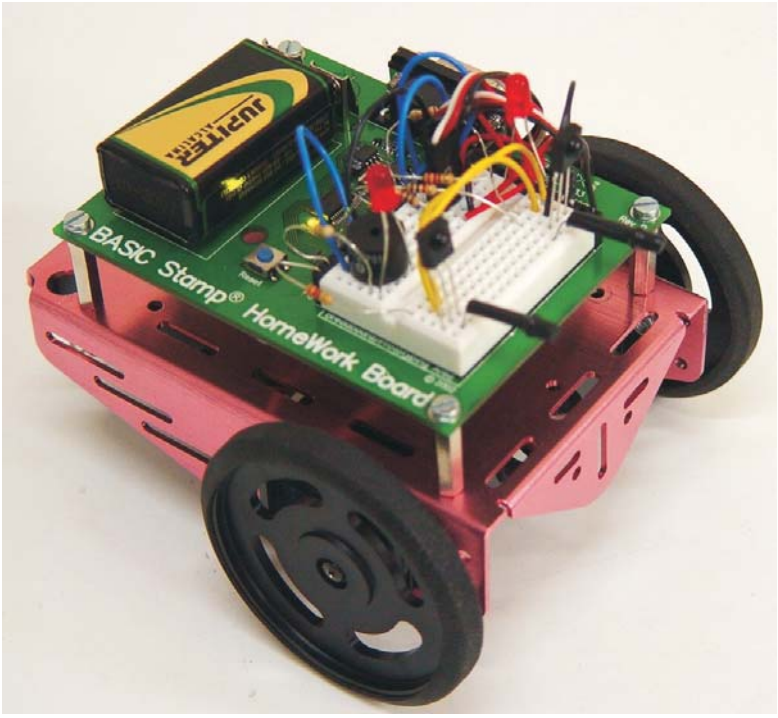
Microbot
Home Boe-Bot

MANUAL DE MONTAJE Y PROGRAMACION

Versión 3.1 (2005)



INGENIERIA DE MICROSISTEMAS PROGRAMADOS S.L.

MSE MICROSYSTEMS ENGINEERING

C/ Alda. Mazarredo Nº 47 - 1º Dpto. 2
48009 BILBAO - BIZKAIA
Tel/Fax: 94 4230651

Email: info@microcontroladores.com
www.microcontroladores.com

MANUAL DE MONTAJE Y PROGRAMACIÓN DEL ROBOT "HOME BOE-BOT"

PRÓLOGO

Te presentamos la herramienta más valiosa y agradable para introducirse en la realidad tecnológica del siglo XXI. La época que nos está tocando vivir está marcada por la rápida evolución de todas las áreas relacionadas con los computadores. La razón es muy sencilla: se ha logrado construir un computador completo en un circuito integrado que ocupa lo mismo que una moneda y vale poco más de 1 euro. Con este invento que se llama "microcontrolador" se ha mejorado extraordinariamente las prestaciones y el funcionamiento de todo tipo de aparatos, desde los más comunes, como la televisión, el teléfono y el automóvil, hasta los más complejos, como los empleados en la navegación espacial y la medicina.

Otra de las aplicaciones asombrosas de los microcontroladores son los robots. Y en este caso no nos referimos a los que se dedican a realizar labores complejas y peligrosas en la Industria, si no a los pequeños robots móviles que desarrollan sencillas y conocidas tareas y que suponen una gran ayuda para todos los seres humanos. Estas bestiecillas mecánicas las denominamos "microbots" y ya existen en el mundo muchas que se encargan de limpiar la casa, preparar la comida, cortar el césped, vigilar, explorar nuevos planetas y acompañar y entretener a sus propietarios. Realizan labores simples, pero muy necesarias y, a veces, muy tediosas. La invasión de los microbots acaba de comenzar y los próximos años serán espectadores de su implantación masiva por todos los confines de la tierra.

Las nuevas tecnologías no son difíciles de comprender ni están reservadas a personas con grandes conocimientos científicos. Pueden y deben ser conocidas y aplicadas por todos. Sólo se necesitan buenos maestros y herramientas adecuadas que sean fáciles de manejar, profesionales y accesibles para todos.

Ingeniería de Microsistemas Programados S.L., con la indispensable colaboración de Parallax, presenta la herramienta más fabulosa para aprender a diseñar aplicaciones con microcontroladores y microbots. Es válida para "todos" y no se requiere conocimientos específicos de Electrónica ni de Informática. ¡Sólo requiere muchas ganas y entusiasmo por aprender!. Se trata del microbot **Home Boe-Bot.**

El presente tutorial está basado en los manuales y productos de Parallax y se ha redactado en Ingeniería de Microsistemas Programados S.L. para personas que comienzan desde el principio. Sólo hay que saber usar el PC y tener las nociones fundamentales de electricidad y programación. ¡Lo imprescindible es el interés y la curiosidad!. Eso lo tienes que poner tú y si no lo tienes te recomendamos que abandones ahora.

Te proponemos la aventura científica más apasionante que vas a disfrutar en tu vida. Vas a construir una máquina que hará todo lo que tú le mandes y nada más que eso. Es tan dócil y obediente que te encariñarás con ella y sufrirás cuando se equivoque. Va a ser tu imagen más leal en el mundo y comenzarás a sentir su corazoncito, por eso la bautizarás, la cuidarás, la enseñarás a realizar muchas tareas interesantes, pasarás muy buenos ratos con tu bestiecilla y, cuando lleguéis a intimar, ampliarás sus sensores, añadirás nuevos actuadores y juntos proyectareis ambiciosas metas. ¡Quizás, hasta te animes un día a presentarla en un certamen para demostrar a todos de lo que los dos sois capaces!. El próximo lo tienes a mediados de Diciembre en la Universidad de Deusto (Bilbao) y puedes obtener más información en http://www.eside.deusto.es/asignaturas/arq.

Esta mascota se convertirá en el mejor maestro y amigo que has tenido y te introducirá con entusiasmo en el diseño de las aplicaciones de la Microelectrónica programada del futuro para hacer de ti un gran profesional.

A lo largo del manual hemos intentado atraer tu atención y simplificar tu aprendizaje, pero si tienes problemas no estás solo. Tenemos un foro en nuestra dirección en Internet (www.microcontroladores.com)donde acuden un montón de enamorados del microbot que siempre están disponibles para echar una mano, compartir sus proyectos y descubrimientos y organizar reuniones de encuentro y certámenes. Otros foros de "piraos" los puedes encontrar en www.parallax.com en el que existe uno en castellano dirigido por el consultor Arístides Alvarez al que agradecemos su valiosísima colaboración y sus siempre acertados consejos.

Prólogo

Esmérate desde el principio. Sé generoso de tu tiempo y sabiduría en la creación de esta criaturita de silicio, metal y plástico. Luego tus programas le insuflarán el alma y estarás orgulloso de ser su creador. ¡Nunca te arrepentirás de participar en esta maravillosa aventura!.

NOTA IMPORTANTE

Una versión de este Manual a todo color y con más claridad en el texto, las figuras y las fotos, junto a una información más amplia puede encontrarla en el CD que acompaña al Kit y en www.microcontroladores.com

MANUAL DE MONTAJE Y PROGRAMACIÓN DEL ROBOT "HOME BOE-BOT"

ÍNDICE

Anexo 4: Relación de materiales del kit del Home Boe-Bot

BIBLIOGRAFÍA

"MICROCONTROLADORES PIC. Diseño práctico de aplicaciones" (contiene CD)
Autor: Angulo J.Mª y Angulo I., Editorial: Mc Graw-Hill

"MICROCONTROLADORES PIC. Diseño práctico de aplicaciones" (2ª parte)
Autor: Angulo J.Mª, Romero S. y Angulo I., Editorial: Mc Graw-Hill

"MICROCONTROLADORES PIC. La clave del diseño"
Autor: Angulo J.Mª y Angulo I., Editorial: ITES Paraninfol

"LABORATORIO DE PRACTICAS DE MICROELECTRÓNICA"
Autor: Angulo J.Mª, Editorial: Mc Graw-Hill

"LABORATORIO DE PRACTICAS DE MICROELECTRÓNICA" Volumen 2
Autor: Angulo J.Mª, Editorial: Mc Graw-Hill

"MICROBOTICA. Tecnología, Aplicaciones y Montaje Práctico"
Autor: Angulo J.Mª, Romero S. y Angulo I., Editorial: ITES Paraninfo

"DISEÑO PRACTICO CON MICROCONTROLADORES PARA TODOS"
Autor: Angulo J.Mª, Romero S. y Angulo I., Editorial: ITES Paraninfo

"CURSO PRACTICO DE DISEÑO CON PIC" , 1ª Parte
Autor: Ingeniería de Microsistemas Programados S.L.

"CURSO PRACTICO DE DISEÑO CON PIC" , 2ª Parte
Autor: Ingeniería de Microsistemas Programados S.L.

DIRECCIONES DE INTERÉS

www.microcontroladores.com
info@microcontroladores.com
www.parallax.com
www.microchip.com

Tema 1

Microcontroladores y Microbots. La invasión tecnológica del siglo XXI

1.1 ¿QUÉ SON LOS MICROCONTROLADORES?

Los microcontroladores son pequeños circuitos integrados en cuyo interior se ha construido un computador completo. Esto significa que en un pequeño espacio y con un coste muy reducido se dispone de un potente computador. Incluir un microcontrolador dentro de cualquier producto no representa ninguna dificultad ni incremento apreciable del precio pero proporciona una gran potencia y enormes posibilidades que suponen un cambio de imagen y unas enormes expectativas de mercado

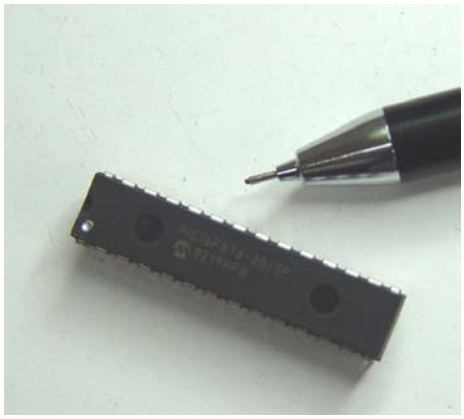


Figura 1-1.- *Un microcontrolador es un circuito integrado en cuyo interior hay un computador.*

1.2 ¿DÓNDE ESTÁN LOS MICROCONTROLADORES?

En la actualidad en un hogar medio de un país avanzado existen más de un centenar de aparatos que contienen uno o varios microcontroladores. El horno microondas, el lavaplatos, el TV, el vídeo, el teléfono, el aire acondicionado, el PC y todos sus periféricos, el sistema de alarma, el ascensor y el automóvil son ejemplos elementos comunes que disponen microcontroladores para su gobierno.

Figura 1-2.- *Fotografía de un teléfono móvil que incluye varios microcontroladores.*

Si el hogar está invadido por los microcontroladores "embebidos", en los lugares de trabajo son indispensables. Tanto las labores de gestión, como las de diseño, producción, verificación, almacenaje y comercialización exigen múltiples sistemas gobernados por microcontroladores.

Figura 1-3.- *Fotografía de un robot industrial dotado de varios microcontroladores.*

1.3 ¿CÓMO SE DISEÑAN APLICACIONES CON MICROCONTROLADORES?

Para desarrollar una aplicación práctica basada en un microcontrolador hay que hacer dos cosas. Conectar a las correspondientes patitas del microcontrolador los dispositivos de Entrada, que recogen información del mundo exterior y, los de Salida, que realizan las acciones precisas. En segundo lugar hay que confeccionar el programa basado en las instrucciones que admite el microcontrolador, que sirve para organizar y planificar las operaciones a realizar a lo largo del tiempo. Una vez puesto a punto el programa de la aplicación se graba en la memoria que posee el microcontrolador y el sistema queda operativo para siempre.

En resumidas cuentas el diseño exige manejar "hardware", que se refiere al conexionado físico de los elementos que intervienen en la tarea con las patitas del microcontrolador, y desarrollar el "software", consistente en la creación del programa de instrucciones que planifique las acciones a ejecutar en función de las características de las entradas.

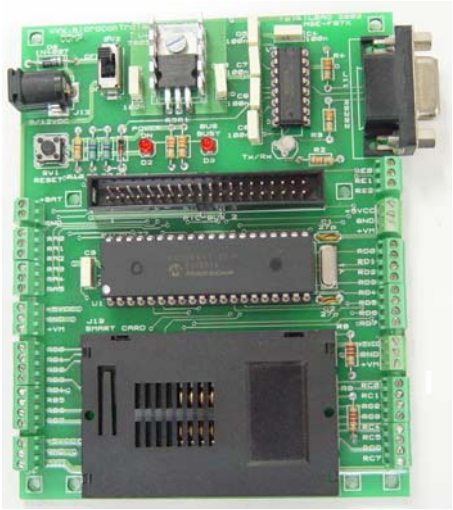


Figura 1-4.- *Una tarjeta con un microcontrolador programado para realizar una tarea.*

1.4 LA GRAN IDEA DE PARALLAX

Para simplificar el diseño de las aplicaciones con microcontroladores dos jóvenes estudiantes americanos desarrollaron unos módulos que resolvían la mayor parte del conexionado de los componentes físicos de la aplicación. Además, esos módulos podían programarse con el lenguaje más fácil del mundo, que se llamó PBASIC y era similar al que usamos los humanos para comunicarnos, sólo que empleando el idioma inglés. Para la explotación comercial en todo el mundo de esa gran idea crearon la empresa Parallax, cuyo distribuidor para España es Ingeniería de Microsistemas Programados S.L. (www.microcontroladores.com) y que es la que se ha encargado de confeccionar este Manual en base a los manuales de Parallax para ayudar a sus clientes españoles e hispanoamericanos a construir y programar el fantástico Robot Home Boe-Bot, que es el equipo que se incluye con el presente informe. Una información complementaria se puede encontrar en www.parallax.com.

Para diseñar proyectos reales controlados por microcontrolador utilizando los módulos BASIC Stamp de Parallax no hace falta tener grandes conocimientos de Electrónica ni de Informática. En muy pocas horas una persona normal puede ser capaz de poner en marcha experiencias de cierta complejidad.

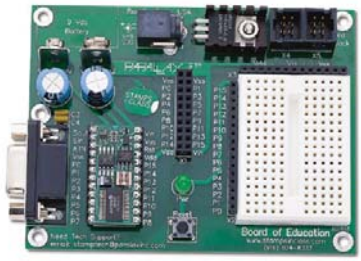


Figura 1-5.- *Fotografía de un típico módulo BASIC Stamp de Parallax.*

1.5 ¿QUIÉNES PUEDEN Y DEBEN USAR LOS MÓDULOS DE PARALLAX?

Los módulos microcontroladores de Parallax han sido creados para poderlos utilizar todo el mundo. Son extraordinariamente interesantes para los muchachos de 14 a 17 años que cursan la Enseñanza Secundaria y la Formación Profesional. Suponen una gran ayuda a los que inician los estudios universitarios de carreras técnicas de cualquier tipo de Ingeniería. Volverá locos de alegría a los aficionados a la Electrónica y al montaje y diseño de proyectos. Para los que saben o se acaban de iniciar en la programación, esta aventura les va a permitir comprobar cómo se transforman los deseos expresados por instrucciones en órdenes, así como algo inmaterial como es un programa, se convierte en algo real que se comprueba en cada instante. Finalmente, los técnicos especialistas en cualquier materia que tengan interés por automatizar y mejorar las prestaciones de su área de trabajo encontrarán en la tecnología que proponemos la mejor herramienta para llevar a cabo un auténtico progreso.

No hay excusas. Esta tecnología es accesible a cualquiera y cuánto antes se inicie en ella mejor preparado estará para los retos del futuro. Cualquier persona que desee materializar sus ideas tiene a su disposición la herramienta más potente del mundo, el computador, y sólo se requiere que ponga de su parte la imaginación, la lógica y la inteligencia de su cerebro.

Figura 1-6.- *El campo de aplicación de los microcontroladores sólo está limitado por la imaginación del diseñador.*

1.6 LA TARJETA "HOME WORK" DE PARALLAX Y SU BIBLIA

El mejor procedimiento para aprender a diseñar sistemas basados en microcontroladores es realizar proyectos reales. Sólo se necesita un pequeño gasto y alguna ayuda inicial que afiance la seguridad del principiante. Para dicho fin Parallax ha creado la tarjeta Home Work, sobre la cual se puede realizar multitud de experiencias y existen manuales y libros que facilitan su manejo.

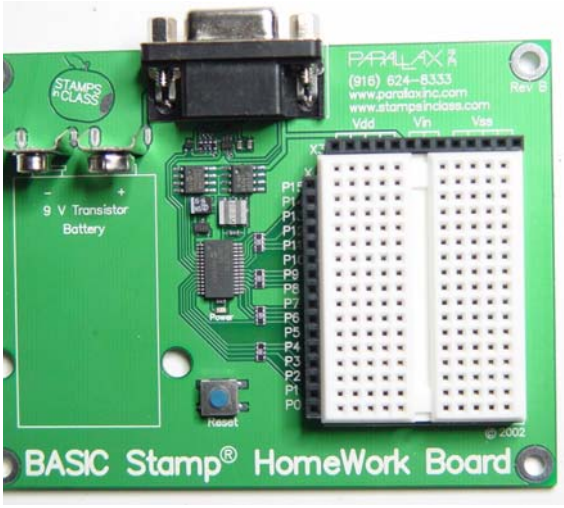


Figura 1-7.- *Fotografía de la tarjeta Home Work.*

Como complemento didáctico a la tarjeta Home Work la editorial Thomson Paraninfo S.A. ha editado un magnífico libro titulado *"DISEÑO PRÁCTICO CON MICROCONTROLADORES. Los sellos mágicos de Parallax"*, que además de presentar con mucha claridad todos los conceptos para todos, ofrece la posibilidad de ir construyendo y programando una colección de experiencias prácticas muy sugerentes y de complejidad progresiva usando un económico kit de materiales y la Home Work.

Figura 1-8.- *Fotografía de la portada del libro que puede considerarse la biblia para aprender a diseñar prácticamente aplicaciones con microcontroladores sin apenas conocimientos previos.*

Otra forma de aprender a trabajar con microcontroladores es la construcción del microbot Home Boe-Bot que está basado en la tarjeta Home Work siguiendo el programa de trabajo que le proponemos en este manual que acompaña al kit (www.microcontroladores.com). El Boe-Bot, así le llamaremos abreviadamente, será para Vd. algo más que una máquina porque seguramente atrapará su corazón al notar que se le parece, piensa y actúa como Vd. proporcionándole intensos momentos de felicidad y también de tristeza cuando no alcance los objetivos. Una recomendación que le asegura una sólida formación es conseguir con el programa de prácticas del libro mencionado y luego construir el robot que sigue utilizando la tarjeta Home Work y no necesitará volverla a comprar.

1.7 ¿QUÉ ES UN MICROBOT?

Es un pequeño robot móvil y programable que realiza una sencilla tarea. Su cerebro consiste en un microcontrolador que gobierna todas sus acciones y según el trabajo encomendado precisa de un programa concreto y de unas entradas de información y salidas para los actuadores.

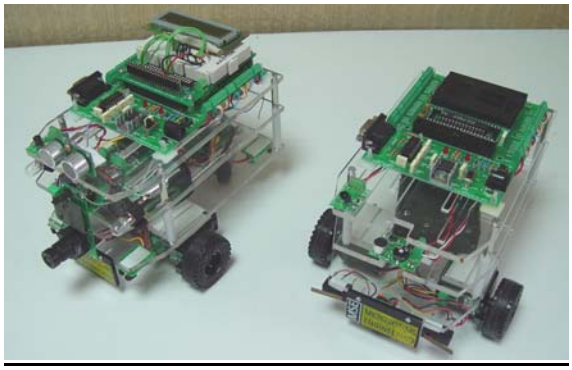


Figura 1-9.- *Fotografía del microbot PICBOT-3 comercializado por Ingeniería de Microsistemas Programados S.L..*

Un microbot es una aplicación basada en microcontrolador. Tiene dos grandes cualidades: es bastante económico y su propietario le puede destinar a realizar cualquier tarea.

1.8 ¿DÓNDE ESTÁN LOS MICROBOTS?

Los microbots se extienden por todo el mundo como una plaga que se multiplica y alcanza todas las áreas de actuación del ser humano. En pocos años será habitual que cada persona disponga de varias de estas "bestiecillas" para que le ayuden, le acompañen y le distraigan. La mayoría realizan labores muy simples, pero enormemente frecuentes tales como limpiar, cocinar, planchar, lavar, vigilar, controlar la calefacción, ayudar a personas discapacitadas y entretener, de tal forma que intentarán conseguir ocupar el puesto de "mascota preferida".

