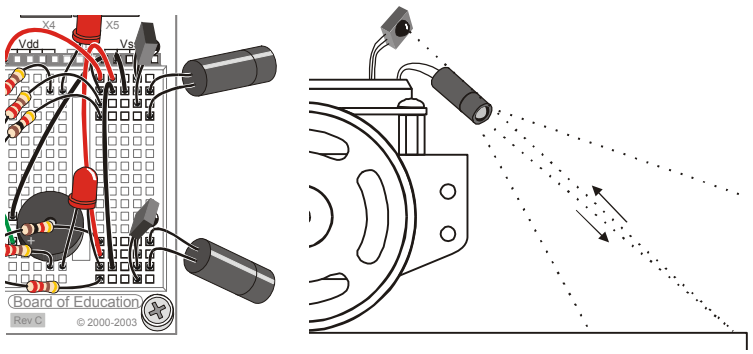
Materiales Requeridos

(1) Hoja de cartulina, dimensiones aproximadas: 22 X 28 pulgadas (56 X 71 cm)
(1) Rollo de cinta eléctrica de vinil negra, ¾" de ancho (19 mm)

- ✓ En su cartulina, use la cinta eléctrica para trazar un curso como se muestra en la Figura 8-7.

Probando la Tira

- ✓ Apunte sus detectores IR de objetos hacia abjo y afuera como se muestra en la Figura 8-8 (Figura 7-11 de la página 242 repetida aquí por comodidad).



Vista superior *Vista lateral*

Figura 8-8
detectores IR de objetos dirigidos hacia abajo para buscar la tira

- ✓ Asegúrese de que su curso de cinta eléctrica tape está libre de interferencia por luz fluorescente. Véa Olfateando Interferencia IR en la página 233.
- ✓ Remplace las resistencias de 1 kΩ en serie con los LEDs IR con resistencias de 2 kΩ para recortar el alcance visual del Boe-Bot.
- ✓ Corra DisplayBothDistances.bs2 de la página 275. Mantenga su Boe-Bot conectado a su cable de programación de tal forma que pueda ver las distancias deplegadas.
- ✓ Comience por colocar su Boe-Bot de tal forma que vea directamente a la cartulina como se muestra en la Figura 8-9.

- ✓ Verifique que sus lecturas de zona indicant que un objeto es detectado en una zona muy cercana. Ambos sensores deben darle una lectura de 1 o 0.

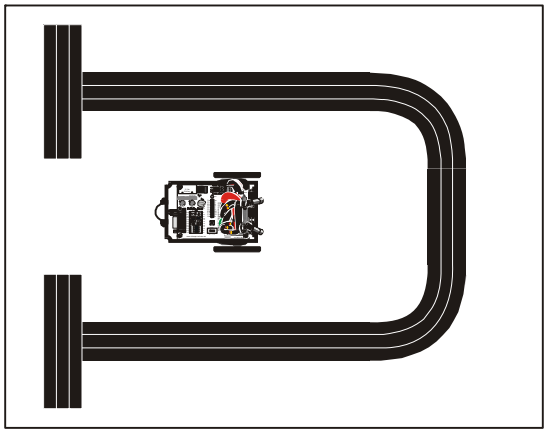


Figura 8-9
Prueba para numerous de zona baja – vista superior

- ✓ Coloque su Boe-Bot para que ambos detectores IR enfoquen directamente al centro de su tira de cinta eléctrica (Véa Figura 8-10 y Figura 8-11).
- ✓ Luego, ajuste la posición de su Boe-Bot (hacia y lejos de la cinta) hasta que ambos valores de zona alcancen el límite de 4 o 5 indicando que un objeto muy lejano es detectado, o que ningún objeto es detectado.
- ✓ Si está teniendo dificultades para obtener las lecturas más grandes con su curso de cinta eléctrica, vea Corrección de problemas en la página siguiente.

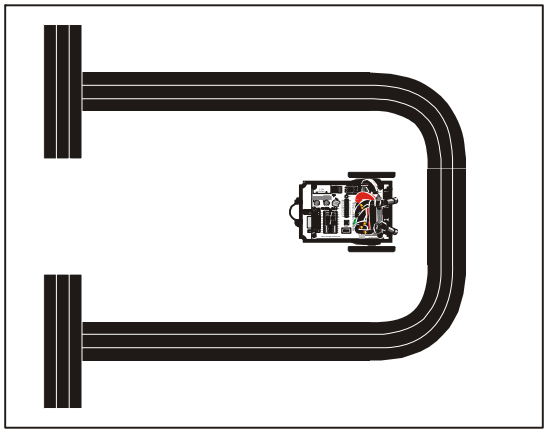


Figura 8-10
Prueba para número de zona alta – Vista Superior

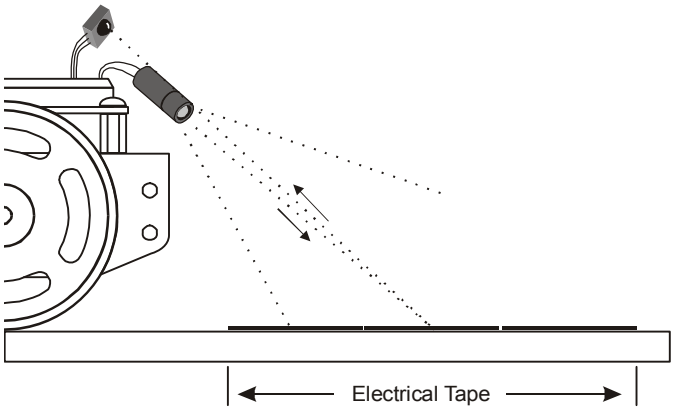


Figura 8-11
Prueba para número de zona alta – vista lateral

Corrección de Problemas del curso de cinta eléctrica

Si no puede obtener un valor de zona alto cuando the detectores IR enfocan a la cinta eléctrica, tome una hoja de papel separada y haga una tira que sea de 4 tiras de ancho en vez de tres. Si los números de zona aún son bajos, asegúrese que está usando resistencias de 2 kΩ (rojo-negro-rojo) en serie con sus LEDs IR. También puede probar una resistencia de 4.7 kΩ para hacer al Boe-Bot mas coto en visión. Si nada de esto funciona, pruebe una marca diferente de cinta eléctrica de vinil negro. También puede ayudar ajustar el LED/detector IR para que enfoque más cerca o más lejos del frente del Boe-Bot (véa la Figura 8-11).

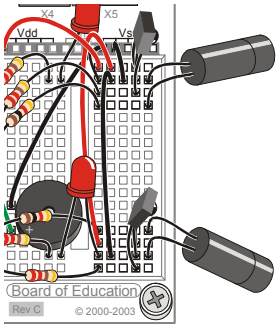
Si está teniendo problemas con las mediciones de zonas bajas al leer la superficie blanca, intente apuntar los LEDs y detectores IR aún más hacia abajo y afuera del frente del Boe-Bot, pero tenga cuidad de no causar reflejo en el chasis. También puede probar un valor más bajo de resistencia como1 kΩ (café-negro-rojo).

- ✓ Ahora, coloque el Boe-Bot en el curso para que sus ruedas estén a ambos lados de la línea negra. Los detectores IR deben estar viendo ligéramente hacia afuera (véa la Figura 8-12). Verifique que la distancia que leen ambos detectores IR es 0 o 1 de nuevo. Si las lecturas son mayores, quiere decir que necesitar ser apuntados ligéramente mas hacia afuera, lejos de la orilla de la cinta.