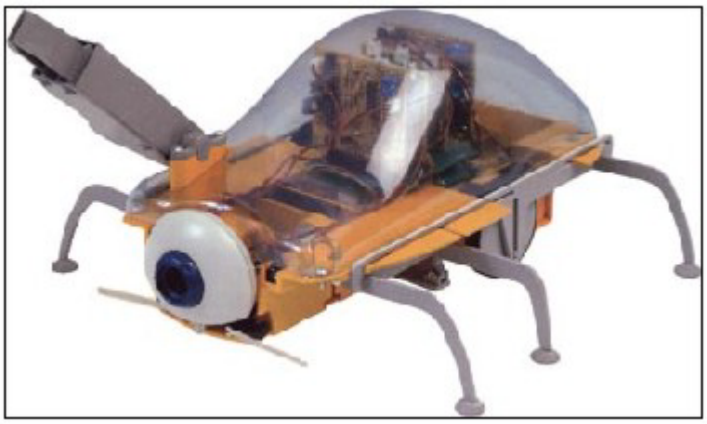


Figura 1-10.- *El microbot Pathfinder intenta ocupar el puesto de mascota de la casa.*

1.9 ¿PARA QUE SIRVEN LOS MICROBOTS?

Los microbots pueden servir para "todo". Sus aplicaciones son ilimitadas y sus restricciones vienen impuestas por la imaginación y la capacitación técnica del diseñador.

¿Quién no conoce los microbots que exploran Marte?. Pero hay otros colegas que llevan a cabo misiones similares en el fondo de los océanos y en los cráteres de algunos volcanes. Existen minúsculos microbots que recorren nuestras venas para encontrar estrechamientos y solucionarlos. Otros simulan el comportamiento de insectos y seres invertebrados y ayudan en el conocimiento y protección de especies vivas.

Los microbots más deseados y populares son los que ayudan en las tareas cotidianas del hogar. ¡Qué alegría al llegar a casa y encontrarla ordenada y limpia, con la comida preparada y nuestra bestiecilla que nos recibe con una de nuestras canciones favoritas, nos trae las zapatillas, guarda los zapatos y nos acompaña al salón para que, una vez acomodados, elija el programa de TV deseado y raudo y veloz vaya a traernos un vaso de vino fresco con un plato de aceitunas rellenas!. ¡Qué tranquilidad cuando nuestro amigo se hace cargo de la vigilancia de la casa y en sus ratos libres limpia las habitaciones, prepara la comida y el tiempo que le queda se dedica a cortar el césped del jardín!.

Figura 1-11.- *Un microbot corta-césped.*

1.10 ¿CÓMO SE CONSTRUYE Y PROGRAMA UN MICROBOT?

Un microbot consta de los siguientes elementos fundamentales:

1.- Una estructura que sujeta todos los componentes necesarios
2.- Unos motores que giran las ruedas motrices
3.- Sensores que recogen la información necesaria del entorno (temperatura, luz, presencia de obstáculo, etc.).
4.- Actuadores que realizan las acciones del microbot (altavoz, motor, LED, display, zumbador, etc.).
5.- Tarjeta de control con un microcontrolador que tiene grabado en su memoria el programa de instrucciones que gobierna el comportamiento de toda la máquina.

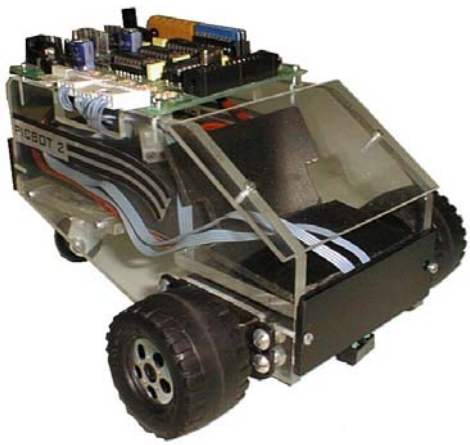


Figura 1-12.- *En la fotografía del microbot PICBOT-2 se aprecian los componentes fundamentales.*

1.11 EL MICROBOT HOME BOE-BOT: TU GRAN AVENTURA

Vas a comenzar a crear una bestiecilla mecánica en cuyo cerebro depositarás tu semilla y es muy probable que te aporte más satisfacciones que todos los elementos de ocio que te rodean. Será lo que tu quieras, pero sólo será capaz de reproducir tus ideas. Al final será tu retrato. Si pones ilusión en la empresa este amasijo de metal y silicio te acabará robando el corazón y pasará a ser tu mascota preferida. Pero, ¡no te engañes!. Él te devolverá todo lo que tú le des, ¡ni un ápice más!. Así que te interesa ser generoso. Le podrás enseñar a traerte cosas, a iluminarte el camino, a limpiar, a vigilar tu casa, pero nunca hará nada que tú no se lo enseñes.

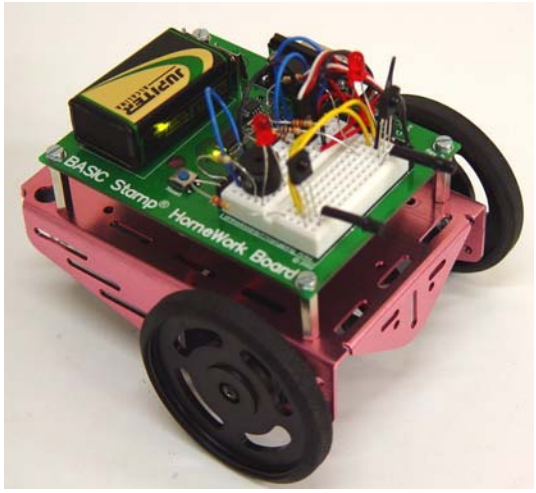


Figura 1-13.- *Fotografía del microbot Home Boe-Bot con algunos sensores conectados para realizar una tarea.*

Parallax ha intentado con sus manuales eliminar las dificultades de esta aventura y nosotros, desde Ingeniería de Microsistemas Programados S.L., hemos querido con este tutorial entusiasmarte con la idea porque si lo conseguimos sabemos que hemos formado un gran profesional que nunca olvidará estos primeros pasos. Te ofrecemos nuestra ayuda como cliente desde Internet, desde el teléfono o en nuestras oficinas, personalmente. Gracias por confiar en nosotros y ahora te toca a ti poner tu granito de arena. Con constancia, paciencia y entusiasmo recorrerás el camino. Todo lo demás te lo dará con creces esta bestiecilla encantadora que se llama Home Boe-Bot.

1.12 EL FUTURO ES PARA TI

¡No te equivoques!. El Home Boe-Bot no es un juguete o un pasatiempo. "Es la herramienta más potente y agradable que existe para hacerte un especialista en el diseño de aplicaciones con microcontroladores y la programación de tareas para microbots, que son las dos tecnologías que tienen más expectativas de crecimiento en el siglo XXI".

El mundo que viene estará invadido por productos gobernados por microcontrolador y por microbots que desarrollarán la mayoría de las tareas habituales y automáticas que realizamos los humanos en la actualidad.

Ese nuevo mundo necesita millones de especialistas que diseñen y programen microcontroladores y microbots. Esos profesionales deben tener una gran ilusión, mucha imaginación y perseverancia en la consecución de sus metas.

Te deseamos que además de disfrutar aprendiendo con esta maravillosa aventura de crear tu bestiecilla los conocimientos que adquieras te sirvan para que llegues a ser un profesional orgulloso de tu trabajo y tu colaboración a construir un mundo mejor.

Tema 2

1ª Parte: Comunicando la Home Work con el PC

2ª Parte: Aprendiendo a programar con el PBASIC

Tema 2:
Primera Parte: Comunicando la Home Work con el PC. Los primeros programas

2.1. HARDWARE Y SOFTWARE DEL HOME BOE-BOT

La mayor parte de los productos modernos disponen de uno o varios computadores para controlar su funcionamiento. El horno microondas, el teléfono móvil y el automóvil son ejemplos muy claros en los que en su interior existen cada vez más computadores enanos, llamados microcontroladores, que gobiernan todas las tareas que realizan. Este tipo de elementos constan de dos partes principales:

1ª. *El hardware*, que hace referencia a todos los dispositivos materiales y piezas físicas que les configuran. La estructura mecánica, los componentes electrónicos, los cables, los motores, etc., son los componentes que constituyen el hardware.
2ª. *El software,* que es algo invisible e inmaterial. Algo que no se ve, pero que es el "alma" del sistema. Se refiere a los programas que se hallan grabados en la memoria de los computadores y que sirven para indicarles lo que tienen que hacer en todo momento, según las condiciones existentes.

El robot que vamos a construir y programar para la realización de múltiples tareas se llama Home Boe-Bot y su cerebro es un microcontrolador, modelo PIC16C57, en cuya memoria se graba el programa que gobierna todo el sistema. En la confección de programas se utilizará el lenguaje PBASIC que es similar al que usamos los humanos para comunicarnos, pero en inglés.

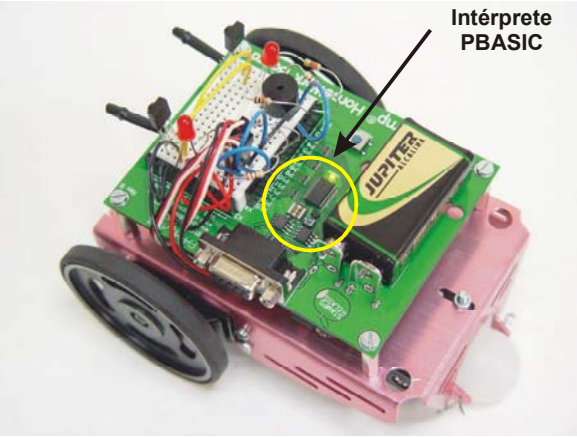


Figura 2-1.- *El cerebro del Home Boe-Bot es el microcontrolador que contiene el intérprete PBASIC*

Con los programas se conseguirá que el robot realice cuatro tareas fundamentales:

1.- Detectar, visualizar y registrar todo lo que ocurre en el entorno del robot utilizando los sensores apropiados.
2.- Tomar decisiones según la información suministrada por los sensores.
3.- Controlar el movimiento del robot mediante los dos motores que incorpora.
4.- Intercambiar información con el usuario (ese eres tú).

En este tema vas a tener que poner en marcha la tarjeta Home Work que es la que controla al robot y en la que se aloja su cerebro: el microcontrolador. Dicha tarjeta se comunica con un cable serie con el PC en donde se confeccionan los programas en PBASIC para luego descargarlos en la memoria de la tarjeta. Para realizar la comunicación entre el PC y la Home Work hay que empezar instalando un software que Parallax pone libremente a disposición de los usuarios en su sitio Web (www.parallax.com) y nosotros, desde Ingeniería de Microsistemas Programados S.L., te lo incluimos en el CD que acompaña al kit de materiales.

Se irán presentando ordenadamente los siguientes pasos a desarrollar:

- Localizar e instalar en el PC el software de comunicación con la tarjeta Home Work.

- Conectar la Home Work a la pila de alimentación.
- Conectar la Home Work al ordenador para su programación.
- Escribir los primeros programas y conocer un poco el lenguaje PBASIC.
- Desconectar las baterías una vez finalizados los trabajos.

Si Vd. pone de su parte interés y entusiasmo nuestras orientaciones y la calidad del material que maneja harán todo lo demás. Pero si le surgen pegas nosotros en Ingeniería de Microsistemas Programados S.L. haremos lo que podamos para resolver las dificultades y en último caso contamos con la colaboración de Arístides Alvarez, asesor de Parallax, que es uno de los expertos más destacados del mundo y además se comunica en español.

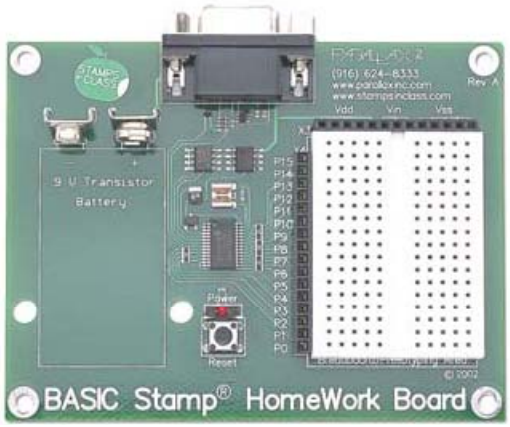


Figura 2-2.- *Fotografía de la tarjeta Home Work en la que reside el microcontrolador que será el cerebro del Boe-Bot*

2.2. PASO 1º: OBTENCION DEL SOFTWARE

El software que vamos a utilizar para realizar todas las pruebas se denomina "BASIC Stamp Editor" (Versión 2.0 ó superior). Con este software se podrán confeccionar en el PC los programas en PBASIC y la posterior descarga del programa en la Home Work del robot. También se pueden mostrar mensajes en la pantalla del PC sobre la tarea que el Home Boe-Bot está realizando en un cierto momento y sobre lo que detecta mediante los sensores.

Requerimientos mínimos del PC:

- Sistema operativo WINDOWS 95 o superior.
- Puerto serie o USB.
- Unidad CD-ROM y/o conexión a Internet para disponer del software "Basic Stamp Editor".

La descarga del software de Internet

Si no tiene a mano el CD que acompaña a nuestro kit de materiales o desea la última versión, es muy sencillo descargarse el Basic Stamp Editor desde el sitio Web de Parallax. Las pantallas que se muestran pueden diferir algo sobre las correspondientes a la última versión que puede encontrar en la Web, pero los pasos para la obtención son muy similares en todos los casos:

- Utilizando un navegador Web, dirigirse a www.parallax.com
- Pinchar dentro del menú en "Donwloads" (figura 2-3).

- Ir al enlace BASIC Stamp Software y hacer clic sobre él.
- Una vez llegamos a la página del software de BASIC Stamp, buscar el Basic Stamp Windows Editor con una versión 2.0 o superior (figura 2-4).
- Finalmente el asistente de instalación nos preguntará donde queremos depositar el Editor en el disco duro, para lo que se recomienda crear una carpeta independiente para el Home Boe-Bot.

Figura 2-3.- *Página principal del sitio Internet de Parallax.*

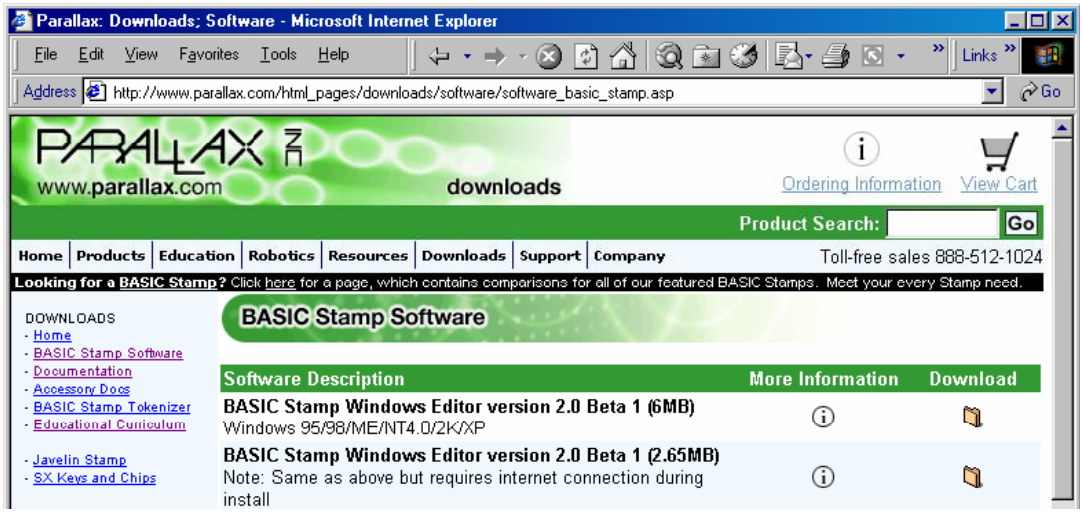


Figura 2-4.- *Pulsar el icono "BASIC Stamps Windows Editor Version 2.0 Beta (6 MB).*

.

Dentro del sitio Web de Parallax, se puede encontrar diverso material que suele ser muy útil, no olvides visitarlo a menudo.

En el caso de tener el CD-ROM de Parallax, buscar la referencia del programa mencionado y de forma similar, instalarlo en el disco duro (Aparecerá un menú, con lo que los pasos a dar son muy sencillos e intuitivos).

2.3. PASO 2º: INSTALACIÓN DEL SOFTWARE

Para instalar el Editor basta con seguir los pasos que va indicando el asistente de instalación que se presentan en las figuras 2-5 a 2-8.

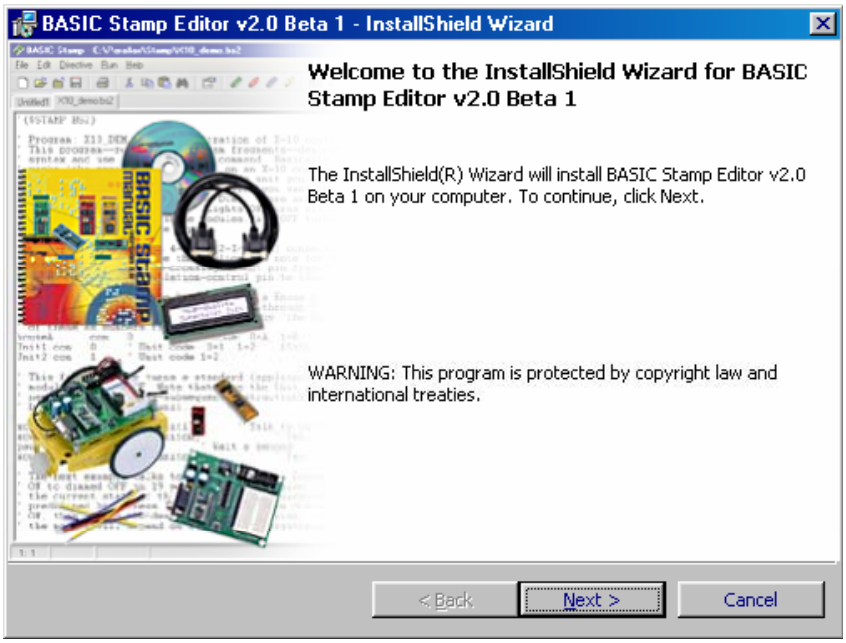


Figura 2-5.- *Ventana inicial de instalación del Editor.*

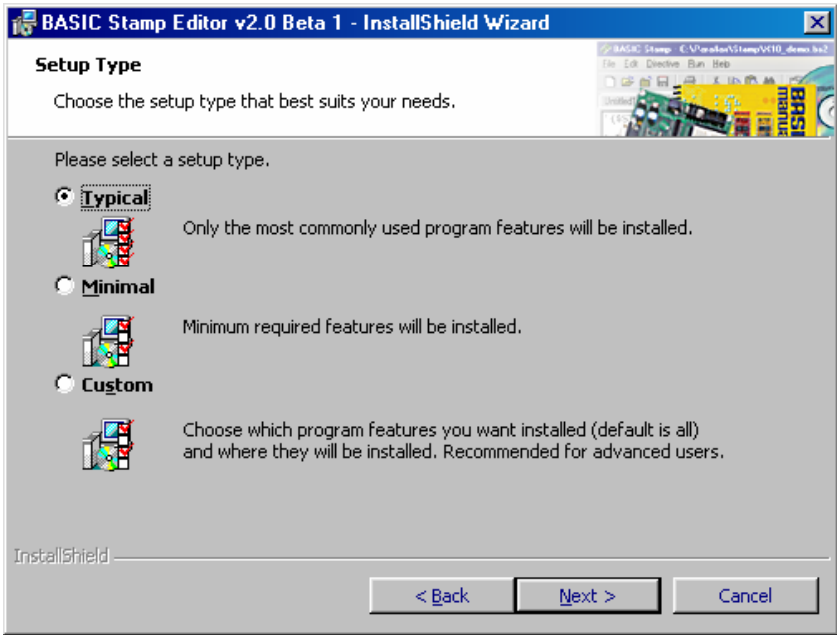


Figura 2-6.- *Segunda ventana de instalación del Editor.*

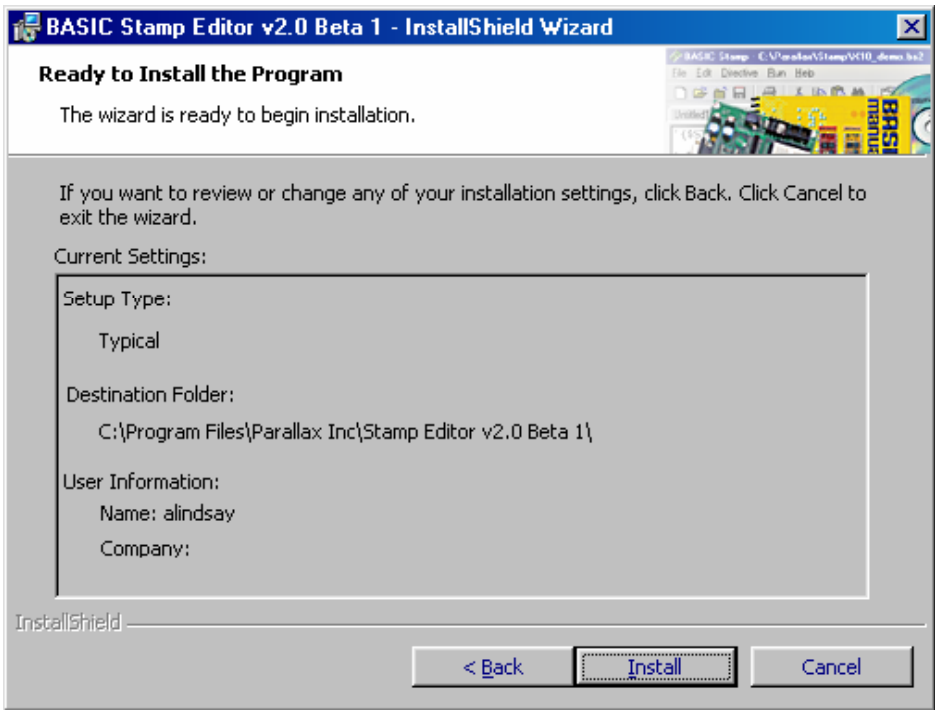


Figura 2-7.- *Tercera ventana del asistente de instalación del Editor.*

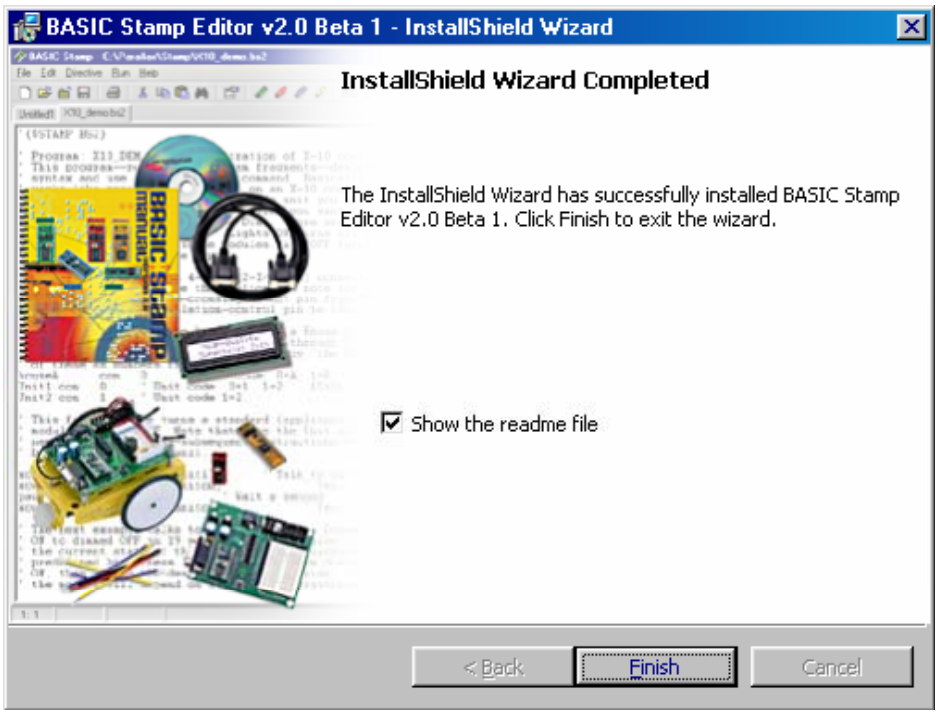


Figura 2-8.- *Cuarta y última ventana y operación que se precisa para la instalación del Editor en el PC.*

2.4. PASO 3º: CONFIGURAR Y PROBAR LA HOME WORK

Para que funcione la tarjeta Home Work hay que suministrarla energía eléctrica y comunicarla con el PC.

Conexión con el cable SERIE

La conexión del PC a la Home Work puede realizarse mediante un cable serie, que debe ser conectado tanto al conector serie de nuestro PC (Figura 2-9) como al de la Home Work. En caso de utilizar un conector USB del PC existe un adaptador USB BAFO y hay que seguir las instrucciones que vienen en el propio adaptador, que se ofrece como opción.

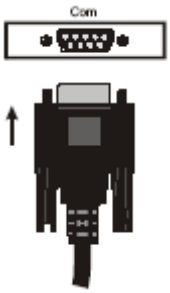


Figura 2-9.-*Uno de los extremos del cable serie se enchufa al conector serie del PC.*

Conexión con el adaptador BAFO USB a serie

Cuando se desea conectar la tarjeta Home Work a un puerto USB del PC, Parallax recomienda el adaptador USB-a canal serie, BAFO BF-810 USB (Parallax Stock# 800-00030), que viene acompañado de Manual y CD con driver para Windows. Además se puede encontrar las especificaciones y el software para este elemento en http://www.bafo.com

Figura 2-10.- *Adaptador USB a serie BAFO BF-810 USB*

Puesta en marcha de la Home Work

Se precisan los siguientes materiales.

- (1) Tira de cuatro gomas para actuar como "patas" de la placa Home Work.
- (1) Batería o pila de 9V.
- (1) Tarjeta Home Work.

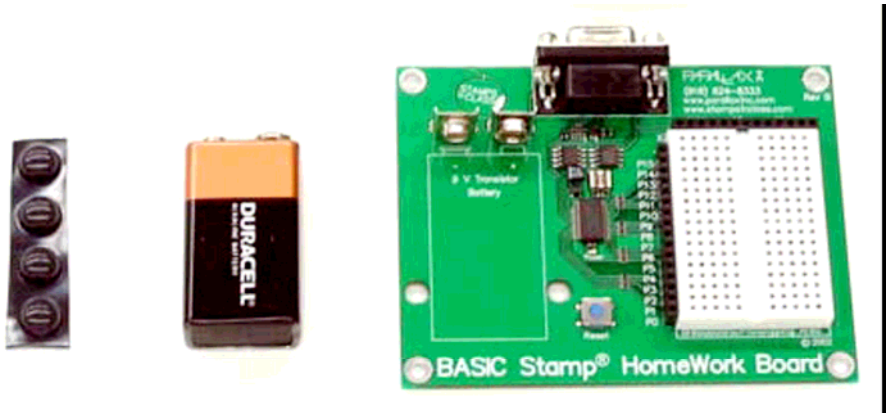


Figura 2-11.- *Gomas, batería y tarjeta Home Work.*

Poner las cuatro gomas en la parte inferior de la placa en cada esquina de la Home Work. Como se muestra en la figura 2-12.

Figura 2-12.- *Las gomas se colocan en las cuatro esquinas de la superficie interior de la Home Work.*

Conectar el cable serie y la pila a la Home Work, Figura 2-13 (pasos 1 y 2).

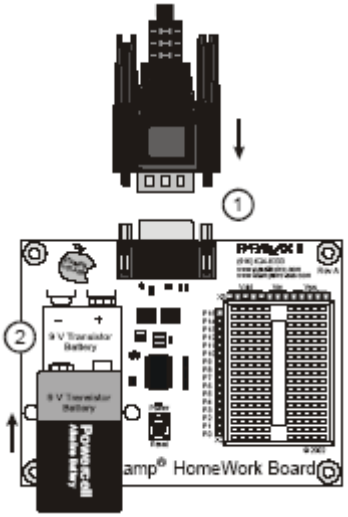


Figura 2-13.- *Un terminal del cable serie se conecta a la Home Work y el otro al puerto serie del PC. También se precisa una pila de 9 VDC.*

Hay que tener en cuenta que el diodo luminiscente (LED) verde de la Home Work no se enciende cuando conectamos la batería, sino cuando hay un programa en ejecución.

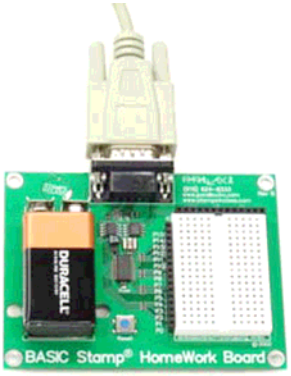


Figura 2-14.- *Aspecto de la tarjeta Home Work lista para funcionar.*

Comprobar la comunicación del PC y la Home Work

Para realizar una rápida comprobación de la correcta comunicación entre el PC y la Home Work se comienza ejecutando el programa Editor "picando" en el icono que aparecerá en la pantalla del PC una vez instalado dicho programa. Figura 2-15.

Figura 2-15.- *Icono del programa de Editor.*

La pantalla que aparece al ejecutarse el programa Editor es similar a la de la figura 2-16.

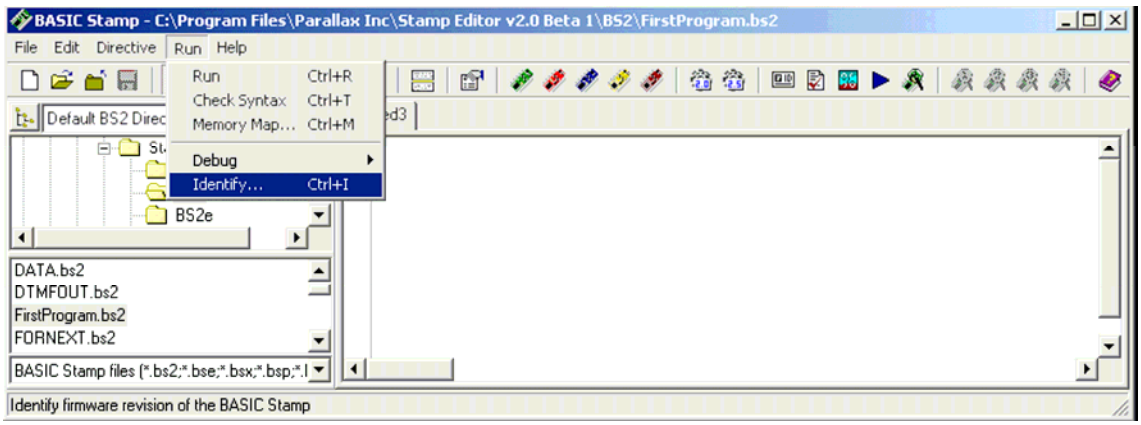


Figura 2-16.- *Pantalla principal del Editor.*

Para comprobar que la Home Work está conectada a tu PC, hacemos clic en Run y después en Identify. La ventana de identificación es similar a la que se muestra en la figura 2-17. En la figura se visualiza que está conectada al puerto COM2 y correctamente detectada.

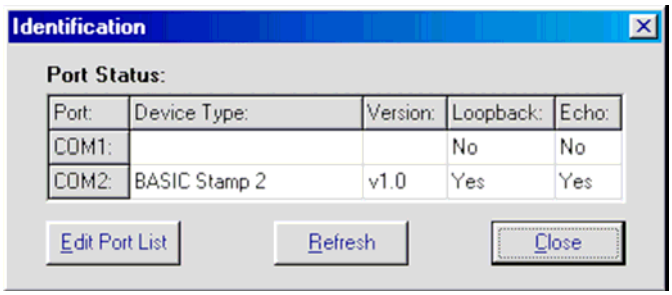


Figura 2-17.- *Identificación desde el PC de la tarjeta Home Work mediante el comando Identify.*

2.5. PASO 5º: TU PRIMER PROGRAMA

El primer programa que vamos a realizar (escribir y probar) va a consistir en transmitir un mensaje de texto desde la Home Work al PC para visualizarlo en su pantalla. En la figura 2-18 se muestra de forma gráfica como la Home Work envía una cadena de ceros y unos para que después el ordenador la interprete como caracteres de texto. El Editor BASIC Stamp instalado en el PC es capaz de detectar, interpretar y visualizar estos mensajes.

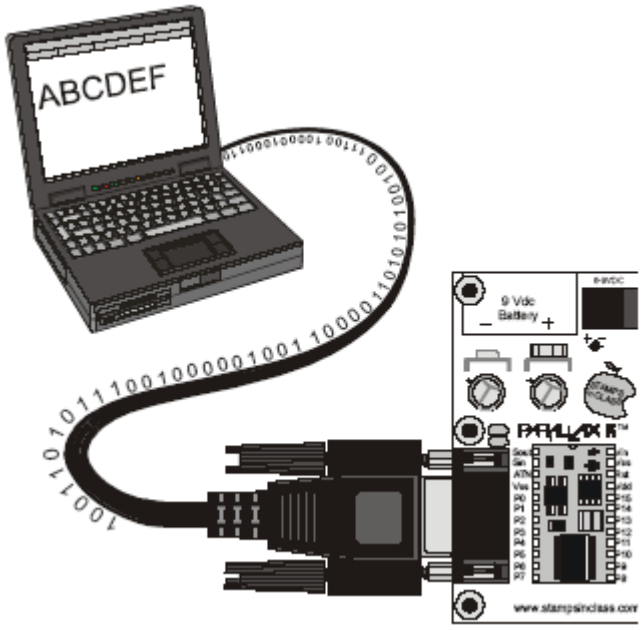


Figura 2-18.- *El Editor instalado en el PC es capaz de detectar y visualizar los mensajes de 1 y 0 que envía la Home Work por el cable serie.*

Los programas ejemplo que vas a escribir o teclear en el Editor BASIC Stamp y descargar a la tarjeta de Parallax los tienes almacenados en el CD del kit del Home Boe-Bot y te puede evitar teclearlos, aunque inicialmente te recomendamos que te acostumbres a editarlos totalmente.

Programa: HelloBoeBot.bs2

```
' El Robot Home Boe-Bot - HelloBoeBot.bs2
' El BASIC Stamp envía un mensaje de texto al PC.
' {$STAMP BS2}
' {$PBASIC 2.5}
DEBUG "Hola, esto es un mensaje desde el Home Boe-Bot."
END
```

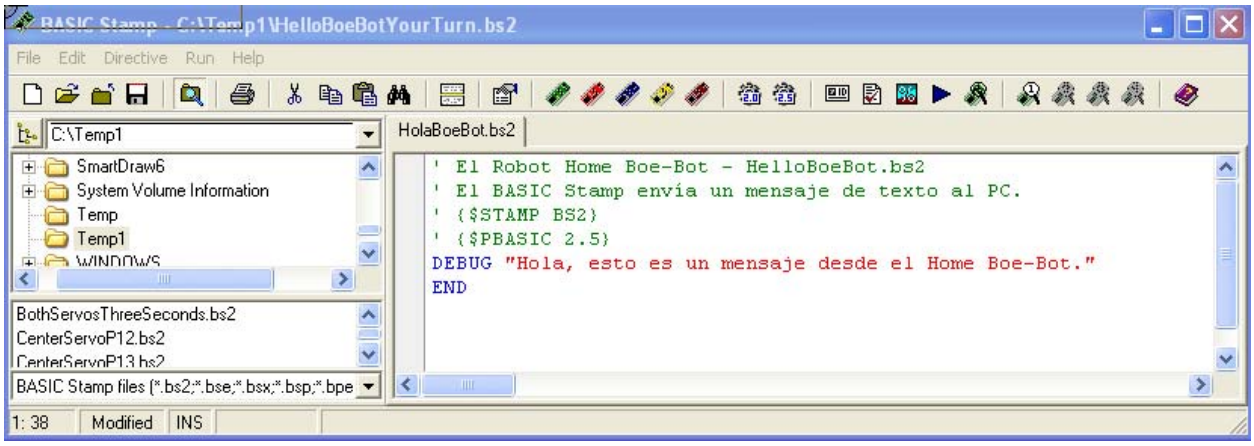


Figura 2-19.- *Teclea en la ventana del Editor del PC, el programa mostrado sin olvidar ningún signo de puntuación.*

Guarda el proyecto haciendo clic en la línea File | Save (Figura 2-20)

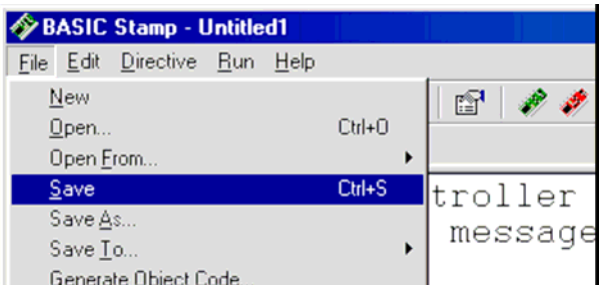


Figura 2-20.- *Comando para salvar el programa escrito en la pantalla.*

Introduce el nombre de HelloBoeBot.bs2 dentro del campo File Name y pulsa Save (Figura 2-21).

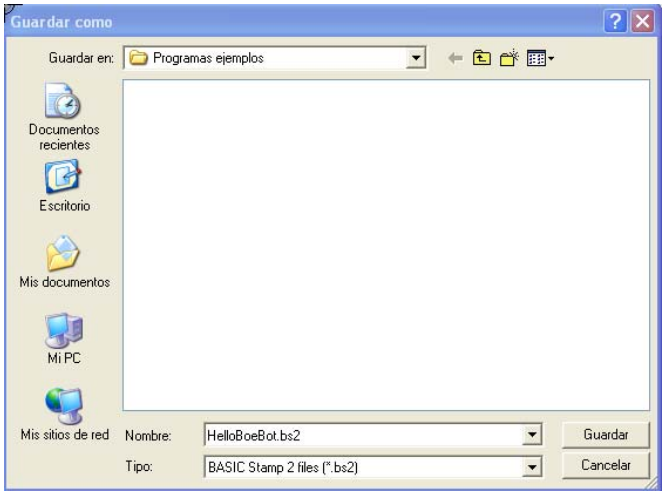


Figura 2-21.- *Se introduce el nombre del programa (HelloBoeBot.bs2) en el campo File name.*

Finalmente ejecuta el programa activando el comando Run como se muestra en la figura 2-22.

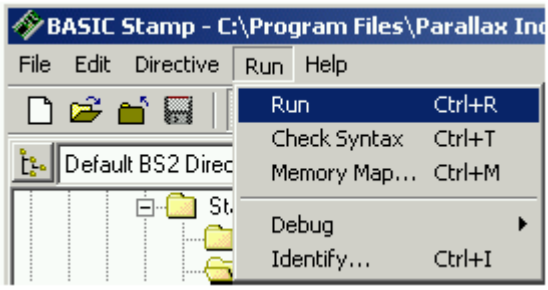


Figura 2-22.- *El comando Run ejecuta el programa confeccionado.*

Tras la ejecución del programa aparecerá una ventana de transferencia y justo después deberá aparecer la ventana de debug (Debug Terminal #1) en la que se visualiza el mensaje mandado desde la Home Work. (Figura 2-23).

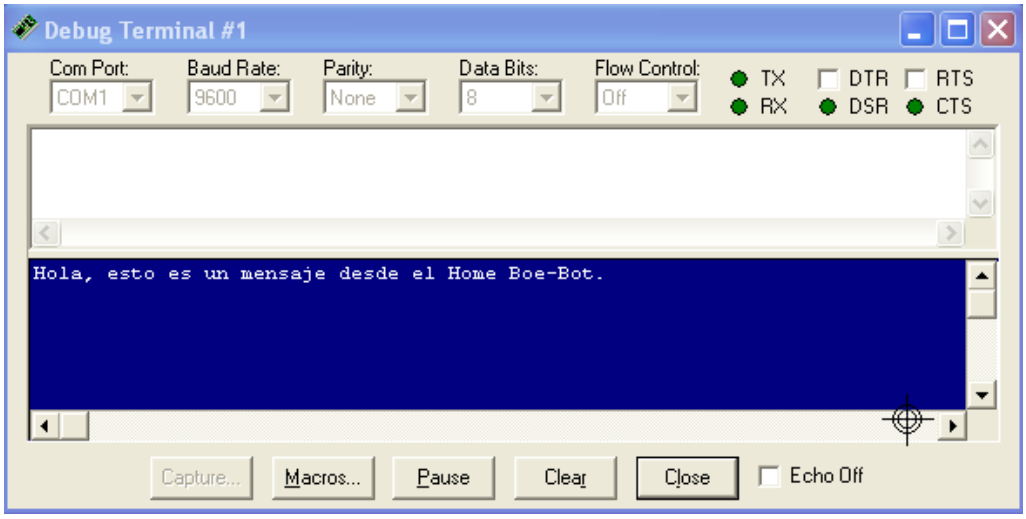


Figura 2-23.- *La ejecución del programa HelloBoeBot.bs2 origina el mensaje mostrado en la ventana Debug Terminal #1.*

Hemos editado nuestro primer programa, lo hemos ejecutado, y si todo ha ido bien, desde la Home Work se ha enviado un mensaje para visualizar en la pantalla del PC. Esto nos asegura el correcto funcionamiento de la Home Work y la comunicación con el PC y su software.

Explicación del programa HelloBoeBot.bs2

Las dos primeras líneas son comentarios. Los comentarios no son interpretados por el BASIC Stamp Editor, pero sirven de gran ayuda tanto para el programador como para la persona que quiera entender lo que hace el programa. En PBASIC, toda línea que se inicia con un apóstrofe significa que es un comentario. En nuestro programa el primer comentario informa del nombre del fichero y el segundo de su finalidad.

'El Robot Home Boe-Bot – HelloBoeBot.bs2'.

Esta primera línea por comenzar con un apóstrofe se trata de un comentario que sólo sirve para informar del nombre del programa. La segunda línea es otro comentario que informa de lo que hace el programa:

'El BASIC Stamp envía un mensaje de texto al PC

Con los comentarios hay una excepción. Existen "mensajes especiales" que van precedidos por apóstrofe pero que actúan como "directivas del compilador" cuya misión es informarle de ciertas condiciones. Todos los programas tienen como mínimo dos de dichas directivas:

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

La primera directiva se llama directiva Stamp y sirve para informar al Editor BASIC Stamp que el programa va a ser descargado en un módulo de Parallax tipo Basic Stamp 2 (BS2), como el que dispone la Home Work. La segunda directiva llamada PBASIC parametriza al compilador para adaptarle al lenguaje de programación que se utiliza, y que en este caso es la versión 2.5 de PBASIC.