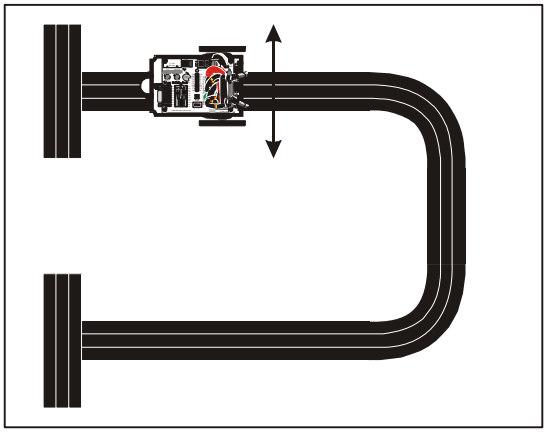
Cuando mueve el Boe-Bot en cualquier dirección indicada por la doble-flecha, uno u otro detector IR de objetos se enfocará en la cinta eléctrica. Cuando hace esto, las lecturas pra el detector de objetos que ahora está sobre la cinta eléctrica debe incrementar a 4 o 5. Recuerde que si mueve el Boe-Bot hacia su izquierda, los detectores derechos deben incrementar su valor y si mueve el Boe-Bot hacia su derechoa, los detectores izquierdos deben mostrar el valor más alto.

- ✓ Adjuste sus detectores IR de objetos hasta que el Boe-Bot pase esta última prueba. Entonces estará listo para tratar seguir la tira.

Figura 8-12: Prueba de búsqueda de tira



Detalle de los detectores IR de objetos



Vista superior del Boe-Bot a ambos lados de la tira

Programación para Seguir la Tira

Solo necesitará hacer algunos cuantos ajustes a FollowingBoeBot.bs2 para hacerlo seguir una tira. Primero, el Boe-Bot debe moverse hacia los objetos más cercanos que **`SetPoint`** y lejos de objetos más lejanos que **`SetPoint`**. Esto es lo opuesto a cómo se comporta FollowingBoeBot.bs2. Para inverter la dirección a la que el Boe-Bot se mueve cuando sensa que el objeto no está a la distancia **`SetPoint`**, simplemente cambie los signos de **`Kpl`** y Kpr. En otras palabras, cambie **`Kpl`** de -35 a 35, y cambie **`Kpr`** de 35 a -35. Necesitará experimentar con su **`SetPoint`**. Valores de 2 a 4 tienden a ser los mejores. El siguiente programa ejemplo usará un SetPoint de 3.

Programa Ejemplo: StripeFollowingBoeBot.bs2

- ✓ Abra FollowingBoeBot.bs2 y sálvelo como StripeFollowingBoeBot.bs2.
- ✓ Cambie la declaración **`SetPoint`** de **`SetPoint CON 2`** a **`SetPoint CON 3`**.
- ✓ Cambie **`Kpl`** de -35 a 35.
- ✓ Cambie **`Kpr`** de 35 a -35.

- ✓ Corra el programa.
- ✓ Coloque su Boe-Bot en la posición "Start" mostrada en la Figura 8-13. El Boe-Bot debe esperar allí hasta que coloque su mano en frente de sus detectores IR de objetos. Entonces avanzará. Cuando pase la tira de inicio retire su mano y debe empezar a seguir la tira. Cuando ve la tira "Finish", debe detenerse y esperar allí.
- ✓ Asumiendo que puede obtener lecturas de distancia de 5 de la cinta eléctrica y 0 de la cartulina, valores constantes **`SetPoint`** de 2, 3, y 4 deben trabajar. Intente diferentes valores **`SetPoint`** y haga notas del desempeño de su Boe-Bot en el camino.

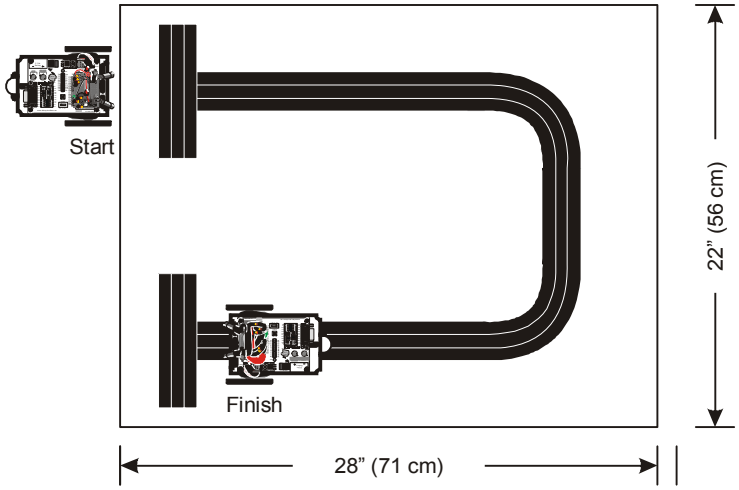


Figura 8-13
Curso de seguimiento de la Tira.

```
' -----[ Titlulo ]-------------------------------------------------------------
' Robotica con el Boe-Bot - StripeFollowingBoeBot.bs2
' Boe-Bot ajusta su posicion para moverse hacia objetos mas cercanos que la
' zona 3 y lejos de objetos mas lejanos que la zona 3. Util para seguir una
' tira de cinta electrica de vinil de 2.25 pulgadas de ancho.
' {$STAMP BS2}                                ‘ Directiva Stamp.
' {$PBASIC 2.5}                               ‘ Directiva PBASIC.

DEBUG "Program Running!"

' -----[ Constantes ]-----------------------------------------------------------

Kpl             CON     35                   ' Cambio de -35 to 35
Kpr             CON     -35                  ' Cambio de 35 to -35
SetPoint        CON     3                    ' Cambio de 2 to 3.
CenterPulse     CON     750
```

```
' -----[ Variables ]------------------------------------------------------

freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
distanceLeft   VAR     Nib
distanceRight  VAR     Nib
pulseLeft      VAR     Word
pulseRight     VAR     Word

' -----[ Inicializacion ]--------------------------------------------------

FREQOUT 4, 2000, 3000

' -----[ Rutina Principal ]-------------------------------------------------

DO

  GOSUB Get_Ir_Distances

  ' Calcula salida proporcional.

  pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse

  GOSUB Send_Pulse

LOOP

' -----[ Subrutina - Get IR Distances ]-------------------------------------

Get_Ir_Distances:
  distanceLeft = 0
  distanceRight = 0
  FOR freqSelect = 0 TO 4
    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency

    FREQOUT 8,1,irFrequency
    irDetectLeft = IN9
    distanceLeft = distanceLeft + irDetectLeft

    FREQOUT 2,1,irFrequency
    irDetectRight = IN0
    distanceRight = distanceRight + irDetectRight
  NEXT
  RETURN
```

```
' -----[ Subrutina - Get Pulse ]-----------------------------------------

Send_Pulse:
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5
  RETURN
```

Su Turno – Concurso de Seguimiento de Tira

Puede convertrir esto en un concurso contra el menor tiempo de curso, dado que el Boe-Bot espera fielmente en las tiras de "Start" y "Finish". Puede hacer otros cursos también. Para un major desempeño, experimente con diferentes valores **`SetPoint`**, **`Kpl`**, y **`Kpr`**.