Un comando o instrucción es una palabra que describe una acción a realizar por el módulo BASIC Stamp. El primer comando de nuestro programa es DEBUG que sirve para mandar un mensaje al PC por el puerto serie. El mensaje es el texto entrecomillado que sigue al comando.

DEBUG "Hola, esto es un mensaje desde el Home Boe-Bot."

El segundo comando END indica la finalización del programa. Es importante que todo programa contenga este comando ya que al ejecutarse, el módulo BASIC Stamp pasa a modo de bajo consumo. En este modo la placa se queda en espera hasta que le pulsemos el botón reset o le introduzcamos otro programa mediante el Editor BASIC Stamp. Si pulsamos el botón reset, el programa comienza a ejecutarse desde el principio y si por el contrario introducimos un nuevo programa, como es lógico el anterior se borrará para siempre.

En conclusión, el primer programa ha tenido como objetivo enviar un mensaje a la pantalla del PC con la principal misión de comprobar el correcto funcionamiento de la Home Work, del PC y del programa de comunicación entre ambos.

Formatos del DEBUG y caracteres de control

El comando DEBUG dispone de diversos formatos para enviar un texto con ciertas características a la ventana de debug. DEC es un ejemplo de formato que sirve para mostrar un valor en decimal. Un ejemplo de formato para un carácter de control es CR, el cual enviará un retorno de carro a la ventana de debug. Ahora puedes modificar tu primer programa con el fin de que tenga más comandos DEBUG con distintos formatos y caracteres de control. He aquí un ejemplo de cómo hacerlo:

Modifica los comentarios y renombra el fichero a otro nombre (por ejemplo: HelloBoeBotYourTurn.bs2)

```
' El Robot Home Boe-Bot - HelloBoeBot.bs2
' El BASIC Stamp realiza un cálculo, y envía el resultado
' al terminal de DEBUG.
```

Añade estas tres líneas entre el primer DEBUG que pusiste antes y el comando END.

```
DEBUG CR, "Cuánto es 7 X 11?"
DEBUG CR, "El resultado es: "
```

DEBUG DEC 7 * 11

Guarda los cambios, deberás tener un código similar a éste (Figura 2-24):

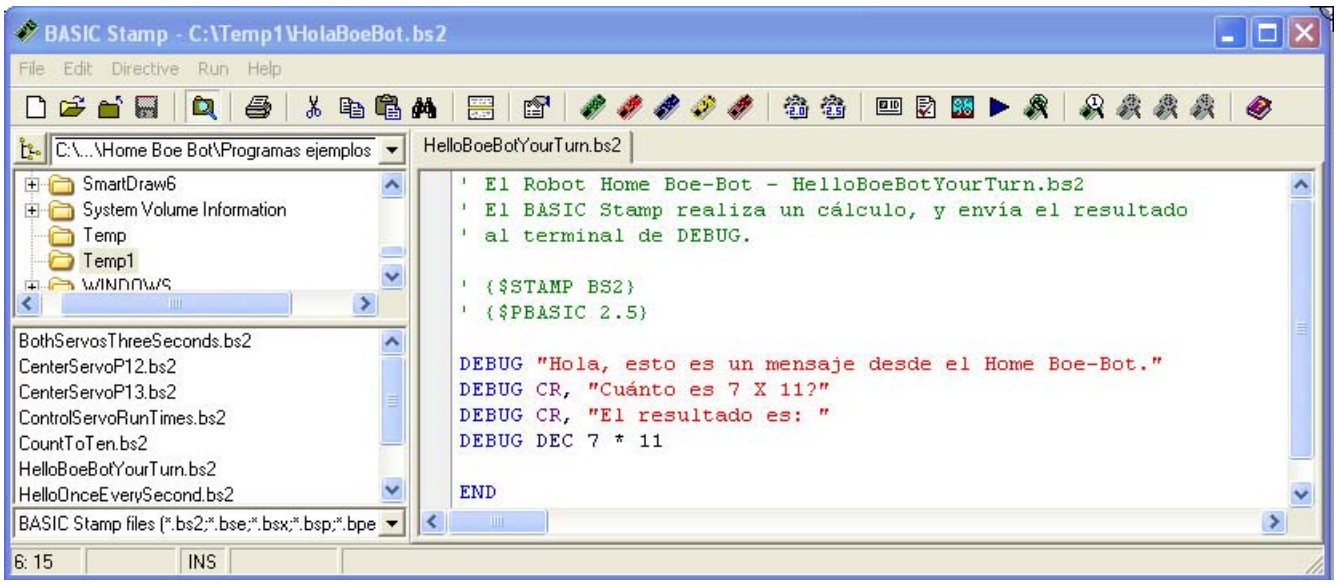


Figura 2-24.- *El nuevo programa modificado.*

Ahora ejecuta (Run | Run) el nuevo programa. Debe visualizarse en la ventana de debug un texto como el de la Figura 2-25.

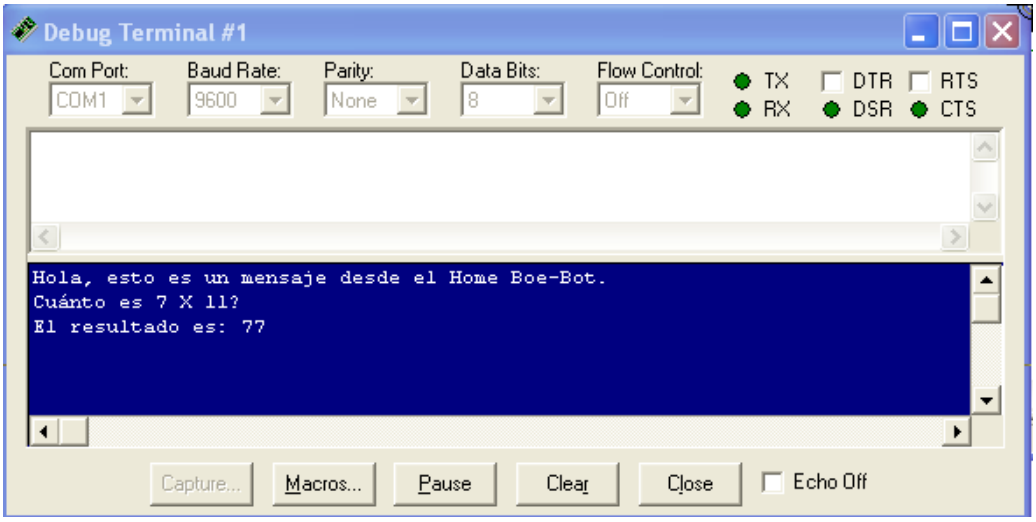


Figura 2-25.- *Ventana de debug que aparece al ejecutar el nuevo programa.*

Recuerda que si se expresan operaciones como 7 * 11, el editor las calcula y genera el resultado. El símbolo % detrás del comando DEBUG significa que se proporcionan los valores en binario, usando sólo 1 y 0. Los números detrás de DEBUG sin ningún símbolo previo representan caracteres ASCII; así por ejemplo, DEBUG 65 sirve para visualizar la letra A que responde al código 65 en ASCII.

Nota: A veces la ventana de debug no se ve, por lo que debes de ir a Run | Debug | Debug Terminal#1 (Figuras 2-26 y 2-27).

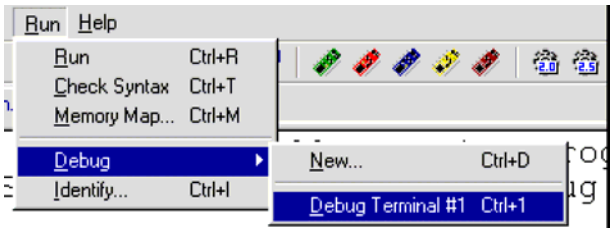


Figura 2-26.- Cuando no aparece la pantalla de debug.

Figura 2-27.- *Selección de la pantalla de Debug Terminal #1.*

2.6. PASO 5º: BUSCANDO AYUDA

Para poder obtener ayuda sobre el lenguaje de programación o sobre otros temas relacionados con el Home Boe-Bot, la Home Work o el Editor de BASIC Stamp existen cuatro métodos distintos:

1º.- Mediante la ayuda que viene integrada en el programa BASIC Stamp Editor .

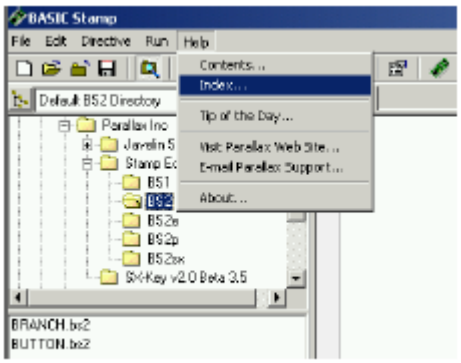


Figura 2-28.- *Ayuda que puede obtenerse en el propio Editor.*

2º.- Obteniendo el manual de BASIC Stamp, bien en el sitio Web (www.parallax.com) o en el CD de Parallax.

3º.- Revisando la información existente en las páginas de Internet de www.parallax.com y www.microcontroladores.com .

4º.- Enviando un e-mail pidiendo ayuda a alguno de los foros que mantienen Parallax e Ingeniería de Microsistemas Programados S.L. en las direcciones anteriores de Internet.

2.7. PASO 6º: Y PARA ACABAR

Una vez terminada la primera sesión de programación y manejo de la Home Work es importante desconectar la pila para ahorrar energía (Figura 2-29).

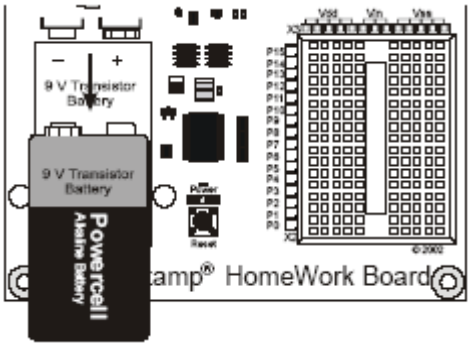


Figura 2-29.- *Tras acabar el trabajo no olvides desconectar la pila.*

2.8 PRUEBA DE AUTOEVALUACIÓN

Preguntas

1. ¿Cuál es el "cerebro" del Home Boe-Bot?
2. ¿Cuál es el propósito del cable serie?
3. Cuando la Home Work envía caracteres al PC, ¿qué tipo de números son enviados a través del cable serie?
4. ¿Qué debes hacer entre la edición del programa y la ejecución del mismo?
5. ¿Cuál es el nombre de la ventana que muestra mensajes enviados desde la Home Work a tu PC?
6. ¿Qué significa el apóstrofe en el comienzo de línea en un programa de PBASIC?
7. ¿Qué comandos o instrucciones PBASIC has aprendido en este tema?
8. ¿Qué debes hacer una vez finalizado el ejercicio o experiencia?

Ejercicios

1. Explica lo que puedes hacer con cada comando que se ha explicado en el tema.
2. Explica qué pasaría si cogiéramos todos los comandos de control CR y los pusiéramos fuera del comando DEBUG y a continuación explica qué texto aparece en la ventana de debug.
3. Explica qué realiza el asterisco en esta instrucción:

```
DEBUG DEC 7 * 11
```

4. Adivina qué aparecería en la ventana del terminal si escribiéramos esta línea de código:

```
DEBUG DEC 7 + 11
```

5. Hay un problema con los dos comandos de los ejercicios 3 y 4. Cuando ejecutas el código, los números que se muestran salen seguidos, sin ningún tipo de espacio entre sí, con lo que parece que es un solo número. Modifica estos dos comandos con el fin de que sus resultados se vean en distintas líneas.

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

Proyectos

1. Utiliza el comando DEBUG para mostrar la solución de 1+2+3+4.
2. Utiliza los formatos de debug explicados como plantilla para realizar tu nuevo proyecto y modificar el original. Utiliza el nombre de HelloBoeBotCh01Project02.bs2

Ahora añade esta línea al programa y ejecútalo:

```
DEBUG 65, 66, 67, 68, 69, 70
```

Después, escribe el formato DEC antes de cada número, vuelve a ejecutar el programa y describe lo que hace dicho formato. El código ASCII para la A es 65, para la B 66, y así sucesivamente. El espacio está representado por el 32 y el salto de línea por el 13. Ejecuta el siguiente trozo de código y explica en un párrafo que hace cada línea:

```
DEBUG "Hola !"
DEBUG 32, 32, 32, 32
DEBUG "Hola de nuevo"
DEBUG 13
DEBUG "Adios."
```

3. Intenta adivinar que ocurriría si quitas el formato DEC. Escribe un programa en PBASIC para confirmar si has acertado.

```
DEBUG DEC 7 * 11
```

4. Échale un vistazo a la figura 2-18. ¿Cómo puedes mandar un numero 65 (A) utilizando únicamente unos y ceros? Modifica el programa añadiendo el siguiente segmento de código y verifica que hace lo mismo que el Proyecto 2.

```
' Envíar los códigos ASCII de 65, 66, 67, 68, 69, y 70.
' El terminal de DEBUG debe visualizar "ABCDEF".
DEBUG %01000001, %01000010, %01000011
DEBUG %01000100, %01000101, %01000110
```

5. ¿Qué líneas de código se pueden eliminar del HelloBoeBotYourTurn.bs2 si añades la línea mostrada abajo justo antes del END?. Asegúrate de que salvas el programa con un nombre distinto.

```
DEBUG "Cuánto es  7 x 11?", CR, "El resultado es: ", DEC 7 * 11
```

2.9 PROGRAMAR UN COMPUTADOR ES DECIRLE LO QUE TIENE QUE HACER

Un computador es una máquina electrónica que sólo sabe realizar unas pocas operaciones con datos, pero a mucha velocidad. Su potencia está basada más en la velocidad que en la complejidad de las tareas que puede llevar a cabo. Muchos computadores no saben ni multiplicar, pero hacen tantos millones de sumas por segundo que podrían proporcionar el resultado de una complicadísima multiplicación en un instante.

La tarjeta Home Work va a ser la que gobierne a nuestro robot y para ello dispone de un pequeño circuito integrado, llamado PIC16C57, que es un computador enano o microcontrolador. Sólo es capaz de interpretar 35 operaciones muy sencillas con datos de ocho bits, pero a una velocidad enorme. Para que haga una cualquiera de dichas operaciones hay que especificarle claramente de cuál se trata y para ello se emplea generalmente una palabra que define la misión de la operación. La palabra que informa al computador la operación que debe realizar está expresada en inglés en el lenguaje PBASIC y se denomina comando o instrucción.

En esta sección vamos a introducirte en la programación usando el lenguaje PBASIC montando pequeños sistemas que van a funcionar como nos interese. Vas a aplicar el lenguaje a la resolución de semáforos, alarmas y otros circuitos y eso será suficiente para que a partir del tema 3 encares el montaje y la programación del Home Boe-Bot. Si quieres ampliar estos conocimientos y realizar muchos más ejercicios y experimentos prácticos de programación te recomendamos una obra excepcional para los que se inician con los microcontroladores y el lenguaje PBASIC, se trata del libro *"DISEÑO PRÁCTICO CON MICROCONTROLADORES PARA TODOS",* escrito por José Mª Angulo, Susana Romero e Ignacio Angulo, y editado en 2004 por la editorial Thomson-Paraninfo.

Un ejemplo que demuestra la sencillez de programar

El fabricante de la Home Work, Parallax Inc., ha creado un lenguaje formado por unas pocas decenas de instrucciones que recibe el nombre de PBASIC, que son las que la tarjeta y el microcontrolador que incluye son capaces de entender y ejecutar. Se llama programa a un conjunto ordenado de instrucciones colocadas para conseguir un resultado.

Una operación elemental, pero muy importante, que puede reconocer y ejecutar el microcontrolador de la Home Work es contar tiempo. Es capaz de esperar un tiempo exacto para proseguir con su tarea de ejecutar las instrucciones siguientes. Para soportar la instrucción de temporización o medida del tiempo el lenguaje PBASIC dispone de una instrucción o comando de nombre PAUSE. Detrás de dicha palabra basta con añadir el tiempo que se desea que el microcontrolador temporice. El tiempo se expresa en milisegundos, por tanto la instrucción PAUSE 2500 regula un tiempo de 2.500 milisegundos equivalentes a 2,5 segundos.

Un programa de comprobación

Como en la sección anterior hicistes y ejecutastes un programa en PBASIC ya conoces tres de sus instrucciones: DEBUG, END y PAUSE, que acabamos de describir. Con ellas vamos a escribir un programa y comprobar que la práctica responde a la teoría, así que hazte con un reloj preciso para medir tiempo.

Conecta la Home Work al PC, ejecuta el programa BASIC Stamp Editor y con la ventana correspondiente en la pantalla inserta la pila de 9 V y edita un programa al que vamos a llamar PRUEBA3_1.bs2 tecleando lo siguiente.

```
' Programa PRUEBA3_1.bs2
'{$ STAMP BS2}
'{$ PBASIC 2.5}

DEBUG "PROBANDO LA INSTRUCCIÓN PAUSE", CR
DEBUG "INICIO DE LA CUENTA DEL TIEMPO", CR
PAUSE 5000
DEBUG "¿HAN PASADO 5 SEGUNDOS?", CR
```

```
END
```

Al ejecutarse este programa hay que poner en marcha la cuenta del tiempo con el reloj al aparecer el mensaje de "INICIO DE LA CUENTA DEL TIEMPO". Aproximadamente habrán transcurrido unos 5 segundos de tiempo al visualizarse el mensaje ¿HAN PASADO 5 SEGUNDOS?. Compruébalo y luego modifica el programa cambiando el tiempo que sigue a PAUSE. Intenta calcular cuál es el tiempo máximo que se puede contar con PAUSE.

2.10 Y SE HIZO LA LUZ

Hasta ahora sólo hemos empleado la tarjeta Home Work y la pantalla del PC y hemos sido capaces de enviar mensajes con los programas y contar el tiempo. Vamos a comenzar a controlar el funcionamiento de diversos dispositivos materiales, para lo cual les conectaremos a las patitas de entrada y salida de la Home Work para que ella con su microcontrolador les gobierne.

Vamos a comenzar regulando el comportamiento de una luz pequeñita que consume muy poco y la has visto innumerables veces como "piloto" de casi todos los aparatos electrónicos. Por ejemplo, la mayoría de los aparatos, como la TV, cuando se encienden al conectarse a la red y pulsar el botón ON se les ilumina un pequeña luz roja que indica que reciben la energía eléctrica. Estas lucecitas se denominan LED (Diodos Electroluminiscentes) y son dispositivos diodos semiconductores de Silicio y otros materiales que disponen de dos patitas o terminales y cuando se les aplica una diferencia de tensión entre ambos con la polaridad correctamente aplicada se encienden y emiten una luz que puede ser de diversos colores. Un terminal se denomina ánodo y se corresponde con el más largo, y el otro, cátodo y es el más corto y el que está situado en la parte achaflanada de la cápsula que contiene al LED. En la figura 2-30 se muestra el aspecto externo de un LED y el símbolo que se usa en los esquemas para representarlo.

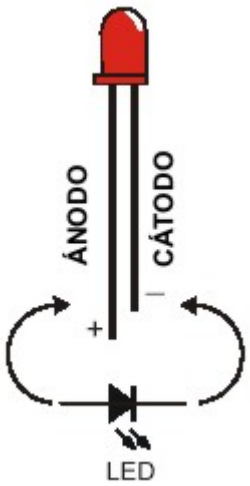


Figura 2-30.- *Aspecto externo y símbolo eléctrico del diodo LED. El terminal más corto situado en la zona del chaflán de la cápsula es el cátodo y el otro es el ánodo.*

Para que se ilumine el LED hay que aplicar unos 2 ó 3 V entre sus terminales, de forma que el polo positivo se aplique al ánodo y el negativo al cátodo. Como por las patitas de salida de la Home Work se van a obtener 5 V de tensión hay que colocar delante del LED una resistencia que se encargará de absorber la tensión sobrante. La resistencia es un dispositivo que no tiene polaridad y da lo mismo la posición en la que se coloquen sus dos terminales. En la figura 2-31 se ofrece el esquema de encendido de un LED con su resistencia de absorción cuando se aplican 5 V de tensión a ambos componentes.

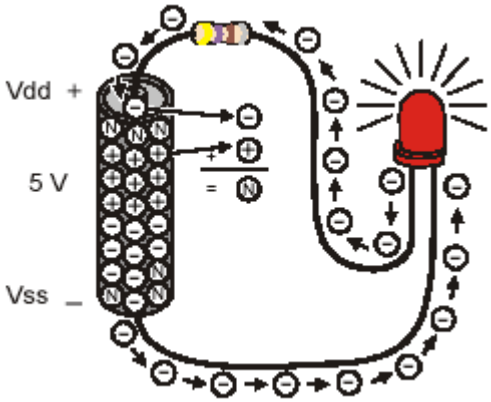


Figura 2-31.- *Los 5 v se reparten entre el LED y la resistencia. Para que el LED se ilumine su ánodo debe recibir el polo positivo y el cátodo el negativo.*

2.11 UN VISTAZO A LAS RESISTENCIAS DE COLORES

El efecto de absorción de tensión de las resistencias depende del valor que tengan, que se mide en ohmios (Ω). Una buena iluminación del LED puede conseguirse con una resistencia de 220 Ω, aunque también se puede conseguir que se encienda algo con una resistencia de 470 Ω. Como las resistencias suelen ser muy pequeñas y cilíndricas no se puede grabar fácilmente su valor en el cuerpo y por eso se utiliza un código de colores para indicar su valor. El código de colores usado en las resistencias se indica a continuación en la Figura 2-32.

COLOR	NUMERO
Negro	0
Marrón	1
Rojo	2
Naranja	3
Amarillo	4
Verde	5
Azul	6
Violeta	7
Gris	8
Blanco	9

Figura 2-32.- *Para hallar el valor de las resistencias se usan colores que representan números decimales.*

Además de estos colores que se usan para averiguar el valor, existen otros dos encargados de informar sobre la tolerancia o "exactitud" del valor indicado por los colores. Así, si la última franja tiene color ORO significa que el valor que tiene la resistencia real no se desvía más del 5% del que indican sus colores. En caso de ser PLATA la tolerancia es del 10 %.

Para hallar el valor aproximado de una resistencia por sus colores hay que situarla frente a nosotros con sus cuatro franjas de forma que la última, la de la derecha, sea la que corresponde al color de la tolerancia. Figura 2-33.

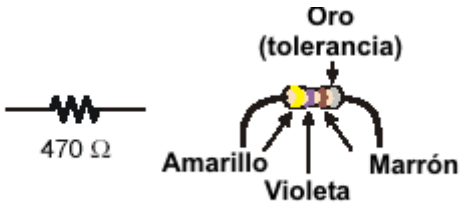


Figura 2-33.- *El valor de esta resistencia con 4 colores: amarillo, violeta, marrón y oro es de 470 Ω, con una tolerancia del 5%.*

Las normas a seguir para averiguar el valor de una resistencia son.

1ª.- El último color es el de la tolerancia que puede ser oro (5%) y plata (10%). Si no existe ninguno de los dos la tolerancia es del 20%.
2ª. El color de la primera franja se corresponde con el primer número del valor.
3ª. El color de la segunda franja se corresponde con el segundo número del valor
4ª. El color de la tercera franja se corresponde con el número de ceros que hay que añadir a los dos anteriores para expresar el valor en ohmios.

Si se dispone de una resistencia con cuatro franjas de colores: amarillo, violeta, marrón y oro, el valor de la resistencia será de 470 ohmios (amarillo, violeta y marón) y una tolerancia del 5% (oro). Obsérvese que el tercer color indica el número de ceros que hay que añadir a los dos primeros números. En nuestro caso por ser marrón habrá que añadir un cero. Como la tolerancia es del 5% significa que el fabricante garantiza que el valor real de esa resistencia no se desvía más del 5% del que se ha indicado con los colores. Como el 5% de 470 ohmios son 23,5 ohmios, el valor real de la resistencia estará comprendido entre 470 + 23,5 y 470 – 23,5 ohmios, o sea, entre 493,5 y 446,5 Ω.

2.12 EL CIRCUITO PRACTICO

Como se aprecia en el esquema de la figura 2-34 para encender a un LED basta conectarle en serie una resistencia y aplicarle una tensíon de 5V, de forma que el polo positivo se conecte al ánodo y el negativo al cátodo.

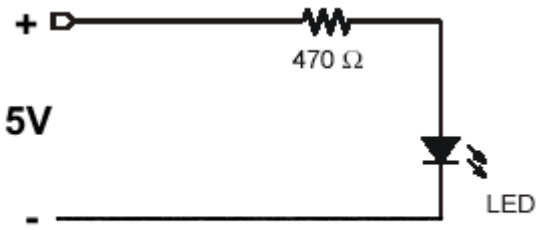


Figura 2-34.- *Circuito práctico para encender un LED.*

Generalmente en los circuitos prácticos el polo negativo de la alimentación se conecta a un terminal común que se llama TIERRA y se representa con la tensión Vss. En la figura 2-35 también se representa el polo positivo con el valor Vdd.

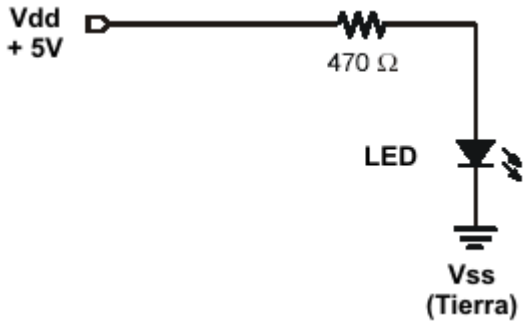


Figura 2-35.- *En este circuito los polos de la tensión de alimentación se representan con Vdd (+) y Vss (-).*

2.13 LA ZONA PARA EL MONTAJE DE LOS COMPONENTES

La tarjeta Home Work dispone de una superficie destinada al montaje de los componentes de los circuitos prácticos basada en una placa protoboard con numerosos orificios interconectados que permiten realizar la conexión entre los componentes sin usar soldadura y sin estropearlos. Los terminales de los componentes entran a presión por dichos orificios.

La protoboard de la Home Work consta de dos zonas blancas de 17 filas con cinco orificios internconectados entre sí cada una. Los cinco orificios de cada fila están conectados entre sí. Figura 2-36. Si a un orificio de la protoboard se introduce a presión el terminal del cátodo del LED y a otro orificio de la misma fila de cinco oirificios se introduce un terminal de una resistencia, quedarán conectados el cátodo del LED con el terminal de la resistencia.

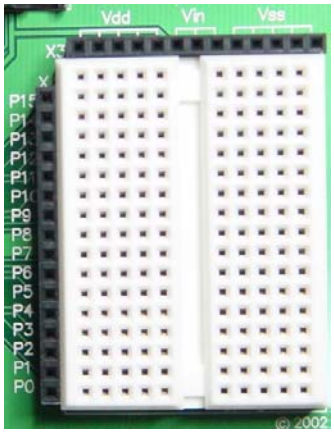


Figura 2-36.- Los cinco orificios de cada fila de cada sección están conectados internamente entre sí.

A la izquierda y en la parte superior de la protoboard existen dos conectores negros con orificios. La tira vertical se llama X4 y tiene 16 orificios que se denominan P0 a P15 y se hallan conectando las 16 líneas de entrada o salida del microcontrolador. Nuestros programas podrán definir a cada línea como entrada o salida y en éstas últimas podremos sacar nivel alto o + 5 V,o bien , nivel bajo o 0 V. Si la línea es de entrada el microcontrolador leerá en ella la tensión exterior aplicada y reconocerá nivel alto o nivel bajo. El nivel alto puede estar comprendido entre 2 y 5 V, mientras que el bajo siempre será inferior a 1,4 V. En resumen, las patitas P0 a P15 cuando actúan como salida pueden sacar niveles lógicos altos (Vdd) o bajos (Vss). A veces al nivel alto le llamamos 1 y al bajo 0.

El conector negro superior se denomina X3 y consta de 13 orificios, correspondiendo los cinco de la izquierda a los que sacan la tensión vdd (+ 5V) y los cinco de la derecha sacan la tensión Vss (0 V). Los tres

terminales centrales se llaman Vin y proporcionan el voltaje positivo de la pila de entrada, que aún no ha sido regulada y estabilizada a + 5 V, proporcionando los terminales Vss el polo negativo.

2.14 HACIENDO PARPADEAR UN LED

El primer experimento real con control de dispositivos va a consistir en encender un LED 3 segundos, luego volverle a apagar otros 3 segundos y repetir estas dos acciones tres veces. En resumen haremos parpadear el LED tres veces cada 3 segundos.

Elegiremos una de las patitas de entrada/salida de la tarjeta Home Work , la P0, y haremos que actúe como salida generando por ella nivel lógico alto (+ 5V) durante 3 segundos que encenderá al LED conectado a dicha patita a través de una resistencia de absorción. En los 3 segundos posteriores saldrá por la patita anterior un nivel bajo (0 V) y el LED permanecerá apagado. Figura 2-37.

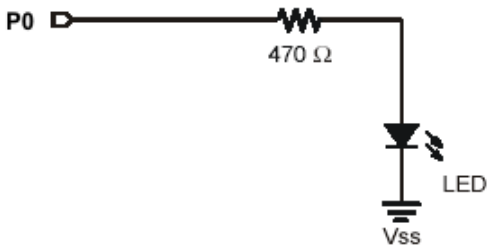


Figura 2-37.- *Cuando por la patita P0 sale nivel alto (+ 5 V) el LED se enciende y si sale nivel bajo (0 V) se apaga.*

Para montar el circuito de la figura 2-37 sobre la placa protoboard de la Home Work se introduce el ánodo del LED en el orificio P0 y el cátodo en un orificio de una fila cercana. A otro orificio de dicha fila se introduce un terminal cualquiera de la resistencia de 470 Ω, mientras que el terminal restante se conecta a un orificio que suministra la tierra y está marcado en el conector X3 como Vss como se presenta en la fotografía de la figura 2-38.

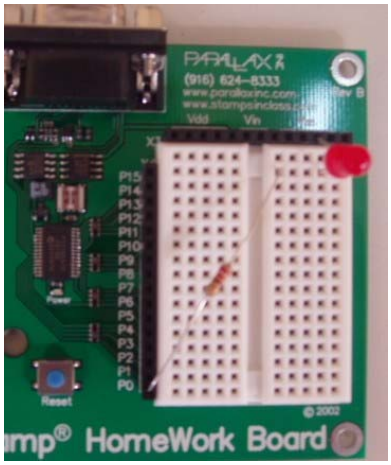


Figura 2-38. *Fotografía del montaje del circuito diseñado para hacer parpadear al led*

2.15. EL PROGRAMA DE PARPADEO

Para encender el LED del montaje de la figura 2-37 hay que sacar un nivel alto por P0 y para apagarle debe salir un nivel bajo. El lenguaje PBASIC dispone de una instrucción que configura a una de las patitas como salida y saca por ella un nivel alto. Otra instrucción PBASIC puede configurar una patita de E/S de la tarjeta Home Work como salida y sacar por ella un nivel bajo.

El comando HIGH seguido por el número de la patita que se desea saca por dicha patita un nivel alto.. Por ejemplo, HIGH 0, saca un nivel alto (Vdd) por la patita P0, lo que originará el encendido del LED del circuito de parpadeo. La instrucción LOW 0 consigue sacar un nivel bajo (Vss) por P0 y apagar el LED mencionado.

Para mantener el estado del microcontrolador un tiempo determinado se utiliza la instrucción PAUSE seguida del número de milisegundos que se desee esperar. Para controlar espacios de tiempo de 3 segundos se utilizará PAUSE 3000.

El programa PARPADEO LED se iniciará con los comandos habituales referentes a la especificación del tipo de módulo de Parallax empleado (BS2) y la versión del lenguaje PBASIC (2.5). Luego repetirá tres veces el encendido del LED durante 3 segundos seguido del apagado del mismo tiempo y con la instrucción DEBUG se visualizará sobre la pantalla del PC se comenzará indicando el nombre del programa.

```
'PROGRAMA "PARPADEO LED"
'{ $STAMP BS2 }
'{ $PBASIC 2.5 }

DEBUG "PARPADEO TRES VECES, 3 SEGUNDOS"

HIGH   0                  'Sale por P0 un nivel alto
PAUSE 3000                ' Pausa de 3 segundos
LOW    0                  'Sale nivel bajo por P0
PAUSE 3000

HIGH   0                  'Sale por P0 un nivel alto
PAUSE 3000                ' Pausa de 3 segundos
LOW    0                  'Sale nivel bajo por P0
PAUSE 3000

HIGH   0                  'Sale por P0 un nivel alto
PAUSE 3000                ' Pausa de 3 segundos
LOW    0                  'Sale nivel bajo por P0
PAUSE 3000
END                       'Fin del programa
```

Una vez que hayas tecleado y editado el programa guárdalo con el nombre de PARPADEO LED y ejecútalo sobre la home Work con la pila conectada. ¿Parpadea tres veces el LED cada 3 segundos?.

Sería muy fácil cambiar el tiempo del parpadeo cada 10 segundos (PAUSE 10000) o repetir el parpadeo seis veces en lugar de tres, pero ¿qué habría que hacer para que el LED quedase parpadeando de forma indefinida?. ¿Habría que repetir la secuencia HIGH O, PAUSE 3000, LOW 0 y PAUSE 3000 miles de veces?. Existe una forma muy cómoda y fácil de repetir una secuencia de instrucciones continuamente con una instrucción PPBASIC llamada GOTO. Con esta instrucción se pasa a ejecutar la instrucción que se desee, para lo cual basta con poner en dicha instrucción una etiqueta identificativa previa, con dos puntos, y colocar dicha etiqueta detrás del comando GOTO. Para el caso del parpadeo indefinido la primera instrucción de la secuencia que se quiere repetir es HIGH 0 y como al acabar la secuencia se quiere regresar a ella se la coloca delante una etiqueta (INICIO :), que es la que se pone detrás de GOTO para que el flujo de control del programa salte a ella para ejecutarla, como se muestra en el programa:

```
'PROGRAMA "PARPADEO INFINITO"
'{ $STAMP BS2 }
'{ $PBASIC 2.5 }

DEBUG "PARPADEO INFINITO DEL LED"
```

```
INICIO:     HIGH   0              'Sale por P0 un nivel alto
            PAUSE 3000            ' Pausa de 3 segundos
            LOW    0              'Sale nivel bajo por P0
            PAUSE 3000

            GOTO  INICIO          'Se salta a la instrucción con etiqueta INICIO
```

Al ejecutar este programa el LED permanecerá parpadeando durante tres segundos de forma indefinida.

2.16. SEMÁFORO SONORO

Para avanzar en el manejo de las instrucciones PBASIC se propone montar un circuito y confeccionar un programa que simule un semáforo que normalmente está cerrado, en rojo, para los peatones y abierto para los vehículos. Existe un pulsador que al presionarlo cambia el estado del semáforo durante 10 segundas para permitir el cruce de los peatones. Además, cuando los peatones tienen encendido la luz verde de paso, un zumbador lo confirma generando un pitido.

El semáforo lo simularemos con dos diodos LED que indican el estado para los peatones. Uno de ellos puede ser de color rojo y el otro verde. El estado habitual del semáforo es estar en rojo para los peatones. Cuando se presiona un pulsador se enciende el LED verde y se apaga el rojo, además genera un pitido el zumbador. El cruce de los peatones dura 10 segundos.

En la Home Work los LED estarán controlados por dos patitas P0 y P2 que actuarán como salidas. Cada LED dispone de su resistencia de absorción. El zumbador piezoeléctrico es un dispositivo con dos terminales con polaridad. Uno de ellos tiene marcado el positivo (+) y a él se debe conectar el polo positivo de la alimentación. También el zumbador dispone de una resistencia de absorción y está controlado por la patita P15 como se muestra en el circuito de la figura 2-39.

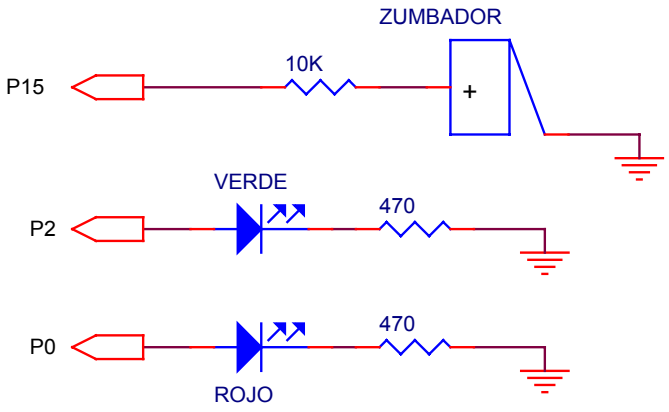


Figura 2-39.- *Circuito eléctrico de conexionado de los tres dispositivos de salida del semáforo sonoro.*

El pulsador es un periférico de entrada porque se encarga de introducir a la Home Work un nivel lógico alto cuando no está presionado, mientras que si se presiona introduce por la patita P5 un nivel bajo. El circuito de conexionado y funcionamiento del pulsador es el de la figura 2-40.

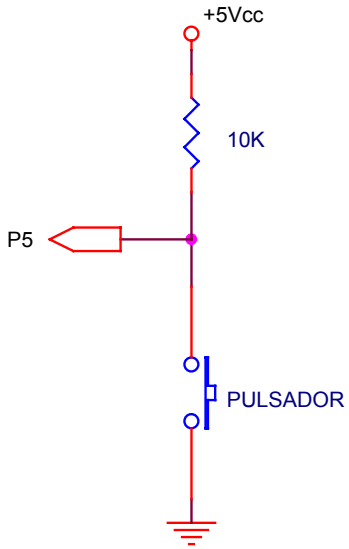


Figura 2-40.- *Cuando el pulsador no está presionado se introduce por la patita P5 un nivel lógico alto (+ 5 V) a través de la resistencia. Cuando se presiona introduce un nivel bajo al conectar P5 a tierra.*

En la fotografía de la figura 2-41 se presenta el semáforo sonoro montado sobre la tarjeta Home Work.

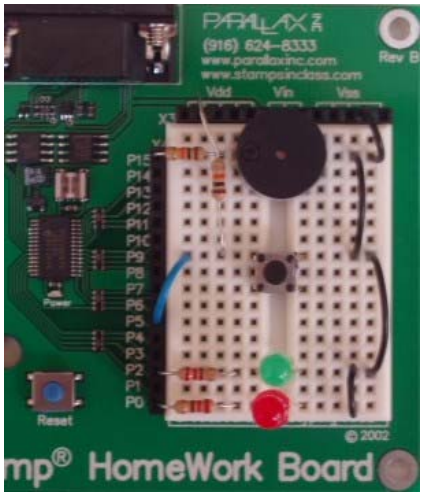


Figura 2-41. *Fotografía del montaje del semáforo sonoro sobre la Home Work*

El programa del semáforo sonoro

Para indicar el estado habitual del semáforo el programa deberá conseguir el siguiente estado.

LED VERDE APAGADO
LED ROJO ENCENDIDO
ZUMBADOR CALLADO
PULSADOR NO PRESIONADO

Al presionar el pulsador e introducir un nivel bajo por P5 se invierte el estado del semáforo y tras 10 segundos retorna al estado habitual. Durante dicho tiempo el estado del semáforo es el siguiente.

LED VERDE ENCENDIDO
LED ROJO APAGADO
ZUMBADOR GENERANDO PITIDO
PULSADOR SE HA PRESIONADO UN INSTANTE

El estado de los dispositivos de salida del semáforo sonoro depende del nivel introducido por el pulsador por P5. Dicho nivel o valor del pulsador será una variable X que puede tomar dos estados (variable binaria). Un estado llamaremos 1 cuando el pulsador no está presionado y se introduce nivel alto por P5. Cuando se presiona el pulsador la variable X pasará a valer 0 e indicará que se está introduciendo nivel bajo por P5. El programa manejará una variable binaria X cuyo valor depende del pulsador. Los dispositivos de salida toman un estado que depende del que tenga en cada momento la variable X. Por eso el programa deberá analizar X y según valga 0 ó 1 los dispositivos de salida tomarán un estado u otro. Para evaluar una condición y pasar a ejecutar una secuencia de instrucciones u otra el lenguaje PBASIC dispone de la instrucción:

IF condición THEN etiqueta (Si se cumple la condición se salta a la instrucción de la etiqueta)

En el ejemplo del semáforo sonoro la condición de la anterior instrucción se corresponderá con el valor de la variable X. Si X = 0 el pulsador está presionado y entonces (THEN) se debe saltar a la instrucción correspondiente a la etiqueta que se indica. En caso contrario que X = 1 se ejecuta la siguiente instrucción a la IF...THEN. Recuérdese que cuando X = 0 se debe poner a los dispositivos de salida en un estado transitorio durante 10 segundos que corresponde con el permiso de cruce de los peatones. El programa que cumple estos condicionantes puede ser el siguiente.

```
'PROGRAMA "SEMÁFORO SONORO"
'{ $STAMP BS2 }
'{ $PBASIC 2.5 }

DEBUG "AL PRESIONAR EL PULSADOR SE PERMITE CRUCE DE PEATONES DURANTE 10 SEGUNDOS"

        X       VAR     BIT             'Se declara la variable X de tipo binario

INICIO: X = IN5                         'La variable X toma el valor de P5 (pulsador)

        IF X = 0 THEN PASO              'Si X=0 entonces saltar a la instrucción PASO:

        HIGH    2                       'Si X = 1 el semáforo en estado normal
        LOW     0
        LOW     15
        GOTO INICIO                     'Vuelve a testearse el pulsador

PASO:   LOW     2                       'Si X=0 pasa el semáforo al estado transitorio
        HIGH    0
        HIGH    15
        PAUSE 10000                     ' Pausa de 10 segundos

        GOTO  INICIO                    'Vuelve a testearse el pulsador
```

Debes editar el programa, cargarlo con el nombre de SEMÁFORO SONORO y ejecutarlo. Si todo va bien debes intentar modificar y mejorar el semáforo añadiendo todas las acciones que estimes oportunas.