ACTIVIDAD #4: MAS ACTIVIDADES BOE-BOT Y PROYECTOS EN LÍNEA

Entonces, ¿qué desea hacer a continuación con su Boe-Bot? Algunas posibilidades son:

- Proyectos con accesorios Boe-Bot
- Concursos y retos
- Más actividades con su Boe-Bot usando el juego de piezas que ya tiene
- texto y kit de piezas de IR Remoto para el Boe-Bot

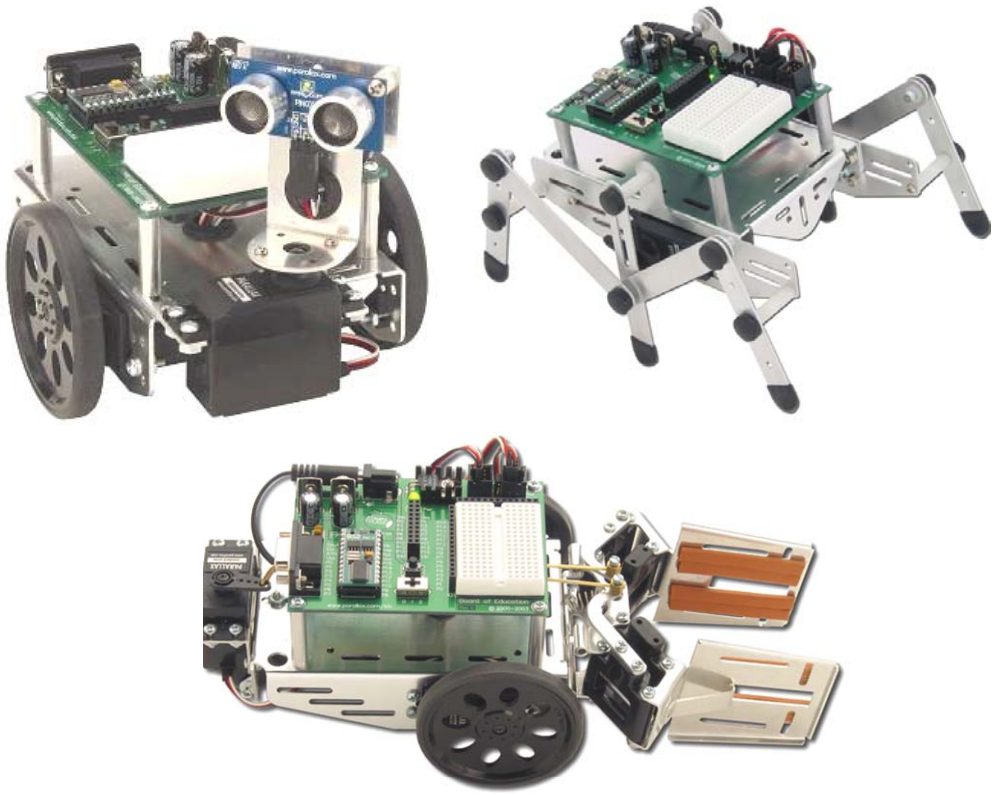
Todos los recursos discutidos en esta Actividad pueden accesarse a través de la página www.parallax.com/go/Boe-Bot.

Accesorios para Proyectos con el Boe-Bot

Parallax tiene sensores adicionales y kits de accessorios para que pueda agregar capacidades y seguir explorando con su Boe-Bot. He aquí algunos ejemplos:

- El Sensor Ultrasónico de distancia Ping))) (#28015) provee un rango mayor y mas exacto de mediciones de distancia a objetos. El kit opcional de montaje (#570-28015) le permite al sensor hacer un barrido de area.
- Acelerómetro Bi-axial para sensar inclinación (#28017)
- Módulo de Bruja para navegación (#29123)
- Kit para hacer a su Boe-Bot un caminador de 6-piernas (#30055)
- Sujetador mecánico para recoger y mover elementos (#28202)
- Kit de banda oruga para navegación en todo terreno (#28106)
- Módulos y adaptadores RF XBee para control y comunicación inalámbrica (Véa www.parallax.com/go/XBee)

Figura 8-14: Accesorios: Sensor y Soporte Ping))), Piernas y Sujetador



Concursos y Retos

¿Interesado en un concurso? La página www.parallax.com/go/Boe-Bot página también tien ligas a reglas de concursos que van de simples a complejos y muy retadores.

Algunas ideas también se incluyen aquí en el Apéndice C: Concursos en la página 299.

Control Remoto IR para el Boe-Bot

Control Remoto IR para el Boe-Bot está disponible en copia dura (#28139) y como descarga PDF gratuita. Este libro usa el mismo circuito que ha hecho en su Boe-Bot, y tiene programas ejemplo que:

- Hacen posible dirigir su Boe-Bot presionando y sosteniendo algunos botones en el control Remoto.
- Hacen que su Boe-Bot escuche comandos de configuración del control remoto que le dicen qué hacer después, como avanzar, seguir objetos, permitir el control remoto y más …

El único element adicional que requiere es un control remoto universal para TV, que es un objeto común en muchos hogares y puede comprarse a buen precio a través de muchas tiendas así como a través de www.parallax.com (#020-00001).

Mas actividades con su Boe-Bot usando el Kit que ya tiene

He aquí algunos ejemplos de actividades que puede encontrar en www.parallax.com/go/Boe-Bot que usan las partes en su kit Boe-Bot. No necesita comprar partes extra para intentar estas actividades:

- Mejor detección de distancia variando la brillantez del LED IR en vez de la frecuencia
- Navegar en un laberinto
- Detectar la llama de una vela

RESUMEN

El barrido de frequencia fue presentado como una manera de determinar la distancia usando el LED y el detector IR del Boe-Bot. **FREQOUT** fue usado para enviar señales IR a frequencias en el rango de 37.5 kHz (sensibilidad máxima) a 41.5 kHz (sensibilidad mínima). La distancia fué determinada siguiendo cuál frecuencia causó que el detector IR reportara que un objeto fue detectado y cuál no. Puesto que no todas las frecuencias fueron separadas por el mismo valor, el comando **LOOKUP** fue presentado como una forma simple de usar la secuencia de conteo entregada por un ciclo **FOR…NEXT** para indexar listas secuenciales de números.
Se presentaron sistemas de control junto con un control de ciclo cerrado. El control proporcional en un sistema de ciclo cerrado es un algorítmo donde el error es

multiplicado por una constante de proporcionalidad para determinar la salida del sistema. El error es la salida medida del sistema restada del valor de configuración. Para el Boe-Bot, ambos salida del sistema y valor de configuración se expresaron en términos de distancia. El BASIC stamp fue programado en PBASIC para operar ciclos de control para ambos servos izquierdo y derecho y los detectores de distancia. Al volver a muestrear la distancia y ajustar la salida al servo antes de enviar pulsos a los servos, el ciclo de control hizo que el Boe-Bot respondiera al movimiento de objetos. El Boe-Bot fue capaz de usar un control proporcional para fijar y seguir objetos, y también lo usó para fijar y seguir una tira de cinta eléctrica negra.

Por ultimo, pero no al menos, se cubrieron algunos indicadores hacia más actividades, recursos y concursos ya que casi se han terminado la totalidad de los temas en este punto.

Vea el Boe-Bot en acción en www.parallax.com!

Puede ver el Boe-Bot y otros video clips de robots en la sección de Videos de Robots de la Galería de Video en www.parallax.com/go/videos.

Preguntas

1. ¿Cuál sería la sensibilidad relative del detector IR si usa **FREQOUT** para enviar una señal de 35 kHz? ¿Cuál es la sensibilidad relativa con una señal de 36 kHz?
2. Considere el extracto de código que sigue. Si la variable **index** es 4, ¿qué número será colocado en la variable **prime** en este comando **LOOKUP**? ¿Qué valores guardará **prime** cuando index es 0, 1, 2, y 7?

```
LOOKUP index, [2, 3, 5, 7, 11, 13, 17, 19], prime
```

3. ¿En qué orden se evaluar las expresiones matemáticas en PBASIC? ¿Cómo puede modificar ese orden?
4. ¿Qué directive PBASIC usa para declarar una constante? ¿Cómo lo daría al número 100 el nombre “BoilingPoint?”

Ejercicios

1. Liste la sensibilidad del detector IR para cada frecuencia en kHz mostrada en la Figura 8-1.
2. Escriba un segmento de código que haga el barrido de frecuencia solo para 4 frecuencias en vez de 5.
3. Haga una lista de verificación condensada para las pruebas que deben ser ejecutadas para asegurar un seguimiento fiel a una tira.

Proyectos

1. Cree tipos diferentes de intersecciones de cinta eléctrica y programe el Boe-Bot para navegar a través de ellas. Las intersecciones pueden ser de 90° a la izquierda, 90° a la derecha, de 3 y 4 vías. Este involucrará hacer que el Boe-Bot las reconozca como intersecciones. Cuando el Boe-Bot ejecute StripeFollowingBoeBot.bs2, el Boe-Bot se quedará quieto en las intersecciones. La meta es hacer que el Boe-Bot se percate de que no está haciendo nada y salga de su ciclo de control proporcional.

   Pistas: Puede hacer esto creando dos contadores, uno con incrementos de 1 cada vez que pase a través del **`DO…LOOP`**, y el otro que solo incremente cuando el Boe-Bot entregue un pulso alo frente. Cuando el contador que incremente cada vez a través del **`DO…LOOP`** llegue a 60, use **`IF…THEN`** para revisar cuántos pulsos al frente fueron aplicados. Si se aplicaron menos de 30 pulsos al frente, el Boe-Bot probablemente esté atorado. Recuerde reiniciar ambos contadores a cero cada vez que el contador de ciclo llegue a 60. Luego de que el Boe-Bot reconozca que está en una intersección, necesita moverse a la parte superior de la intersección, luego retroceder y distinguir si ve cinta eléctrica o fondo blanco a la izquierda y derecha, luego hacer la vuelta de 90° correcta. Use un movimiento pre-programado para girar 90°, sin control proporcional. Para intersecciones de 3 y 4 vías, el Boe-Bot puede dar vuelta a izquierda o derecha.
2. Proyecto avanzado Opcional - ¡Diseñe su propio concurso para solucionar laberintos y programe el Boe-Bot para resolverlo!

Soluciones

Q1. La sensibilidad relativa a 35 kHz es 30%. Para 36 kHz, es 50%.

Q2. Cuando **index** = 4, **prime** = 11.

index = 0, **prime** = 2
index = 1, **prime** = 3
index = 2, **prime** = 5
index = 7, **prime** = 19

Q3. Las expresiones son evaluadas de izquierda a derecha. Para cambiar esto, use parentesis para cambiar el orden.

Q4. Use la directiva **CON**.

```
BoilingPoint CON 100
```

E1. Frecuencia (kHz): 34 35 36 37 38 39 40 41 42
Sensibilidad : 14% 30% 50% 76% 100% 80% 55% 35% 16%

E2. Para resolver este problema, ponga solo 4 frecuencias en la list **LOOKUP**, y decremente el índice **FOR…NEXT** en uno.

```
FOR freqSelect = 0 TO 3
  LOOKUP freqSelect, [37500, 38750, 39500, 40500], irFrequency
  FREQOUT 8, 1, irFrequency
  irDetect = IN9
  … comandos no mostrados
NEXT
```

Agregue un comando **DEBUG** al **IF...THEN**. No olvide el **ENDIF**.

```
  READ Dots + index, noteDot
  IF noteDot = 1 THEN
    noteDuration = noteDuration * 3 / 2
    DEBUG "Dotted Note!", CR
  ENDIF
```

E3. • Olfatee interferencia IR con IrInterferenceSniffer.bs2.
• Corra DisplayBothDistances.bs2.
• Lecturas de blanco deben ser 0-1 en ambos sensores.
• Lecturas de negro deben ser 4-5 en ambos sensores.
• Con la línea enmedio, ambos sensores deben leer 0-1.
• Mueva el Boe-Bot adelante y atrás sobre la línea, el sensor sobre la línea negra debe leer 4-5.

P1. En la solución que sigue, la subrutina **`Check_For_Intersection`** implementa el algoritmo trazado. El servo izquierdo fue arbitrariamente escogido para contar los pulsos al frente. Una variable de tamaño bit llamada **`isStuck`** se usa como bandera para que la Rutina Principal sepa se ha alcanzado una intersección. En la subrutina **`Navigate_Intersection`**, el Boe-Bot va al frente pasando la intersección y luego regresa, checa los sensores usando **`DO…LOOP…UNTIL`**. Luego hace una vuelta preprogramada de 90 grados en la dirección correcta. Si la intersección es de 3 o 4 vías, el Boe-Bot arbitrariamente girará en la dirección en que el negro es primero detectado. Una constante, **`Turn90Degree`**, se prove para afinar la vuelta de 90 grados. Se incluyen algunas señales audibles y visuales, que ayudan para corregir errores y entender lo que el Boe-Bot está viendo y decidiendo, asi como para agregar un pincelazo de personalidad y diversión.

```
' -----[ Titulo ]-----------------------------------------------------
' Robotica con el Boe-Bot - IntersectionsBoeBot.bs2
' Navega izq/der 90 grados, intersecciones de 3 y 4 vias.
' Basado en StripeFollowingBoeBot.bs2

' {$STAMP BS2}                    ‘ Directiva Stamp.
' {$PBASIC 2.5}                   ‘ Directiva PBASIC.
DEBUG "Program Running!"

' -----[ Constantes ]-------------------------------------------------

Kpl             CON     35          ' constante proporcional izq
Kpr             CON     -35         ' constante proporcional der
SetPoint        CON     3           ' 0-1 es Blanco, 4-5 es Negro
CenterPulse     CON     750
Turn90Degree    CON     30          ' pulsos necesarios para vuelta 90

RightLED        PIN     1           ' Indicadores LED
LeftLED         PIN     10

' -----[ Variables ]-------------------------------------------------

freqSelect      VAR     Nib         ' barrido a través de 5 frecuencias
irFrequency     VAR     Word        ' Freq enviada al emisor IR
irDetectLeft    VAR     Bit         ' Guarda resultados de detectores
irDetectRight   VAR     Bit
distanceLeft    VAR     Nib         ' Calcula zonas de distancias
distanceRight   VAR     Nib
pulseLeft       VAR     Word        ' Anchos de pulsos Servos
pulseRight      VAR     Word
numPulses       VAR     Byte        ' Cuenta pulsos totales
fwdPulses       VAR     Byte        ' Cuenta pulsos al frente
counter         VAR     Byte
isStuck         VAR     Bit         ' Variable Booleana,Boe-Bot trabado?
```

```
' -----[ Inicializacion ]-------------------------------------------

FREQOUT 4, 2000, 3000

' -----[ Rutina Principal ]------------------------------------------

DO
  GOSUB Get_Ir_Distances           ' Lee sensores IR
  GOSUB Update_LEDs                ' Indica linea blanca/negra

' Calcula salida proporcional y se mueve en consecuenca.
  pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse
  GOSUB Send_Pulse

  GOSUB Check_For_Intersection     ' ¿Atorados en la interseccion?
  IF (isStuck = 1) THEN
    GOSUB Make_Noise               ' indicacion Audible
    GOSUB Navigate_Intersection    ' Navega a través
  ENDIF

LOOP

' -----[ subrutinas ]-----------------------------------------------

Navigate_Intersection:
'  Al frente hasta que ambos sensores vean blanco en la intersection.
  DO
    pulseLeft = 850:  pulseRight = 650  ' Al frente
    GOSUB Send_Pulse
    GOSUB Get_Ir_Distances
    GOSUB Update_LEDs
  LOOP UNTIL (distanceLeft <=2) y (distanceRight <=2)

  GOSUB Stop_Quickly                    ' No pasar de largo

' Ahora retrocede hasta que un detector vea negro. Vuelta izq y der
' veran negro en 1 detector. 3 o 4 vías veran negro ambos,giraran
' hacia cualquiera que el Boe-Bot vea primero (aleatorio).
  DO
    pulseLeft = 650:  pulseRight = 850  ' Retrocede
    GOSUB Send_Pulse
    GOSUB Get_Ir_Distances
    GOSUB Update_LEDs
  LOOP UNTIL (distanceLeft >=4) OR (distanceRight >=4)

  GOSUB Stop_Quickly                    ' No pasar de largo atras

' Vuelta 90 grados en direccion del detector que ve negro
  IF (distanceLeft >=4) THEN           ' Detector izq lee negro
  FOR counter = 1 TO Turn90Degree      ' Gira 90 grados izq
```

```
    PULSOUT 13, 750                    ' sin control proporcional
    PULSOUT 12, 650
    PAUSE 20                            ' entonces usa PAUSE 20
  NEXT
  ELSEIF (distanceRight >=4) THEN       ' Detector der lee negro
   FOR counter = 1 TO Turn90Degree      ' Gira 90 grados der
    PULSOUT 13, 850
    PULSOUT 12, 750
    PAUSE 20
  NEXT
  ENDIF

' Fin. En este punto el Boe-Bot giro 90 grados para seguir la
' interseccion. Continua siguiendo la linea negra.

  RETURN

Check_For_Intersection:
' Mantiene chequeo del # de pulsos vs pulsos al frente. Si hay menos
' de 30 pulsos al frente por un total de 60 pulses, es muy probable
' que el robot esta trabado en una interseccion.

  isStuck = 0                       ' Inicializa variable Booleana
  numPulses = numPulses + 1         ' Cuenta total pulsos enviados

  SELECT numPulses
    CASE < 60
      IF (pulseLeft > CenterPulse) THEN
        fwdPulses = fwdPulses + 1    ' Cuenta pulsos al frente
      ENDIF                         '(frente es cualquier pulse > 750)

    CASE = 60                       ' Si hemos enviado 60 pulsos
      IF (fwdPulses < 30) THEN      ' ¿cuantos fueron al frente?
        isStuck = 1                 ' Si < 30, robot esta trabado
      ENDIF

    CASE > 60
      numPulses = 0                 ' Reestablece contadores a cero
      fwdPulses = 0                 ' (Puede hacerlo en caso =60 pero
  ENDSELECT                         ' no dejaria disfrutar Make_Noise)
  RETURN

Make_Noise:
' Hace un tono incremental, proporcional al numero de pulsos al frente
  FOR counter = 1 TO fwdPulses STEP 3
    FREQOUT 4, 100, 3800 + (counter * 10)
  NEXT
  RETURN
```

```
Update_LEDs:
' Usa LEDs para indicar si los detectores estan viendo blanco o negro.
' Blanco = Off, Negro = On. Negro es una lectura de distancia > o = 4.
  IF (distanceLeft >= 4)  THEN HIGH LeftLED  ELSE LOW LeftLED
  IF (distanceRight >= 4) THEN HIGH RightLED ELSE LOW RightLED
  RETURN

Stop_Quickly:
' esto detiene las ruedas para que el Boe-Bot no pase de largo.
  PULSOUT 13, 750
  PULSOUT 12, 750
  PAUSE 20
  RETURN

Get_Ir_Distances:
' Lee ambos detectores IR de objetos y calcula la distancia.
' Linea negra da lecturas de 4-5. Superficie blanca da lecturas 0-1.
  distanceLeft = 0
  distanceRight = 0
  FOR freqSelect = 0 TO 4
    LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency

    FREQOUT 8,1,irFrequency
    irDetectLeft = IN9
    distanceLeft = distanceLeft + irDetectLeft

    FREQOUT 2,1,irFrequency
    irDetectRight = IN0
    distanceRight = distanceRight + irDetectRight
  NEXT
  RETURN

Send_Pulse:
' Manda un solo pulso a los servos entre las lecturas IR.
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5                         ' PAUSE reducida por las lecturas IR
  RETURN
```