2.17. CONTROLANDO EL NUMERO DE REPETICIONES

Para completar un vistazo rápido del manejo de las principales instrucciones del PBASIC vamos a confeccionar un programa derivado del semáforo sonoro. En lugar de que el zumbador pite continuamente durante los 10 segundos en los que pueden cruzar los peatones lo que va a generar son pitidos y silencios alternados cada segundo. Esto supone que "aproximadamente" durante los 10 segundos habrá 5 intervalos de 1 segundo en los que el zumbador pita y otros tantos de la misma duración en los que está en silencio.

El programa que sólo controla al zumbador y le hace pitar y callar durante 10 segundos en intervalos de 1 segundo podía ser el siguiente que puedes probar si funciona en la práctica.

```
PROGRAMA "INTERMITENCIA DEL ZUMBADOR"
'{ $STAMP BS2 }
'{ $PBASIC 2.5 }

DEBUG "EL ZUMBADOR PITA Y CALLA CADA SEGUNDO DURANTE 10 SEGUNDOS"

HIGH   15          'Se activa el zumbador y pita
PAUSE  1000        'Retardo de 1 segundo
LOW    15          'Se calla el zumbador
PAUSE 1000         'Retardo de 1 segundo

HIGH   15          'Se activa el zumbador y pita
PAUSE  1000        'Retardo de 1 segundo
LOW    15          'Se calla el zumbador
PAUSE 1000         'Retardo de 1 segundo

HIGH   15          'Se activa el zumbador y pita
PAUSE  1000        'Retardo de 1 segundo
LOW    15          'Se calla el zumbador
PAUSE 1000         'Retardo de 1 segundo

HIGH   15          'Se activa el zumbador y pita
PAUSE  1000        'Retardo de 1 segundo
LOW    15          'Se calla el zumbador
PAUSE 1000         'Retardo de 1 segundo

HIGH   15          'Se activa el zumbador y pita
PAUSE  1000        'Retardo de 1 segundo
LOW    15          'Se calla el zumbador
PAUSE 1000         'Retardo de 1 segundo

END
```

Teclea este programa, sálvalo con el nombre de INTERMITENCIA ZUMBADOR y ejecútalo sobre la Home Work con la pila conectada y el circuito del semáforo sonoro montado. Sólo deberá activarse el zumbador con 5 intermitencias de pitido sonido de 1 segundo. Observa lo que pasa con los dispositivos restantes a los que el programa no les controla.

El programa anterior repite cinco veces la misma secuencia de instrucciones haciendo muy larga y tediosa la edición del mismo. El lenguaje PBASIC dispone de una instrucción que posibilita repetir una secuencia de instrucciones el número de veces que se desee. Tiene una estructura que comienza con

FOR VARIABLE = 1 TO N

A la que sigue la secuencia de instrucciones que se desea repetir N veces. Tras la secuencia repetitiva se coloca el comando NEXT. Por ejemplo, si se quiere repetir cinco veces la alternancia del zumbador (HIGH 15, PAUSE 1000, LOW 15, PAUSE 1000) se podría confeccionar este programa.

```
PROGRAMA "REPETICIÓN CONTROLADA"
'{ $STAMP BS2 }
'{ $PBASIC 2.5 }
```

```
DEBUG "EL ZUMBADOR PITA Y CALLA CADA SEGUNDO DURANTE 10 SEGUNDOS"

CONTADOR    VAR    BYTE          'Contador es una variable de tipo byte

FOR CONTADOR = 1  TO  5          'Desde CONTADOR=1 hasta 5 repetir

HIGH   15                        'Inicio de la secuencia repetita
PAUSE 1000
LOW    15
PAUSE 1000

NEXT                         ' Repetición de la secuencia hasta completar veces variable
```

En el programa anterior la línea CONTADOR VAR BYTE sirve para definir una variable llamada CONTADOR que tiene un tamaño byte lo que supone que su valor máximo puede alcanzar 255. En nuestro caso el valor a alcanzar en el programa es 5. También se pueden declarar variables de tamaño WORD, en cuyo caso el máximo valor posible es de 65.535.

La instrucción FOR variable = A TO B lo que hace es repetir la secuencia de instrucciones que la siguen desde que la variable vale A hasta que llega a valer B, al incrementarse una unidad cada vez que se repite la secuencia. Se podría alcanzar el valor final B en pasos de 2 en 2 o de 5 en 5, en cuyo caso detrás de la línea FOR ... TO... habría que indicar el valor de cada repetición con el comando STEP 2 , STEP 5, etc.

Intenta ejecutar el programa y si sale bien haz un nuevo programa de SEMÁFORO SONORO con las últimas instrucciones que has aprendido.

2.18. LAS SUBRUTINAS

Es frecuente que un programa deba ejecutar una tarea en diversas ocasiones. Un posible ejemplo sería una aplicación en la que se tuviese que generar el pitido alterno del zumbador en varias situaciones diferentes. Esto exigiría tener que insertar en el programa esa tarea repetitiva tantas veces como fuese necesario. Si a la parte que se repite se la llama subrutina, el lenguaje PBASIC permite escribirla una sola vez en el programa y cada vez que se precise realizar una llamada a la misma. Se antepone a la secuencia de instrucciones de la subrutina la etiqueta identificativa oportuna y cada vez que se quiera ejecutar se utiliza el comando GOSUB "etiqueta" (IR a la SUBRUTINA etiqueta). La secuencia de la subrutina debe terminar con la instrucción RETURN que provoca que el flujo de control pase a realizar la siguiente instrucción a GOSUB etiqueta que hizo la llamada.

Para manejar las subrutinas se propone confeccionar el programa del semáforo sonoro pero en el que se contemple como una subrutina el conjunto de instrucciones correspondiente a las cinco alternancias de 1 segundo de pitido-silencio del zumbador. A este programa final le llamaremos SEMÁFORO SONORO COMPLETO.

```
PROGRAMA "SEMÁFORO SONORO COMPLETO"
'{ $STAMP BS2 }
'{ $PBASIC 2.5 }

DEBUG "PROGRAMA DEL SEMAFORO SONORO CON SUBRUTINA"

X             VAR    BIT
CONTADOR      VAR    BYTE

INICIO:   X = IN5
          IF X = 0  THEN  PASO
```

```
            HIGH   2
            LOW    0
            LOW    15
            GOTO   INICIO

PASO:       LOW    2
            HIGH   0
            GOSUB PITAR                 'Llamada a la subrutina PITAR
            GOTO   INICIO               'Salto a INICIO

PITAR:      FOR CONTADOR = 1 TO 5           'Subrutina del zumbador
            HIGH   15
            PAUSE 1000
            LOW    15
            PAUSE 1000
            NEXT
            RETURN

            END
```

Con estos sencillos ejercicios de montaje y programación es posible pasar a iniciar el montaje y programación del robot Home Boe-Bot que al estar basado en la tarjeta Home Work utilizará las mismas técnicas e instrucciones que las que acabamos de revisar experimentalmente. En el Anexo 1 de este Manual se recogen las características principales de todas las instrucciones PBASIC con algunos ejemplos de uso. Puedes descargarte del sitio www.parallax.com manuales más completos del PBASIC.

Sólo se han presentado las instrucciones más comunes y las técnicas de programación básicas y aunque son suficientes para acometer tu aventura con el robot si aún quieres profundizar y realizar experimentos más divertidos y complejos te recomendamos los libros:

1º) "***Diseño Práctico con Microcontroladores para todos***", de Angulo J. Mª., Romero, S. y Angulo I., editado por Thomson-Paraninfo.

2º) "***Microcontroladores PIC. Diseño Práctico de Aplicaciones***" (Primera Parte con CD), de Angulo J. Mª y Angulo, I., editado por Mc Graw-Hill en.

3º) "***Laboratorio de Prácticas de Microelectrónica***", Volumen II, de Angulo, J. Mª y editado por Mc Graw-Hill.

Tema 3

Servomotores. La fuerza de la bestia

3.1. INTRODUCCIÓN AL SERVOMOTOR DE ROTACIÓN CONTINUA

En este tema vamos a conectar, ajustar y probar los motores del Home Boe-Bot. Se trata de unos motores de corriente continua de enorme fuerza y gran precisión de giro que controlarán el movimiento de las ruedas motrices del robot. Hay que entender ciertos comandos PBASIC y algunas técnicas de programación que controlarán la dirección, velocidad y duración del movimiento de los motores. De esta manera, las experiencias #1, #2 y #5 te describirán las herramientas de programación y en las experiencias #3, #4 y #6 se aplicarán a los servomotores. Como el control de la precisión del servo es la clave para conseguir el mejor rendimiento del Home Boe-Bot, es importante completar estas experiencias antes de montar los servos en el chasis del Home Boe-Bot.

Los servomotores son motores de corriente continua que incorporan un circuito electrónico que permite controlar de forma sencilla y segura la dirección, la velocidad y la duración del giro de sus ejes mediante impulsos eléctricos.

En la figura 3-1 se muestran dos vistas externas de los servos de giro completo de Parallax. Muchas de las partes que se muestran en la figura se harán referencia en este tema. Los servos especiales de Parallax que usaremos pueden girar de forma continua en los dos sentidos, mientras que los servos estándar sólo giran unos 270º.

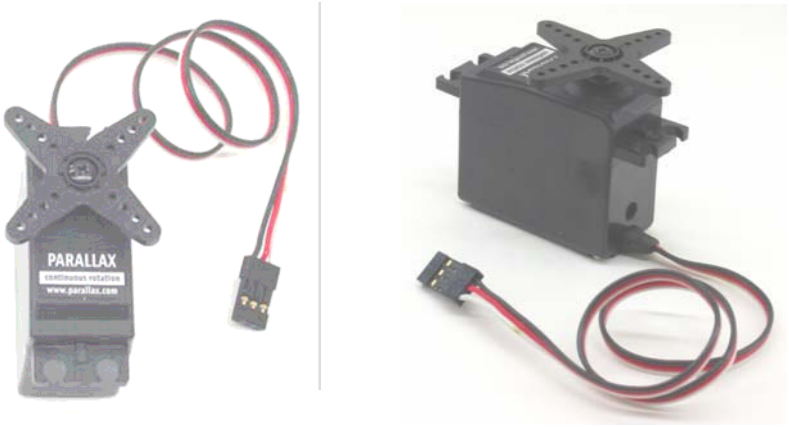


Figura 3-1.- *Aspecto externo de los servomotores de rotación completa de Parallax.*

3.2. EXPERIENCIA #1: MEDICIÓN DEL TIEMPO Y CONTROL DE REPETICIONES

Controlar la velocidad de un servo y su dirección requieren un programa que envíe desde el módulo microcontrolador un conjunto de señales repetidamente. Dichas señales tienen que enviarse unas 50 veces por segundo para que el servo mantenga su dirección y su velocidad. Esta experiencia consta de varios programas de ejemplo escritos en PBASIC, que muestran cómo repetir la misma señal y controlar su duración.

El comando **PAUSE** indica al BASIC Stamp que espere durante un tiempo antes de ejecutar el siguiente comando. Su formato es el siguiente:

PAUSE Duración

El número que pongas a la derecha del comando **PAUSE** es el parámetro ***Duración*** y es el valor que indica al BASIC Stamp cuánto tiempo tiene que esperar antes de pasar al siguiente comando. Las unidades del parámetro ***Duración*** son milisegundos (ms). De este modo, si quieres esperar un segundo, utiliza el valor 1000.

PAUSE 1000

Si quieres esperar el doble de tiempo, utiliza:

PAUSE 2000

Programa ejemplo: TimedMessages.bs2

Hay muchas formas distintas de utilizar el comando PAUSE. El programa que se propone usa PAUSE para hacer que se muestren durante un instante mensajes que dicen cuánto tiempo ha pasado. El programa debería esperar un segundo antes de enviar el mensaje "Ha transcurrido 1 segundo..." y otros dos más antes de mostrar "Han transcurrido 3 segundos...".

- Teclea el programa que se muestra a continuación en el Editor de BASIC Stamp.
- Guarda o salva el programa con el nombre TimedMessages.bs2.
- Ejecuta dicho programa y comprueba prácticamente el retardo entre mensajes.

```
' El robot Home Boe-Bot - TimedMessages.bs2
' Mostrar cómo el comando PAUSE permite visualizar mensajes
' a controlando el tiempo transcurrido

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

DEBUG "Inicio..."

PAUSE 1000
DEBUG CR, "Ha transcurrido 1 segundo..."

PAUSE 2000
DEBUG CR, "Han transcurrido3 segundos..."

DEBUG CR, "Fin."

END
```

Diferentes duraciones para Pause

Puedes cambiar el retardo entre mensajes cambiando el parámetro duración de los comandos **PAUSE**.

- Intenta cambiar los parámetros duración 1000 y 2000 por 5000 y 10000, por ejemplo:

```
DEBUG "Inicio..."
PAUSE 5000
DEBUG CR, "Han transcurrido 5 segundos..."
PAUSE 10000
DEBUG CR, "Han transcurrido 10 segundos..."
```

- Ejecuta el programa modificado.
- Prueba además con valores como 40 y 100. Irá bastante rápido.

Cómo repetir las acciones

Una de las mejores cosas de los ordenadores y de los microcontroladores es que nunca se quejan por estar haciendo la misma tarea aburrida una y otra vez. Aquí tienes un ejemplo de un programa que repite la misma operación constantemente.

Puedes colocar comandos entre las palabras **DO** y **LOOP** si quieres que se ejecuten reiteradamente.

Por ejemplo, digamos que quieres mostrar un mensaje indefinidamente, una vez cada segundo.

Simplemente coloca los comandos **DEBUG** y **PAUSE** entre las palabras **DO** y **LOOP** de esta forma:

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

Con la estructura del PBASIC formada por DO...LOOP se puede repetir de forma indefinida la secuencia de instrucciones que se encuentra entre dichos comandos

Programa ejemplo: HelloOnceEverySecond.bs2

- Teclea, guarda y ejecuta el programa HelloOnceEverySecond.bs2.
- Observa cómo el mensaje "Hola !" se muestra una vez cada segundo.

```
' El Robot Home Boe-Bot - HelloOnceEverySecond.bs2
' Visualizar un mensaje por segundo.
' {$STAMP BS2}
' {$PBASIC 2.5}

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

Un mensaje distinto

Puedes modificar el programa de tal forma que parte de él se ejecute una sola vez y otra parte se ejecute varias veces.

- Modifica el programa con estos comandos:

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

- Ejecútalo y observa qué ocurre. ¿Habías previsto el resultado?

3.3. EXPERIENCIA #2: CIRCUITO QUE MIDE EL TIEMPO Y REPITE ACCIONES

En este ejemplo construirás circuitos que emitan luz para poder "comprobar" el tipo de señales que se usan para controlar los motores del Home Boe-Bot.

Circuito de prueba de LED

- Construye el circuito mostrado en la figura 3-2.
- Asegúrate de que el extremo corto de cada LED (el cátodo) se conecta en los conectores negros etiquetados como Vss.
- Asegúrate de que el extremo largo (el ánodo, marcado con un + en el diagrama) se conecta en los conectores blancos tal y como se muestra en la figura 3-2.

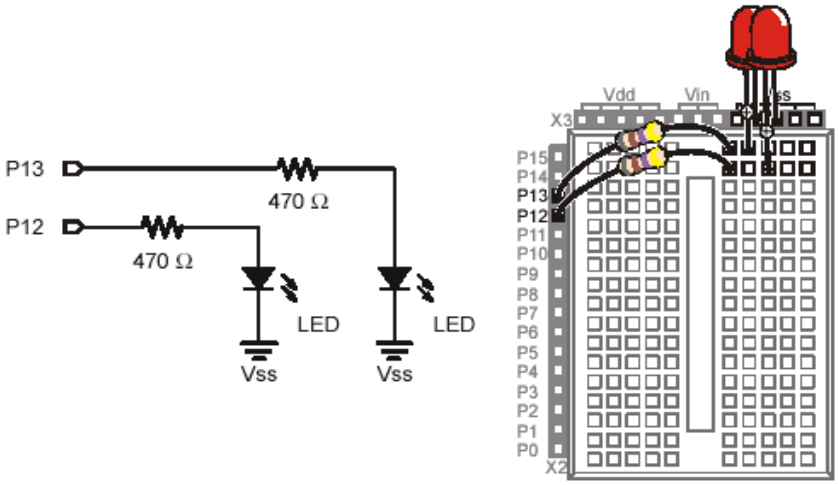


Figura 3-2.- *Circuito para medir el tiempo y repetir las mismas acciones. Utiliza dos LED, cuyos cátodos se conectan con tierra (Vss) y sus ánodos con las patitas de I/O de la Home Work a través de sendas resistencias.*

Los comandos **HIGH** y **LOW** sirven para sacar por las patitas de salida del microcontrolador un nivel alto (Vdd) y bajo (Vss), respectivamente. El parámetro Pin que sigue al comando identifica la patita de I/O (Entradas/Salidas) por la que sale el nivel lógico.

HIGH Pin
LOW Pin

Por ejemplo, si usas el comando ***HIGH 13*** le ordenas al módulo microcontrolador que saque por el pin de I/O nº 13 un nivel lógico alto, o sea la tensión positiva (Vdd), lo que conecta el ánodo del LED con el positivo y como el cátodo está permanentemente conectado a tierra, figura 3-2, el LED se ilumina.

Del mismo modo, si usas el comando ***LOW 13,*** sale por el pin 13 el voltaje Vss, lo que apaga el LED. Probémoslo.

Programa ejemplo: HighLowLed.bs2

- Teclea, guarda y ejecuta el programa HighLowLed.bs2.
- Comprueba que el LED conectado a P13 se enciende y se apaga una vez cada segundo.

```
' El Robot Home Boe-Bot - HighLowLed.bs2
' Activar/desactivar el led conectado en P13 cada segundo.

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

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

Haz parpadear el otro LED

Hacer parpadear el otro LED (conectado a P12) es tan simple como cambiar el parámetro ***Pin*** en los comandos **HIGH** y **LOW** y ejecutar de nuevo el programa.

- Modifica el programa del siguiente modo:

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

- Ejecuta el programa modificado y comprueba que el LED se enciende y se apaga.
- También puedes hacer que ambos LED parpadeen al mismo tiempo.
- Modifica el programa de la siguiente manera:

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

- Ejecuta el programa modificado y observa cómo ambos LED parpadean aproximadamente al mismo tiempo.

También puedes conseguir que cuando un LED esté encendido, el otro permanezca apagado ajustando en el comando **PAUSE** el parámetro ***Duración*** dándole valores mayores o menores.

- Pruébalo

Ver las señales de control del servo con un LED

Los niveles lógicos altos y bajos que genere el programa que controla el giro de los servos deben durar una cantidad muy precisa de tiempo. Esto es debido a que el circuito electrónico que gobierna a los servo miden la cantidad de tiempo que una señal permanece alta, y la usan como si se tratase de una instrucción para girar. Para controlar con precisión el giro del servo, el tiempo que esas señales permanecen altas tiene que ser mucho más preciso de lo que se puede conseguir utilizando los comandos **HIGH** y **PAUSE**. En el caso del comando **PAUSE** sólo se puede controlar su parámetro ***Duración*** con pasos de 1 ms. Hay un comando PBASIC diferente, llamado **PULSOUT,** que proporciona señales de nivel alto durante periodos de tiempo muy pequeños. En este caso, su parámetro ***Duración*** expresa valores expresados en unidades de "dos" millonésimas de segundo.

PULSOUT Pin, Duración

Puedes enviar una señal **HIGH** que encienda el LED P13 durante 2 µs con el siguiente comando:

PULSOUT 13, 1

El siguiente comando encenderá el LED durante 4 µs:

PULSOUT 13, 2

El comando que se propone seguidamente envía una señal alta durante el tiempo suficiente para que se puede llegar a observar:

PULSOUT 13, 65000

¿Durante cuánto tiempo permanece encendido el LED conectado en P13 cuando envías este pulso? Veamos. El tiempo que está encendido es 65000 veces 2 µs. Es decir:

$$Duración = 65000 \times 2\ \mu s = 130.000\ \mu s = 0.13\ s$$

lo cual es un tiempo bastante pequeño y equivale 13 centésimas de segundo.

Programa ejemplo: PulseP13Led.bs2

El diagrama de tiempos de la figura 3-3 muestra el tren de pulsos que estás a punto de enviar al LED con este nuevo programa. Esta vez la señal alta dura 0.13 segundos y la señal baja 2 segundos. Esta señal es 100 veces más lenta que la señal que necesita el servo para controlar su movimiento.