P2. Si crea un proyecto de laberinto Boe-Bot interesante y quiere compartirlo con otros, quizá quiera unirse a los foros Stamps in Class o Proyectos en http://forums.parallax.com. O puede enviar un email al Equipo de Educación Parallax directamente en education@parallax.com.

A

Apéndice A: Lista de Partes y Opciones del Kit

Para completar las actividades en este texto, necesitará un robot Boe-Bot completo y los componentes electrónicos necesarios para construir los circuitos ejemplo. Las acciones de tipo están escritas en este apéndice. Toda la información en este apéndice estaba actualizada en el momento de la impresión. Parallax hace sustituciones a nuestra discreción, por necesidad o para mejorar la calidad de nuestros productos. Para información más actualizada, descargas y accesorios visite www.parallax.com/go/Boe-Bot.

Opciones Completas del Kit Robot Boe-Bot

Fuera de una PC con un puerto serial USB y algunos pocos elementos caseros comunes, las opciones del Kit Robot Boe-Bot contiene todas las partes y documentación que necesita para completar los experimentos en este texto.

Kit Robot Boe-Bot - Serial con Adaptador USB (#28132) Partes y cantidades sujetas a cambio sin aviso previo		
Código	**Descripción**	**Cantidad**
BS2-IC	Módulo microcontrolador BASIC Stamp 2	1
28124	Kit de Partes “Robótica con el Boe-Bot “	1
28125	Guía del Estudiante “Robótica con el Boe-Bot”	1
28150	Tarjeta de Educación - Serial	1
700-00064	Desarmador Parallax	1
800-00003	Cable Serial	1
28031	Adaptador USB a Serial y Cable USB A a Mini B	1

Kit Robot Boe-Bot – Solo USB (#28832) Partes y cantidades sujetas a cambio sin aviso previo		
Código	**Descripción**	**Cantidad**
BS2-IC	Módulo microcontrolador BASIC Stamp 2	1
28124	Kit de Partes “Robótica con el Boe-Bot “	1
28125	Guía del Estudiante “Robótica con el Boe-Bot”	1
28850	Tarjeta de Educación USB	1
700-00064	Desarmador Parallax	1
805-00006	Cable USB A a Mini B	1

Kit de Partes "Robótica con el Boe-Bot"

Si ya tiene una tarjeta Board of Education y el BASIC Stamp 2, puede adquirir el kit de partes "Robótica con el Boe-Bot", con o sin este libro impreso:

Partes de Robótica con el Boe-Bot y Texto, #28154 Solo Partes de Robótica con el Boe-Bot, #28124 Partes y cantidades sujetas a cambio sin aviso previo		
Código	**Descripción**	**Cantidad**
150-01020	Resistencia de 1 kΩ	2
150-01030	Resistencia de 10 kΩ	4
150-02020	Resistencia de 2 kΩ	2
150-02210	Resistencia de 220 Ω	8
150-04710	Resistencia de 470 Ω	4
150-04720	Resistencia de 4.7 kΩ	2
200-01031	Capacitor de 0.01 µF	2
200-01040	Capacitor de 0.1 µF	2
350-00003	LED Infrarrojo	2
350-00006	LED Rojo	2
350-00029	Fototransistor	2
350-00014	Receptor Infrarrojo (Panasonic PNA4602M o equivalente)	2
350-90000	LED standoff para LED infrarrojo	2
350-90001	Guarda de luz LED para LED infrarrojo	2
400-00002	Pushbutton, normalmente abierto	2
451-00303	Conector de 3-Pines	2
700-00056	Cable Whisker	2
800-00016	Cables de conexión (bag of 10)	2
900-00001	Piezoparlante	1
	Paquete Hardware Boe-Bot	1

Contenido del Paquete de Hardware Boe-Bot

Se pueden comprar individualmente partes de reemplazo de hardware para el Boe-Bot, como podrá ver en nuestra tienda de componentes para Robot en línea. Porfavor note que

el Paquete de Hardware no se vende como una unidad separada de los Kits (completos) del Robot Boe-Bot o del Kit de Partes del Boe-Bot.

Contentido de Pack Hardware Boe-Bot Partes y cantidades sujetas a cambio sin aviso previo		
Código Parallax	**Descripción**	**Cantidad**
700-00002	Desarmador Phillips, 3/8” cabeza 4-40	8
700-00003	Tuerca Hexagonal, 4-40 zincada	10
700-00009	Tail wheel ball	1
700-00015	Rondana de Nylon, tamaño de tornillo #4	2
700-00016	Desarmador plano, 4-40 x 3/8”	2
700-00022	Chasis de alumio del Boe-Bot	1
700-00023	Cotter pin, 1/16" x 1.5” long	1
700-00025	Tapa de goma para barreno pasado, 13/32"	2
700-00028	Desarmador Phillips, cabeza 4-40 x 1/ 4”	8
700-00038	Sostenedor de batería con cable y barrel plug	1
700-00060	Standoff, de aluminio threaded, round 4-40	4
710-00007	Desarmador Phillips, 7/8” cabeza 4-40	2
713-00007	Espaciador 1/2” , aluminio, #4 redondo	2
721-00001	Rueda de plastico Parallax	2
721-00002	Rueda tipo liga de hule	4
900-00008	Servo de Rotación Continua Parallax	2

Construyendo un Boe-Bot con una tarjeta BASIC Stamp HomeWork

La tarjeta HomeWork, incluída en el Kit de Actividades BASIC Stamp (#90005) puede ser usada con el kit de Partes “Robótica con el Boe-Bot” y estos artículos adicionales:

(2) conectores de 3-pines macho/macho, #451-00303
(1) Paquete batería metal-plomo, #753-00001

Una nota para los Educadores: Hay disponibles por cantidad para todas los kits antes listados; vea la página de producto de cada kit en www.parallax.com para detalles. Además, la tarjeta BASIC Stamp HomeWork Board está disponible por separado en paquetes de 10 como una solución económica para el uso en clase, a un costo

significativamente menor que el módulo Board of Education + BASIC Stamp 2 (#28158). Contacte al equipo de ventas de Parallax sin costo al (888) 512-1024.

Apéndice B: Códigos de Color de Resistencia y Reglas de tableta

B

CODIGOS DE COLOR DE RESISTENCIA

Las resistencias que usamos en esta guía tienen tiras coloreadas que dicen el valor de resistencia que tiene. Hay una diferente combinación de color para cada valor de resistencia.

Puede haber una cuarta tira que indica la tolerancia de la resistencia. La tolerancia se mide en por ciento, indica qué tan lejos está el valor real de la resistencia respecto al valor etiquetado. Esta tira puede ser dorada (5%), plateada (10%) o sin tira (20%). Para las actividades en este libro, la tolerancia de una resistencia no es relevante, no así su valor.

Cada barra de color corresponde a un dígito y estos colores/dígitos están enlistados en la tabla. La Figura B-1 le muestra cómo usar cada barra de color para determinar el valor de cada resistencia.

Dígito	Color
0	Black
1	Brown
2	Red
3	Orange
4	Yellow
5	Green
6	Blue
7	Violet
8	Gray
9	White

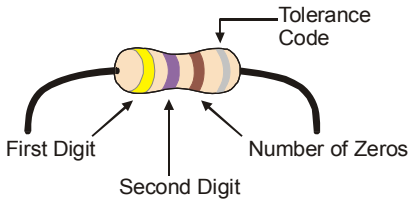


Figura B-1
Códigos de Color de Resistencia

- La primera tira es amarilla, el dígito más a la izquierda es un 4.
- La segunda tira es violeta, el segundo dígito es un 7.
- La tercera tira es café o uno. Quiere decir que se agrega un cero a la derecha de los primeros dos dígitos.

El valor de esta Resistencia es 470 Ω.

REGLAS PARA EL USO DE TABLETAS

Observe la tarjeta de su Board of Education o HomeWork Board. El cuadro blanco con muchos hoyos o sockets se conoce como tableta. Esta tableta, combinada con las tiras negras de sockets a lo largo de dos de sus lados, se llama área de prototipos (mostrada en la Figura B-2).

Los circuitos ejemplo en este texto son construídos conectando componentes como resistencias, LEDs, bocinas y sensores en estos pequeños sockets. Los componentes son conectados uno al otro con los sockets de la tableta. Suministrará energía a su circuito con electricidad desde las terminales de energía que están en los sockes negros a lo largo de la parte superior, marcados Vdd, Vin, y Vss. Los sockets negros a la izquierda están marcados P0, P1, hasta P15. Estos sockets le permiten conectar su circuito a los pines de entrada/salida del BASIC Stamp.

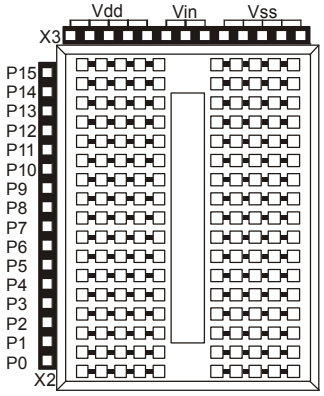


Figura B-2
Area de Prototipos

Las terminales de energía (sockets negros arriba), pines de acceso E/S (sockets negros laterales), y tableta (sockets blancos).

La tableta tiene 17 renglones de sockets separados en dos columnas por un separador. El separador divide cada uno de los 17 renglones en dos renglones de cinco. Cada renglón de cinco sockets está eléctricamente conectado dentro de la tableta. Puede usar este renglón de sockets para conectar componentes según lo indica el esquemático de un circuito. Si inserta 2 cables en 2 sockets cualesquiera en el mismo renglón de 5 sockets, están eléctricamente conectados al otro.

El esquemático de un circuito es un mapa que muestra cómo conectar los componentes. Usa símbolos únicos que representan componentes diferentes. Estos símbolos de componentes están conectados por líneas que indican una conexión eléctrica. Cuando dos símbolos de circuitos están conectados por líneas en un esqsuemático, la línea indica que

se hace una conexión eléctrica. Las líneas también pueden usarse para conectar los componentes a fuentes de tensión. Vdd, Vin, y Vss tienen todos símbolos. Vss corresponde a la terminal negative de la fuente o batería para la tarjeta Board of Education o tarjeta BASIC Stamp HomeWork Board. Vin es la terminal positive de la bacteria y Vdd está regulada a +5 volts.

Veamos un ejemplo que usa un esquemático para conectar las partes mostras en la Figura B-3. Para cada una de estas partes se muestra el dibujo de la parte arriba del símbolo esquemático.

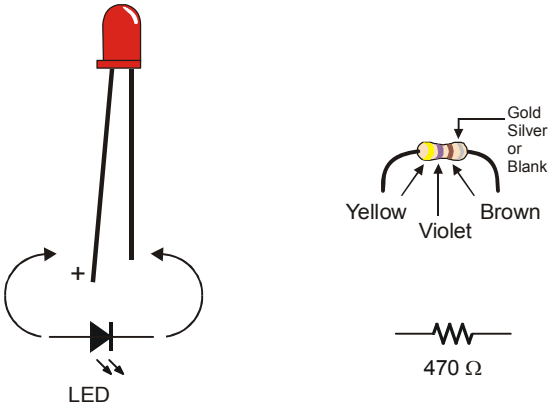


Figura B-3
Dibujos de Parte y Símbolo esquemático

LED(izquierda) y resistencia de 470 Ω (right)

La Figura B-4 muestra un ejemplo del esquemático de un circuito a la izquierda y a la derecha el dibujo del circuito que puede ser construido usando este esquemático. Note como el esquemático muestra que una terminal de la línea de sierra que denota una resistencia está conectado al símbolo para Vdd. En el dibujo, una de las dos terminales de la resistencia está conectada en uno de los sockets etiquetados Vdd. En el esquemático, la otra terminal del símbolo de resistencia está conectado por una línea a la terminal + del símbolo LED. Recuerde, la línea indica que 2 partes están eléctricamente contectadas. En el dibujo, esto se consigue conectando la otra parte de la resistencia dentro de los sockets de la misma línea de 5 en donde se también se conecta la terminal + del LED. Esto conecta eléctricamente las dos terminales. La otra terminal de LED se muestra conectada al símbolo Vss en el esquemático. En el dibujo, la otra terminal del LED está conectada dentro de los socketa marcados Vss.