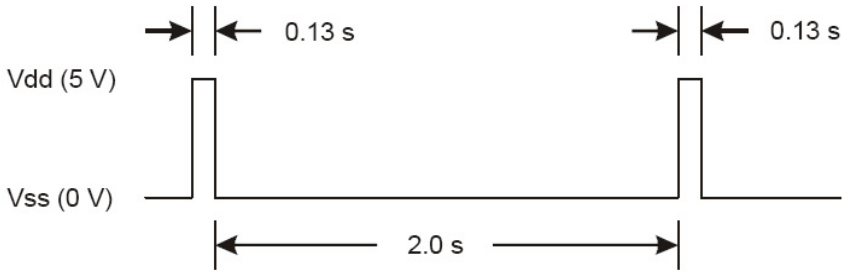


Figura 3-3.- *El nivel alto dura 0,13 segundos y el bajo 2 segundos.*

- Teclea, guarda y ejecuta el programa PulseP13Led.bs2.
- Comprueba que el circuito del LED conectado a P13 se enciende durante 13 centésimas de segundo cada dos segundos.

```
' El Robot Home Boe-Bot - PulseP13Led.bs2
' Enviar un pulso de 0.13s. cada 2 seg al led conectado en P13.

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

DO
  PULSOUT 13,65000
  PAUSE   2000
LOOP
```

Programa ejemplo: PulseBothLeds.bs2

Este programa envía un pulso al LED conectado a P13 y después envía otro al LED conectado a P12, tal y como se muestra en la figura 3-4. Después, hace una pausa de dos segundos.

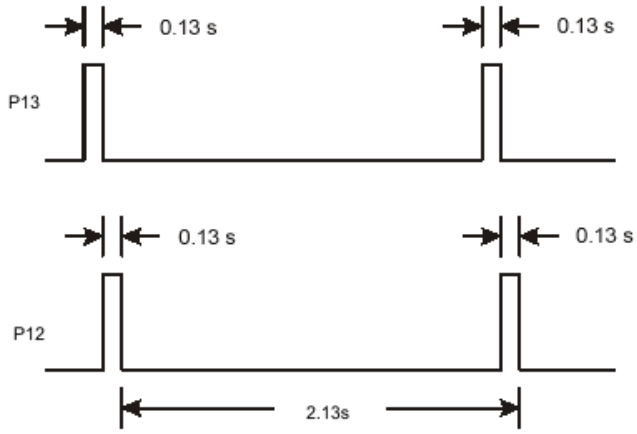


Figura 3-4. *Señales en las salidas P13 y P12*

- Teclea, guarda y ejecuta el programa PulseBothLeds.bs2.
- Comprueba que ambos LEDS emiten un pulso de luz simultáneamente durante unas trece centésimas de segundo cada dos segundos.

```
' El Robot Home Boe-Bot - PulseBothLeds.bs2
' Enviar un pulso de 0.13s. cada 2s a las salidas P12 y P13.

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

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

Comprobar la señal enviada al servo cuando gira a toda velocidad

Recuerda que la señal del servo debe ser 100 más rápida que la del programa que acabas de ejecutar. Primero probemos a ejecutar el programa diez veces más rápido. Para ello debemos dividir los parámetros ***Duración*** por 10 (**PULSOUT** y **PAUSE**).

- Modifica el programa así:

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

- Ejecuta el programa modificado y comprueba que hace que el LED parpadee 10 veces más rápido.

Ahora probemos a hacer que el parpadeo sea 100 veces más rápido. El LED parecerá que no parpadea y que se ilumina con cierta intensidad . Eso es debido a que el LED está encendiéndose y apagándose tan rápidamente que el ojo humano no puede detectar los cambios de estado, sólo del brillo promedio.

- Modifica el programa para que quede así:

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

- Ejecuta el programa modificado y observa que ambos LED aparecen iluminados aproximadamente con el mismo brillo.
- Prueba a sustituir por 850 el parámetro Duración del comando PULSOUT que se aplica a P13.

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

- Ejecuta el programa modificado y comprueba que el LED de P13 parece más brillante que el LED en P12. Posiblemente necesites tapar los LED con las manos para poder apreciar la diferencia, ya que la luz que emitan será muy tenue. El brillo de ambos es distinto porque el LED conectado a P13 permanece más tiempo encendido que el que está conectado a P12.
- Prueba a sustituir por 750 el parámetro Duración del comando PULSOUT que va a ambos LED.

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

- Ejecuta el programa y verifica que la intensidad con la que brillan ambos LED es la misma de nuevo.

3.4. EXPERIENCIA #3: CONEXIÓN DE LOS SERVOS

En esta práctica montarás los servos y los conectarás a sus pilas exclusivas de alimentación así como a las patitas de la Home Work por las que saldrán las señales que los gobernarán. Mientras los circuitos electrónicos de la tarjeta Home Work se alimentan con una pila de 9 V ubicada en la propia tarjeta, los servos precisan de una alimentación de 6 V procedentes de 4 pilas de 1,5 V que se alojan en un portapilas. Los servos consumen bastante intensidad por lo que se recomienda que sus pilas sean potentes y recargables.

Los componentes necesarios para esta experiencia son los siguientes (Figura 3-5):

(1) Portapilas con cables estañados
(2) Servos Parallax de rotación continua.
(2) Conectores macho de 3 pines.
(4) Cables
(4) Pilas alcalinas AA de 1'5 V. Se recomiendan especiales de buena intensidad (mA/h) y recargables.
(2) Diodos LED rojos
(2) Resistencias de 470 Ω (amarillo – violeta – marrón)

Nota: Mientras que los circuitos electrónicos montados en la tarjeta Home Work se alimentan con la pila de 9V ubicada en la propia tarjeta, los servos precisan de una alimentación de 6V procedentes de las 4 pilas de 1,5V que se insertan en el porta pilas. Los servos consumen bastante intensidad por lo que se recomiendan pilas potentes y recargables.

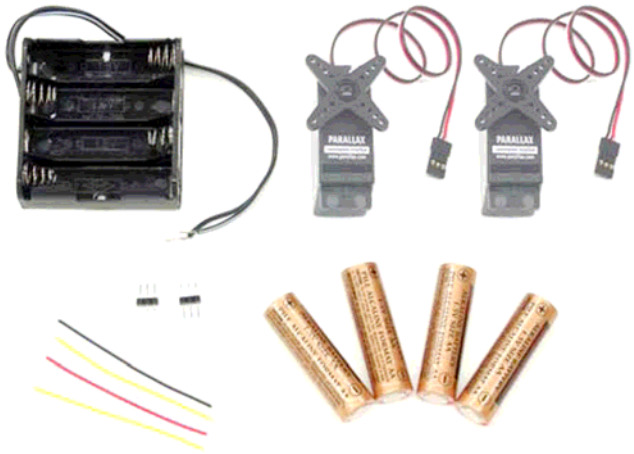


Figura 3-5.- *Materiales necesarios para la conexión de los servos.*

La figura 3-6 muestra el esquema de conexión de los servos. Antes de empezar a montar este circuito, asegúrate de que has desconectado la pila de 9 V de la Home Work.

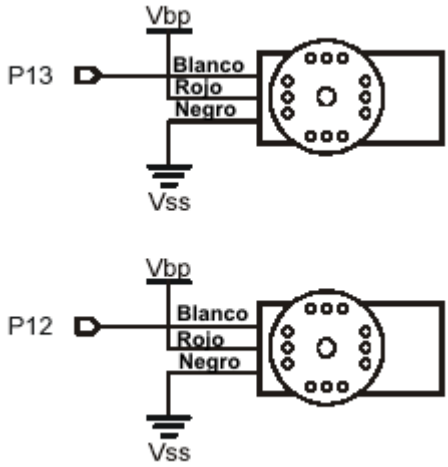


Figura 3-6.- *Conexionado de los tres cables de los servos (White:blanco, Red:rojo y Black:negro)*

- Desmonta el circuito anterior y guarda los componentes.
- Conecta los conectores hembra de los servos como muestra la parte izquierda de la figura 3-7.
- Asegúrate de que el cable negro está conectado a Vss y el rojo a Vbp.
- Asegúrate de que todas las conexiones de P13, Vbp, Vss, y P12 están tal y como muestra el diagrama de la figura.
- Conecta los conectores hembra de los servos a los conectores macho tal y como se muestra en la figura 3-7.

- Comprueba que los colores de los cables de los servos coinciden con lo que se muestra en la figura 3-7.

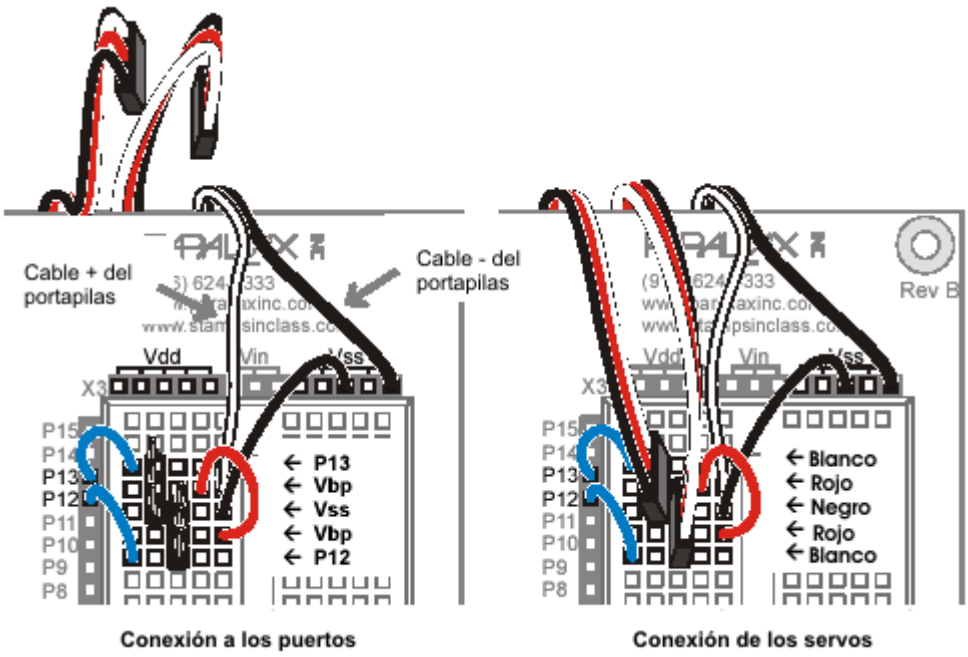


Figura 3-7. *Conexión de los servos*

El montaje debería quedar de forma similar al que aparece en la figura 3-8

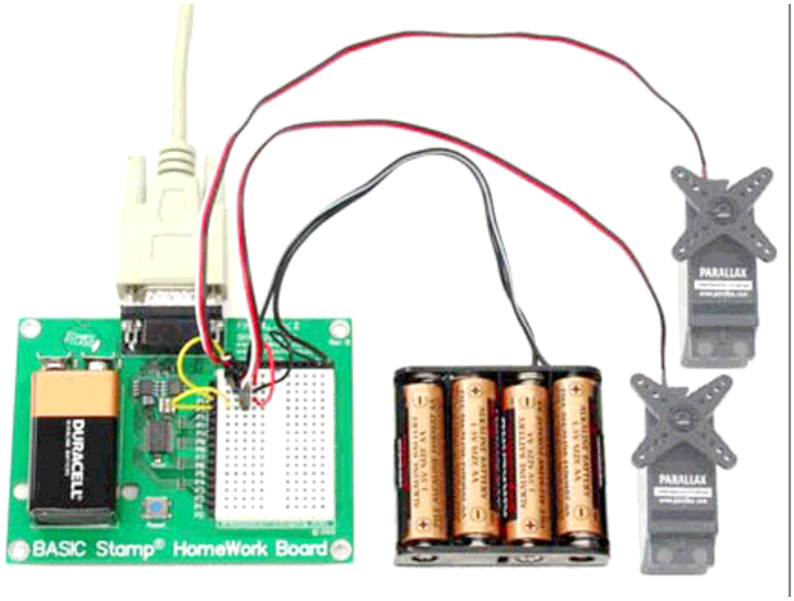


Figura 3-8.- *Aspecto final del conexionado de los servos. Durante el montaje no tener conectadas las pilas*

- Añade los LED y las resistencias al circuito para la visualización de las señales como se muestra en la figura 3-9, de esta forma las señales que se envíen a los servos también iluminarán los LED.

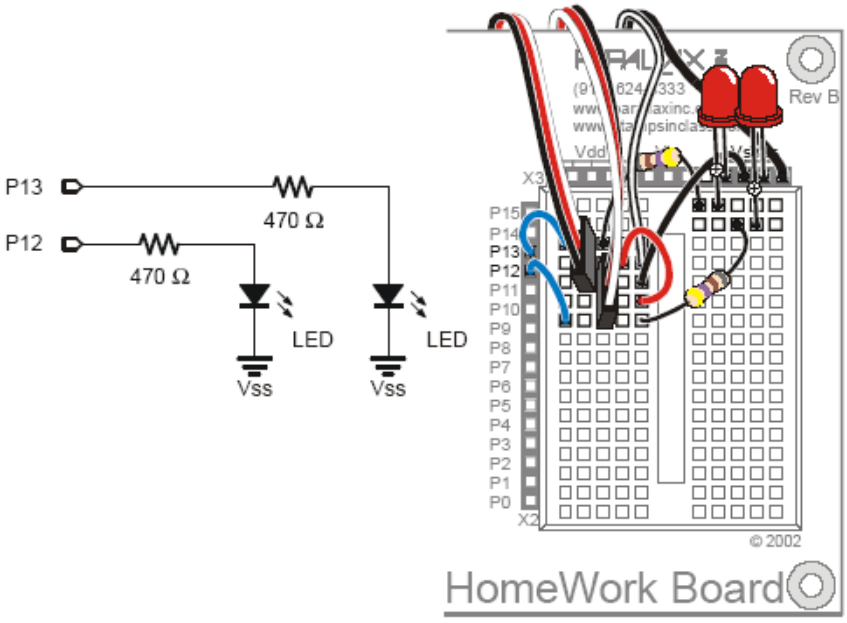


Figura 3-9.- *Se conectan dos LED con sus resistencias de absorción para visualizar las señales enviadas a los servos.*

- Cuando hayas hecho y comprobado todas las conexiones coloca las pilas del portapilas y conecta la de 9 V a la Home Work.

3.5. EXPERIENCIA #4: AJUSTANDO LOS SERVOS

En esta actividad ejecutarás un programa que enviará una señal a los servos que les obligará a que estén parados. Como los servos no vienen ajustados de fábrica, cuando se les aplica la señal de "paro" en vez de estar quietos tendrán un ligero movimiento, que habrá que eliminar.

La señal de ajuste

La figura 3-10 muestra la señal que se enviará para calibrar los servos. Dicha señal consiste en un pulso de 1'5 ms que se genera con el comando **PULSOUT** que utilizamos en prácticas anteriores para hacer parpadear los LED. ***Los servos vienen preparados para permanecer parados con un pulso de 1'5 ms***, así que pasaremos al comando **PULSOUT** el parámetro ***Duración*** con un valor de 750.

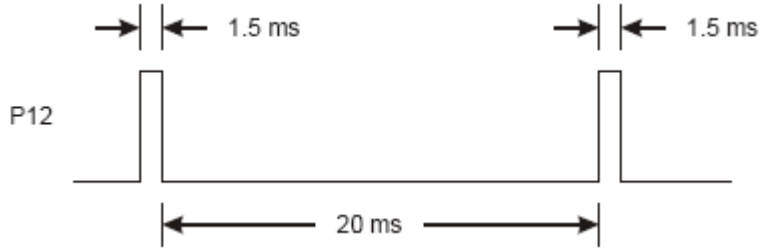


Figura 3-10.- *Señal necesaria para que los servos estén parados. Se usa en el ajuste inicial.*

Es conveniente que ajustes los servos de uno en uno. De ese modo podrás oír con claridad si uno de ellos se está moviendo ligeramente e identificarlo. El siguiente programa te permite ajustar el servo conectado a P12. Cuando termines repite el mismo procedimiento para el otro servo.

- Comprueba las conexiones a las baterías tanto de la Home Work como de los servos.

- Teclea, guarda y ejecuta el programa CenterServoP12.bs2

```
' El Robot Home Boe-Bot - CenterServoP12.bs2
' Este programa envía un pulso de 1.5mS al servo conectado a
' P12 para el centrado manual.
' {$STAMP BS2}
' {$PBASIC 2.5}
DO
  PULSOUT 12, 750
  PAUSE 20
LOOP
```

- Comprueba que el LED de monitorización conectado a P12 muestra actividad. Debería estar emitiendo luz, indicando que se están transmitiendo pulsos al servo conectado a P12.

Si el servo todavía no está bien ajustado su aspa fijada al eje comenzará a girar y podrás oír el motor interno haciendo ruido.

- Si el servo no está ajustado deberás usar un pequeño destornillador de cabeza plana para ajustar delicadamente su potenciómetro, tal y como se muestra en la figura 3-11. Ajusta el potenciómetro hasta que consigas que el servo deje de girar. Hazlo con mucho cuidado porque una imprudencia puede romper al servo.

Figura 3-11.- *Mueve cuidadosamente el potenciómetro del servo con un pequeño destornillador de cabeza plana hasta que deje de girar, cuando está recibiendo la señal de calibración.*

Ajusta el servo conectado a P13

- Repite el proceso anterior para el servo conectado a P13 utilizando el siguiente programa (CenterServoP13.bs2):

```
' El Robot Home Boe-Bot - CenterServoP13.bs2
' Este programa envía un pulso de 1.5mS al servo conectado a
' P13 para el centrado manual.
' {$STAMP BS2}
' {$PBASIC 2.5}
DO
  PULSOUT 13, 750
  PAUSE 20
LOOP
```

3.6. EXPERIENCIA #5: REGISTRANDO VALORES Y CONTANDO

En esta experiencia aprenderás lo que son las variables y a utilizarlas en los programas para almacenar valores. Todos los programas del Home Boe-Bot que aparecerán a partir de ahora utilizarán, en gran medida, las variables. Si, en los programas, somos capaces de guardar valores en variables, el siguiente paso será contar. Y desde el momento en que somos capaces de contar, podemos controlar el número de veces que se repiten las cosas.

Como ya hemos dicho, las variables sirven para almacenar valores, pero antes de utilizarlas hay que declararlas:

Nombre de la variable* VAR *tamaño

El siguiente ejemplo declara dos variables para su posterior utilización:

```
valor       VAR   Word
otro_valor  VAR   Word
```

Las variables que se declaran con tamaño Word admiten un valor máximo en decimal de 65.535. Una vez declarada una variable, ésta puede ser inicializada, es decir, se le puede asignar un valor inicial:

```
valor = 500
otro_valor = 2000
```

El símbolo "=" es un operador. Se pueden utilizar distintos operadores para realizar operaciones matemáticas y asignar valores a las variables, por ejemplo:

```
valor = 10 * valor
ancho = ancho * promedio
```

Programa ejemplo: VariablesAndSimpleMath.bs2

Este programa muestra la forma de declarar variables, inicializarlas y realizar algunas operaciones con ellas.

- Antes de ejecutar el programa intenta deducir cuál va a ser el comportamiento de cada comando **DEBUG**.
- Teclea, salva y ejecuta el programa VariablesAndSimpleMath.bs2.
- Compara los resultados con lo que habías previsto y explica las diferencias.

```
' El Robot Home Boe-Bot - VariablesAndSimpleMath.bs2
' Declara variables que se emplean en diversos cálculos matemáticos

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

valor     VAR     Word
otro_valor VAR    Word       'Declaración de variables

valor = 500
otro_valor = 2000             'Inicialización de variables

DEBUG ? valor
DEBUG ? otro_valor           'Visuliaza valores

valor = 10 * otro_valor       'Cálculo matemático
```

```
DEBUG ? valor
DEBUG ? otro_valor          'Visualiza resultados

END
```

Explicación del programa VariablesAndSimpleMath.bs2

Primero se declaran dos variables, **valor** y **otro_valor**.

```
valor       VAR Word
otro_valor  VAR Word
```

Después se asignan valores a ambas variables

```
valor = 500
otro_valor = 2000
```

Los comandos **DEBUG** ayudan a ver el valor de las variables después de su inicialización. Como **valor** ha recibido un valor de 500 y **otro_valor** de 2000, los comandos **DEBUG** envían los mensajes "valor = 500" y "otro_valor = 2000" al terminal.

```
DEBUG ? valor
DEBUG ? otro_valor
```

Con las siguientes tres líneas de código surge la pregunta ¿qué se mostrará en la pantalla del terminal? La respuesta es que como **valor** toma un valor diez veces **otro_valor** y como **otro_valor** sigue siendo 2000, **valor** mostrará 20,000 y **otro_valor** no cambiará.