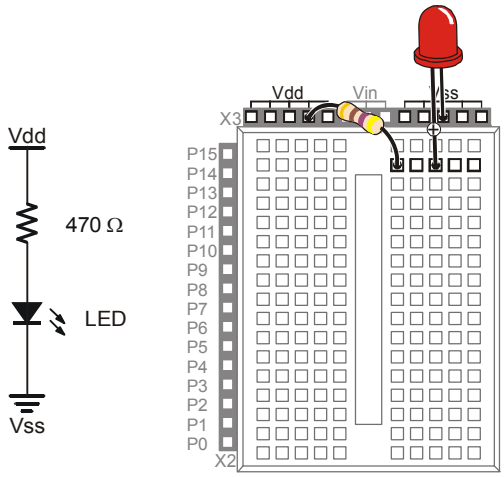


Figura B-4
Example Schematic y Wiring Diagram

Schematic (left) y Diagrama de Conexiones (right)

La Figura B-5 muestra un segundo ejemplo de un esquemático y un diagrama de cableado. Aquí, P14 está conectado a un extremo de una resistencia con su otro extreme conectado a la terminal + de un LED y la terminal – del LED está conectada a Vss. Estos esquemáticos difieren solo por 1 conexión. La punta de la resistencia antes conectada a Vdd ahora está conectada al pin 14 de E/S del BASIC Stamp. El esquemático puede verse más distinto que eso porque la Resistencia se muestra dibujada horizontalmente en vez de vercialmente, pero en términos de conexiones, solo difiere en P14 en lugar de Vdd.

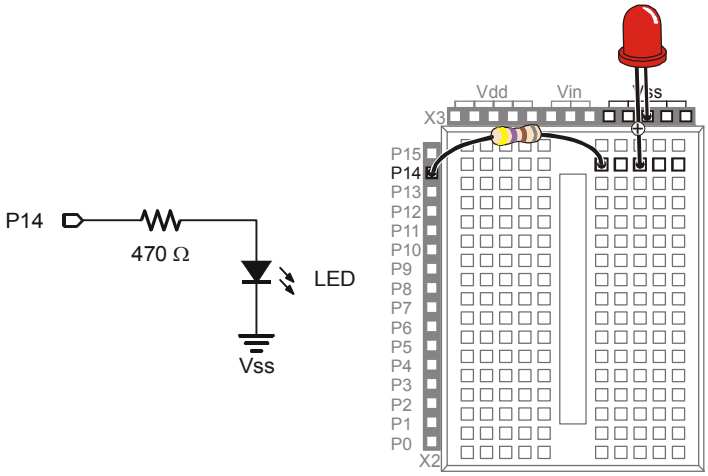


Figura B-5
Esquemático de ejemplo y diagram de conexiones

Esquemático (izquierda) y diagrama de cableado (derecha)

He aquí un ejemplo más complejo que involucra dos partes adicionales, una resistencia de 1 kΩ, un fototransistor y un capacitor. Los símbolos esquemáticos y dibujos de los componentes con los que no está familiarizado se muestran en la Figura B-6. Las terminales de fotoransistor están etiquetadas C, B, y E. La terminal B es óptica y por eso no tiene conexiones eléctricas. La terminal C es el pin más largo y la terminal E es el pin más corto que salen del encapsulado plástico por la cara plana en uno de sus lados.

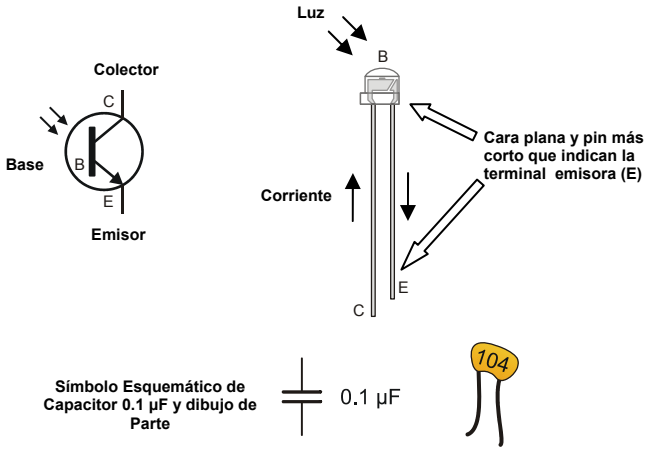


Figura B-6
Dibujos de Parte y Símbolos esquemáticos

Fototransistor (arriba) capacitor No-polar (abajo)

Puesto que el esquemático mostrado en la Figura B-7 indica una Resistencia de 1 kΩ, o sea 1000 Ω, el primer paso es consultar el Apéndice C: Códigos de color de resistencias para determinar el código de color. El código es Café, Negro, Rojo. Esta resistencia está conectada a P6 en el esquemático, que corresponde a la Resistencia conectada en el socket etiquetado P6 en el área de prototipos (Figura B-8). En el esquemático, la otra punta de la Resistencia está conectada no a uno sino a 2 terminales de components: la terminal C del fototransistor y una de las terminals del capacitor. En la tableta, la punta de la resistencia esta conectada en una de las filas de 5 sockets de la tableta. Esta fila también tiene la punta C del fototransistor y una de las puntas del capacitor conectadas allí. En el esquemático, la terminal E del fototransistor y y la otra punta del capacitor están conectadas a Vss. He aquí un truco que recordad al construer circuitos en una tablet: puede usar un cable para conectar una fila completa de la tableta a otra, o incluso a pins E/S o terminales de energía como Vdd o Vss. En este caso, se usó un cable para conectar Vss a una fila de la tableta. Asi, la punta E del fototransistor y la otra punta del capacitor están conectadas en la misma fila, lo que las conecta a Vss, completando el circuito.

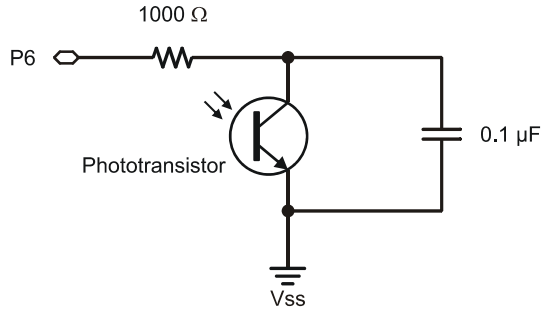


Figura B-7
Esquemáticos de Resistencia, fototransistor y Capacitor

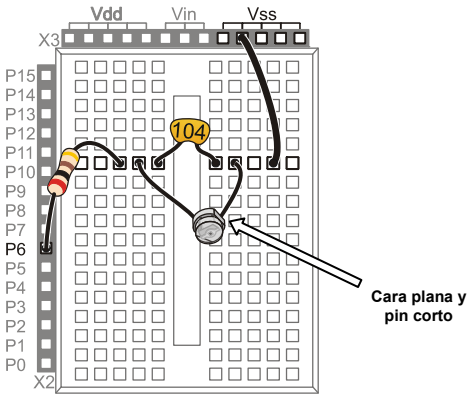


Figura B-8
Diagrama de cableado de Resistencia, fototransistor, y Capacitor

Tenga en mente que los diagramas de conexiones aquí presentados como Soluciones a los esquemáticos no son las Soluciones UNICAS de estos esquemáticos. Por ejemplo, la Figura B-9 muestra otra solución al esquemático recién discutido. Siga las conexiones y convénzase de que satisface el esquemático.

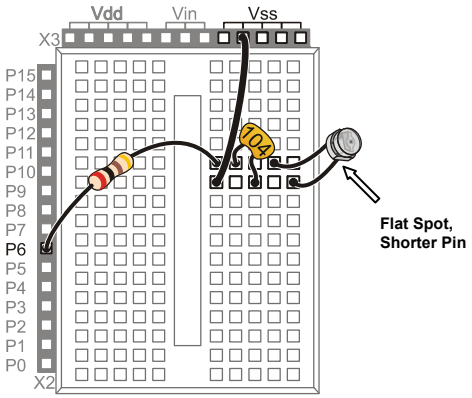


Figura B-9
Diagrama de cableado de Resistencia, fototransistor y Capacitor

Note la colocación alternativa de las partes.

Apéndice C: Concursos de Navegación Boe-Bot

Si está planeando una competencia de robots autónomos, estas reglas son provistas como cortesía de Seattle Robotics Society.

CONCURSO #1: EJERCICIO DE PISO PARA EL ROBOT

Propósito

El propósito de la competencia de ejercicio de piso es dar a los inventores de un robot la oportunidad de mostrar sus robots u otros aparatos técnicos.

Reglas

Las reglas de esta competencia son muy simples. Se identifica un área plana de 10 pies por 10 pies, preferentemente con algunos límites físicos. A cada concursante le serán dados un máximo de 5 minutos en esta área para demostrar lo que puede hacer su robot. Como siempre, cualquier robot que pueda dañar el área o presentar un daño al público no será permitido. Los robos no necesitan ser autónomos, pero se recomienda ampliamente. La votación es determinada por la audiencia, ya sea aplaudiendo (el aplauso más alto determinado por un jurado), o algún mecanismo de votación.

CONCURSO #2: SEGUIR UNA LINEA

Objetivo

Construir un robot autónomo que inicie en el área "A" (en la posición "S"), viaja al área "B" (completamente por una línea), luego viaja al área "C" (completamente por la línea), luego regresa al área "A" (a la posición "F"). El robot que hagá esto en la menor cantidad de tiempo (incluyendo bonos) gana. El robot debe entrar a las áreas "B" y "C" para calificar. El curso exacto del viaje no se conocerá hasta el día de la competencia, pero tendrá las 3 a previamente descritas.

Habilidades probadas

La habilidad de reconocer una ayuda navegacional (la línea) y usarla para llegar a una meta.

Tiempo Máximo para Completar el Curso

Cuatro minutos.

Curso Ejemplo

Todas las mediciones en el curso de ejemplo son aproximadas. Hay una línea sólida dividiendo el área “A” del área “T” en la posición “F”. Esto indica dónde termina el curso. La línea es negra, aproximadamente de 2.25 pulgadas de ancho y espaciada aproximadamente a 2 pies de las paredes. Todas las curvas tienen un radio de al menos un pie y a lo más tres pies. Las paredes son de 3 ½ pulgadas de alto y rodean el curso. El piso es blanco y está hecho ya sea de papel o Tyvek® de Dupont. Tyvek es un fuerte plastic usado en sobres de correo y construcción de casas.

Las posiciones son meramente ilustrativas y no son precisamente ubicaciones. Un competidor puede colocar el robot en cualquier lugar en el área “A”, de frente en cualquier dirección cuando se inicie. El robot debe estar completamente dentro del área “A”. Las áreas "A," "B" y "C" no están coloreadas en rojo en curso real.

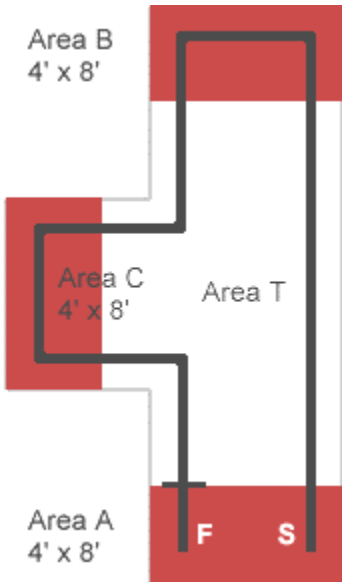


Figura D-1
Curso ejemplo de competición

Puntaje

El marcador de cada concursante es calculado tomando el tiempo necesario para completar el curso (en segundos) menos el 10% por cada “cumplimiento”. El concursante con el menor puntaje ganara.

Puntaje de segumiento de línea	
Cumplimiento	**Porcentaje deducido**
Detenerse en el area A después de llegar a B y C	10%
Sin tocar ninguna pared	10%
Comando de inicio	10%

("Comando de inicio" significa que el robot inicia por un comando no táctil externo, éste podría ser, por ejemplo, un sonido o un comando de luz.)

CONCURSO #3: SIGUIENDO EL LABERINTO

Propósito

El gran laberinto pretende presentar una prueba de habilidades navegacionales para un robot autónomo. La puntuación es hecha de tal forma que favorece al robot que sea brutalmente rápido o que pueda apreender el laberinto después de una pasada. El objetivo para un robot que es colocado a la entrada del laberinto es encontrar su camino a través del laberinto y llegar a la salida en la menor cantidad de tiempo.

Características físicas

El laberinto está construído de madera triplay de ¾". Las paredes son de aproximadamente 24 pulgadas de alto y están pintadas en colores primarios brillantes. Las paredes están dispuestas en una cuadrícula con espaciamiento de 24 pulgadas. Debido al espesor de la madera y las limitaciones en la precisión, los corredores pueden ser tan estrechos como 22 pulgadas. El laberinto puede ser de hasta 20 pies cuadrados, pero puede ser menor, dependiendo del espacio disponible para el evento.

El laberinto será puesto ya sea sobre alfombra de tipo industrial o sobre el piso sólido (dependiendo de donde se lleve a cabo el evento). El laberinto estará cubierto y su robot no tendrá que ser a prueba de lluvia; sin embargo, puede ser expuesto a diferentes temperaturas, viento y condiciones de luz. El laberinto tiene una forma clásica de 2 dimensiones: hay una ruta desde el inicio hasta el final y no hay islas en el mismo. A la entrada como la salida están localizadas sobre paredes exteriores. Los laberintos pueden ser solucionados siguiendo ya sea la pared izquierda o la pared derecha. El laberinto es cuidadosamente diseñado para que no haya ventajas y se sigue la pared derecha o la pared izquierda.

Limitaciones del Robot

La mayor limitación para el robot es que sea autónomo: una vez encendido por el propietario o manejador, no se permite interacción hasta que salga por la puerta de salida, o quede irremediablemente atrapado. Obviamente el robot necesita ser lo suficientemente pequeño para entrar en las paredes del laberinto. Puede tocarlas pero no puede moverlas para su ventaja - sin destrucción. Los jueces pueden descalificar a un robo que a su juicio muevan las paredes excesivamente. No debe dañar las paredes del laberinto ni él piso. Cualquier forma de energía está permitida mientras que las leyes locales no requieran protección auditiva en su presencia o colocar alguna otra limitante.

Puntuación

Cada robot debe hacer 3 carreras por el laberinto. El robot con el menor tiempo es el ganador. El mayor tiempo permitido por carrera es 10 minutos. Si un robot no puede terminar en ese tiempo, la carrera se detiene y el robot recibe un tiempo de 10 minutos. Sin ningún robot encuentra la salida del laberinto, el que haya llegado más lejos será el ganador, determinado así por el jurado.

Logística

Cada robot hará una carrera, prosediendo hasta que todos hayan intentado cruzar el laberinto. Cada robot entonces hace una segunda carrera a través del laberinto y luego los robot hacen todos la tercera carrera. El jurado decidirá a discreción si un contendiente deber retrasar su carrera debido a dificultades técnicas. Un robot puede recordar lo que encontró en una carrera previa para tratar de mejorar su tiempo (mapear el laberinto de la carrera) y puede usar esta información mientras que el robot lo haga por sí mismo. No se permite configurar manualmente al robot a través de hardware o softwer para mapear el laberinto.

Las partes y sus cantidades son sujetas a cambio sin aviso previo. Las partes pueden diferir de com se muestran en esta imagen. Si tiene preguntas respecto a su kit, envíe por favor un email a stampsinclass@parallax.com.