```
valor = 10 * otro_valor
DEBUG ? valor
DEBUG ? otro_valor
```

Cálculos con números negativos

Para trabajar con números negativos, puedes utilizar el comando **DEBUG** en conjunción con el formateador **SDEC** para mostrarlos correctamente. Aquí tienes un ejemplo basado en el ejemplo VariablesAndSimpleMath.bs2.

- Elimina esta parte del programa VariablesAndSimpleMath.bs2:

```
valor = 10 * otro_valor
DEBUG ? valor
DEBUG ? otro_valor
```

- Reemplázala por la siguiente:

```
valor = valor – otro_valor
DEBUG "valor = ", SDEC valor, CR
DEBUG otro_valor
```

- Ejecuta el programa modificado y comprueba que **valor** cambia de 500 a -1500.

Contaje y control de repeticiones

La manera más acertada de controlar el número de veces que se ejecuta un fragmento de código de programa es utilizando el bucle **FOR…NEXT.** Su sintaxis es la siguiente:

FOR ***Contador*** **=** ***ValorInicial*** **TO** ***ValorFinal*** ***{*****STEP** ***ValorIncremento}*****…NEXT**

Los puntos suspensivos se sustituyen por el bloque de código de programa que se desea repetir. Asegúrate de declarar previamente unas variables con el nombre ***Contador, ValorInicial*** y ***ValorFinal*** que pueden ser tanto números como variables. Cada vez que veas algo entre llaves {} en la descripción de la sintaxis de una instrucción, quiere decir que es un parámetro opcional.

No es necesario que la variable se llame "contador". Por ejemplo puedes llamarla "miContador":

```
miContador     VAR     Word
```

He aquí un ejemplo de un bucle **FOR…NEXT** que usa **miContador** como variable para contar. Además muestra el valor que toma **miContador (myCounter)** en cada iteración del bucle.

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

Programa ejemplo: CountToTen.bs2

- Teclea, guarda y ejecuta CountToTen.bs2.

```
' El Robot Home Boe-Bot – CountToTen.bs2
' Empleo de una variable en un bucle FOR...NEXT.

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

myCounter VAR Word

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

DEBUG CR, "He acabado !"

END
```

Diferentes valores de inicio, final e incremento

Puedes utilizar distintos valores para los parámetros ***ValorInicial*** y ***ValorFinal***.

- Modifica el bucle **FOR…NEXT** de modo que quede así:

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

- Ejecuta el programa modificado. ¿Has observado que el módulo microcontrolador es capaz de decrementar el valor de la variable, es decir, contar hacia atrás?

¿Recuerdas el parámetro opcional **{STEP ValorIncremento}**? Se puede utilizar para hacer que el bucle cuente "por pasos". En vez de 9, 10, 11…, puedes hacer que cuente de dos en dos (9, 11, 13…) o de cinco en cinco (10, 15, 20…), o de tantos en tantos como quieras, especificando dicho valor que le des a **STEP**. Se propone un ejemplo de un programa que usa "pasos" de tres unidades:

- Añade **STEP 3** al comando **FOR…NEXT** para que quede de esta manera:

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

- Ejecuta el nuevo programa y comprueba que cuenta hacia atrás con intervalos de tres unidades.

3.7. EXPERIENCIA #6: COMPROBANDO LOS SERVOS

Aún queda una última tarea por realizar antes de montar los servos en el Home Boe-Bot, se trata de comprobar que responden correctamente. Para ello haremos un programa que envíe señales de giro a los servos en diferentes direcciones y a diferentes velocidades.

Control de la velocidad y la dirección mediante la duración del pulso

Recordemos que para calibrar los servos enviábamos una señal con una amplitud de pulso de 1.5 ms que hacía que permaneciesen en reposo. Lo conseguíamos usando el comando **PULSOUT** con el parámetro ***Duración*** con un valor de 750. ¿Qué pasaría si el valor enviado no fuera de 1.5 ms?

Ya se hizo un programa para enviar series de pulsos de 1.3 ms a un LED. Observemos con más detenimiento esas series de pulsos para ver cómo podríamos utilizarlas para controlar un servo. La figura 3-12 muestra cómo gira a toda velocidad un servo de rotación continua en el sentido de las agujas del reloj cuando le llegan pulsos de 1.3 ms. La máxima velocidad que se alcanza oscila entre las 50 y las 60 RPM (Revoluciones Por Minuto).

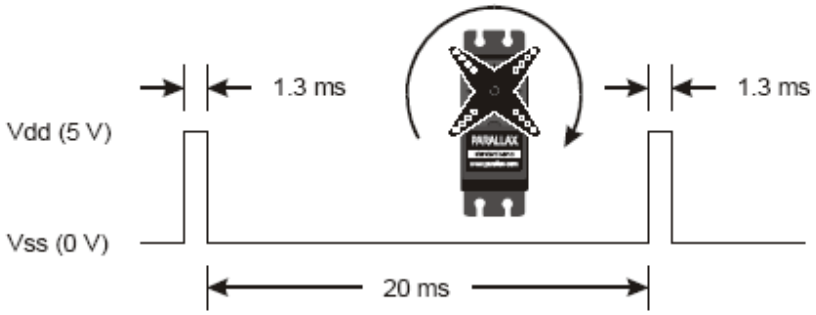


Figure 3-12.- *Señal para girar a toda velocidad en el sentido de las agujas del reloj.*

El circuito electrónico que gobierna al servo está preparado para que cuando reciba impulsos de 1,3 ms de duración cada 20 ms haga girar al eje del motor en el sentido de las agujas del reloj y a la máxima velocidad, que está comprendida entre 50 y 60 RPM. Cuando aumenta la duración del impulso de 1,3 a 1,5 ms la velocidad decrece que se para el eje del motor. Cuando se aplica al circuito electrónico del servo un impulso positivo de 1,7 ms de duración cada 20 ms, origina el giro del eje del motor en el sentido contrario a las agujas del reloj y a su máxima velocidad (50 – 60 RPM). Al reducir la duración del impulso de 1,7 a 1,5 ms decrece la velocidad de giro hasta pararse.

Puedes utilizar el programa ServoP13Clockwise.bs2 para enviar este tren de pulsos al servo conectado a P13.

Programa ejemplo: ServoP13Clockwise.bs2

- Teclea, guarda y ejecuta el programa ServoP13Clockwise.bs2.
- Comprueba que la cabeza del servo gira entre 50 y 60 RPM en el sentido de las agujas del reloj.

```
' El Robot Home Boe-Bot - ServoP13Clockwise.bs2
' Activa el servo P13 a velocidad máxima en sentido horario

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

DO
  PULSOUT 13, 650
  PAUSE 20
LOOP
```

Observa que un pulso de 1.3 ms necesita que en el comando **PULSOUT** especifiquemos un valor de 650 para el parámetro ***Duración***, que es menor que 750. Los pulsos menores que 1.5 ms y, por tanto, los valores de ***Duración*** menores que 750, harán que el servo gire en el sentido de las agujas del reloj.

Programa ejemplo: ServoP12Clockwise.bs2

Cambiando el parámetro ***Pin*** del comando **PULSOUT** de 13 a 12, el servo conectado a P12 girará a toda velocidad en el sentido de las agujas del reloj.

- Guarda ServoP13Clockwise.bs2 como ServoP12Clockwise.bs2.
- Cambia el parámetro ***Pin*** a 12.
- Ejecuta el programa y observa cómo ahora el servo conectado a P12 gira entre 50 y 60 RPM en el sentido de las agujas del reloj.

```
' El Robot Home Boe-Bot - ServoP12Clockwise.bs2
' Activa el servo P12 a velocidad máxima en sentido horario

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

DO
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

Programa ejemplo ServoP12Counterclockwise.bs2

Como seguramente habrás adivinado, si utilizamos un valor superior a 750 como parámetro ***Duración*** lograremos que el servo gire en sentido contrario. Por ejemplo, un valor de 850 enviará un pulso de 1.7 ms tal y como se muestra en la figura 3-13. En esta ocasión el servo girará a toda velocidad pero en el sentido contrario al del giro de las agujas del reloj.

Cuando se aplica al circuito electrónico del servo un impulso positivo de 1.7 mS se origina el giro del eje en el sentido contrario a las agujas del reloj a su máxima velocidad (50-60 RPM). Al reducir la duración del impulso de 1.7 a 1.5 mS, la velocidad disminuye hasta cero.

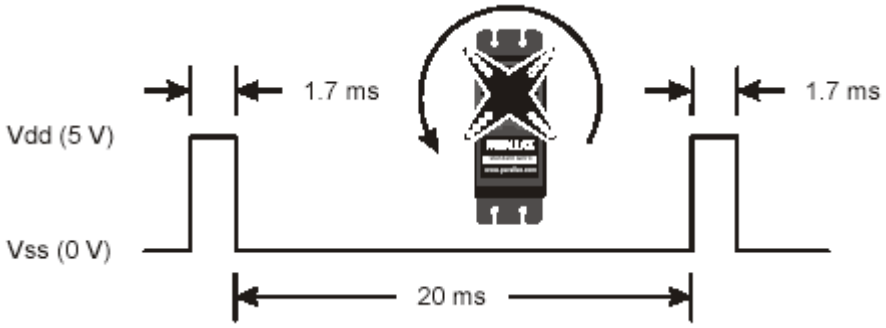


Figura 3-13.- *Señal necesaria para hacer girar el eje del servo a toda velocidad en sentido contrario a las agujas del reloj.*

- Guarda ServoP12Clockwise.bs2 como ServoP12Counterclockwise.bs2.
- Modifica el comando PULSOUT haciendo que el valor del parámetro Duración cambie de 650 a 850.
- Ejecuta el programa y comprueba que el servo conectado a P12 gira a una velocidad entre 50 y 60 RPM en sentido antihorario.

```
' El Robot Home Boe-Bot - ServoP12Counterclockwise.bs2
' Activa el servo P12 a velocidad máxima en sentido anti horario

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

DO
  PULSOUT 12, 850
  PAUSE 20
LOOP
```

Prueba de velocidad y la dirección de giro de los servos

- Modifica el comando **PULSOUT** para que el servo conectado a P13 gire en sentido antihorario.

Programa ejemplo: ServosP13CcwP12Cw.bs2

Se pueden utilizar dos comandos **PULSOUT** para hacer que ambos servos giren al mismo tiempo e incluso lo hagan en direcciones contrarias.

- Teclea, guarda y ejecuta el programa ServosP13CcwP12Cw.bs2.
- Comprueba que el servo conectado a P13 gira a la máxima velocidad en el sentido contrario al de las agujas del reloj, mientras que el que está conectado a P12 lo hace en sentido horario.

```
' El Robot Home Boe-Bot - ServosP13CcwP12Cw.bs2
' Mueve el servo P13 a velocidad máxima en sentido anti horario
' Mueve el servo P12 a velocidad máxima en sentido horario

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

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

El hecho de que los servos giren en sentidos opuestos es de gran importancia. Al encontrarse enfrentados ambos servos (simétricos), si giran en el mismo sentido sólo consiguen que el Home Boe-Bot dé vueltas sobre su propio eje. Para que avance en la misma dirección, cada uno tiene que girar en un sentido distinto y a la misma velocidad.

Ajustar la velocidad y la dirección

Hay cuatro combinaciones distintas para la instrucción **PULSOUT *Duración*** que se usarán ampliamente durante la programación del Home Boe-Bot en los próximos temas. ServosP13CcwP12Cw.bs2 envía una de esas combinaciones, 850 a P13 y 650 a P12. Prueba distintas combinaciones y rellena la columna "Descripción" de la tabla 3-1, así conseguirás familiarizarte con cada una de ellas y tener una tabla de referencia para consultar después. Cuando tengas montado el robot podrás observar qué hace con cada una de las combinaciones y rellenar la columna "Movimiento".

- Prueba las siguientes combinaciones de **PULSOUT *Duración*** y rellena la columna Descripción con los resultados que obtengas. La columna de Movimiento podrás rellenarla cuando esté montado el robot y veas hacia donde se mueve (adelante, atrás, giro, etc.).

DURACIÓN		Descripción	Movimiento
P13	P12		
850	650	Velocidad máxima. Servo P13 en sentido anti horario, servo P12 en sentido horario.	
650	850		
850	850		
650	650		
750	850		
650	750		
750	750	Ambos servos deben estar detenidos	
760	740		
770	730		
850	700		
800	650		

Tabla 3-1.- *Prueba las distintas combinaciones del parámetro Duración para los servos y anota los resultados en la columna Descripción. La de Movimiento podrás rellenarla cuando tengas montado el robot.*

FOR…NEXT para controlar el tiempo que gira el servo

Una vez conocida la forma de controlar la velocidad de giro y su dirección, lo único que queda es determinar el tiempo que permanece girando el servo. Para conseguirlo utilizaremos el bucle **FOR…NEXT**. He aquí un ejemplo de bucle **FOR…NEXT** que hace girar al servo durante unos segundos:

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

Calculemos la cantidad exacta de tiempo que el programa anterior mantiene al servo en movimiento. Cada vez que se recorre el bucle, el comando **PULSOUT** tarda 1.7 ms, el comando **PAUSE** dura 20 ms, y el bucle tarda unos 1.3 ms en ejecutarse.

Un ciclo a través del bucle = 1.7 ms + 20 ms + 1.3 ms = 23.0 ms. = 0,023 s.

Como el bucle se ejecuta 100 veces:

tiempo =100 × 0.023s = 2.30 s

Supongamos que queremos que el servo gire durante 4.6 segundos.. El bucle **FOR…NEXT** tendrá que ejecutarse el doble de veces:

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

Programa ejemplo: ControlServoRunTimes.bs2

- Teclea, guarda y ejecuta el programa ControlServoRunTimes.bs2.
- Comprueba que el servo conectado a P13 gira en sentido horario durante aproximadamente 2.3 segundos y a continuación el servo conectado a P12 gira el doble de tiempo

```
' El Robot Home Boe-Bot - ControlServoRunTimes.bs2
' Mover el servo P13 durante 2.3s a plena velocidad y en sentido
' anti horario. A continuación mover P12 el doble de tiempo

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

Si se desea que giren ambos servos, el conectado a P13 con un pulso de 850 y el conectado a P12 con un pulso de 650, cada repetición del bucle consumirá:

```
1.7ms  –    Servo conectado a P13
1.3 ms –    Servo conectado a P12
20 ms  –    Duración de la pausa
1.6 ms –    Ejecución del código
24.6 ms –   Total
```

Para hacer que giren durante un tiempo determinado, hay que calcular el equivalente así:

Número de pulsos = Tiempo (s) / 0.0246 s = Tiempo / 0.0246

Supongamos por ejemplo que queremos que giren durante 3 segundos:

Número de pulsos = 3 / 0.0246 = 122

Una vez calculado, puedes usar el valor 122 como ***ValorFinal*** en el bucle **FOR…NEXT**:

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

Programa ejemplo: BothServosThreeSeconds.bs2

- He aquí un ejemplo que hace que los servos giren en una dirección durante 3 segundos y luego inviertan la dirección de giro.
- Teclea, salva y ejecuta el programa BothServosThreeSeconds.bs2.

```
' El Robot Home Boe-Bot - BothServosThreeSeconds.bs2
' Mover ambos servos en direcciones opuestas durante 3s. A continuación
' cambiar el sentido de giro de ambos servos y moverlos otros 3s

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

Comprueba que cada servo gire en una dirección durante 3 segundos, invierte la dirección de giro y rota durante otros 3 segundos más.

Predecir el tiempo de giro del servo

- Elige un tiempo (en segundos) durante el cual quieras que giren los servos
- Divide el número de segundos por 0.024.
- El resultado es el número de repeticiones del bucle que se necesitan.
- Modifica BothServosThreeSeconds.bs2 de forma que ambos servos giren durante el tiempo que hayas decidido
- Compara el tiempo que has previsto con el que han girado realmente
- Recuerda desconectar las pilas del sistema (tarjeta y servos) cuando hayas terminado

3.8 PRUEBA DE AUTOEVALUACION

Preguntas

1. ¿Para qué se usan normalmente los servos estándar?
2. ¿En qué se diferencian los Servos de Rotación Continua Parallax de los servos estándar?
3. ¿Qué hace el comando **PAUSE**? ¿Qué hace **PAUSE 3000**? ¿Y **PAUSE 20**?
4. ¿Cuánto dura un milisegundo? ¿Cuál es su abreviatura?
5. ¿Qué comandos PBASIC puedes utilizar para conseguir ejecutar una y otra vez otros comandos PBASIC?
6. ¿Qué hace una resistencia en un circuito? ¿Para qué sirven las franjas de colores que hay en las resistencias?
7. ¿Qué significan las siglas LED? ¿Qué hace un LED en un circuito? ¿Cuáles son los dos extremos de un LED?
8. ¿Qué comando hace que la BASIC Stamp internamente conecte uno de sus pines I/O a Vdd? ¿Qué comando hace lo mismo pero a Vss?
9. ¿Qué comando se usa para enviar una señal alta durante un cierto número de milisegundos?
10. ¿Cuáles son los nombres de los distintos tamaños de variables que puede haber en un programa PBASIC? ¿Qué tamaño de valores puede albergar cada uno?
11. ¿Cuál es la principal diferencia entre los bucles **FOR…NEXT** y **DO…LOOP**?
12. ¿Cómo sabe un bucle **FOR…NEXT** dónde empezar a contar y dónde terminar?
13. ¿Con qué se controla la velocidad y sentido de giro de un servo? ¿Qué tiene esto que ver con los diagramas de tiempos? ¿Cuál es el comando y parámetro con el cual se puede controlar el sentido de giro y la velocidad de un servo?
14. ¿Cómo se consigue controlar el tiempo durante el cual gira un servo? ¿Qué comandos se necesitan para controlar el tiempo de giro?

Ejercicios

1. Escribe un comando **PAUSE** que haga permanecer en espera durante 10 segundos a la BASIC Stamp.
2. Dibuja el diagrama de tiempos de este **DO…LOOP**:

```
DO
        PULSOUT 15, 6500
        PULSOUT 14, 6500
        PAUSE 200
LOOP
```

3. Modifica este **FOR…NEXT** de tal modo que cuente de 6 a 24 de 3 en 3. Además escribe la declaración de variable que necesaria.

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

4. Modifica este **DO…LOOP** para conseguir que ambos servos giren en sentido antihorario.

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

5. Modifica la respuesta de la pregunta anterior para que los servos giren durante 6.5 segundos.

Proyectos

1. Dibuja el esquema de un circuito con un par de LED conectados a P14 y P15 y servos conectados a P12 y P13.
2. Escribe un programa que haga que el servo conectado a P12 gire en sentido antihorario durante dos segundos. Durante ese tiempo, el LED conectado a P14 deberá estar emitiendo luz. Después el programa no hará nada durante dos segundos y por fin hará que el servo de P13 gire en sentido horario mientras que el LED de P15 permanece encendido.
3. Escribe un programa que haga que el LED conectado a P14 parpadee mientras que el servo de P12 está girando.
4. Escribe un programa para que los servos pasen por tres de las cuatro combinaciones posibles de rotación. Necesitarás cuatro bucles **FOR…NEXT** distintos. Primero ambos servos deberán rotar en sentido antihorario, luego en sentido horario, después el de P12 en sentido horario mientras que el de P13 lo hará en sentido antihorario. Finalmente el servo de P12 girará en sentido antihorario y el de P13 en sentido horario.

Tema 4

Montaje y puesta en marcha del "Home Boe-Bot"

4.1. EL PLAN DE TRABAJO

En esta sección te proponemos montar la estructura del robot y probar el funcionamiento de tu Home Boe-Bot. Es especialmente importante completar todas las experiencias antes de pasar al siguiente tema. Sólo así podrás evitar un gran número de errores muy comunes que podrían condicionar el futuro comportamiento del Home Boe-Bot. El Índice de las actividades que tendrás que realizar en este tema es el siguiente:

ACTIVIDAD	DESCRIPCION
1	Montar el Home Boe-Bot.
2	Volver a probar los servos para asegurarte de que están correctamente ajustados y conectados.
3	Conectar y probar un zumbador sonoro que te permitirá saber cuando las baterías del Home Boe-Bot están bajas y hay que sustituir.
4	Dibujar las curvas de transferencia para tus servos, para saber la anchura de pulso que requiere cada velocidad de giro.

4.2. EXPERIENCIA #1: MONTAJE DEL HOME BOE-BOT

Se describen las operaciones que tendrás que ir realizando para montar la estructura y las ruedas del robot. En cada paso se precisan de ciertas piezas que tendrás que ir montándolas como se muestra en las fotografías, las cuales se complementan con sencillas aclaraciones. Síguelas tranquilamente y empezarás a disfrutar de tu trabajo y recoger las primeras satisfacciones de tu bestiecilla.

Las herramientas recomendables para el montaje se muestran en la figura 4-1 y son tres muy comunes.

(1) Destornillador telescópico de 1/8´´ (3'18 mm). Necesario
(1) Llave con cabeza mixta de M3 de paso (1/4"). Opcional
(1) Alicates de puntas. Opcional

Figura 4-1.- *Las tres herramientas recomendadas para el montaje del Home Boe-Bot*.

Comenzarás montando el chasis del robot para lo cual se precisan los componentes mostrados en la figura 4-2.

(1) Chasis del Home Boe-Bot
(4) Separadores de paso M3 x 20 mm.
(4) Tornillos de paso M3 x 10 mm
(1) Goma pasa muros de 15 x 7.5 mm

Las operaciones que debes realizar son las siguientes:

- Inserta la arandela de goma en el hueco del centro del chasis.
- Asegúrate que la muesca que está en el borde exterior de la arandela de goma coincide con el borde del hueco del chasis.
- Utiliza los 4 tornillos para colocar los separadores al chasis.

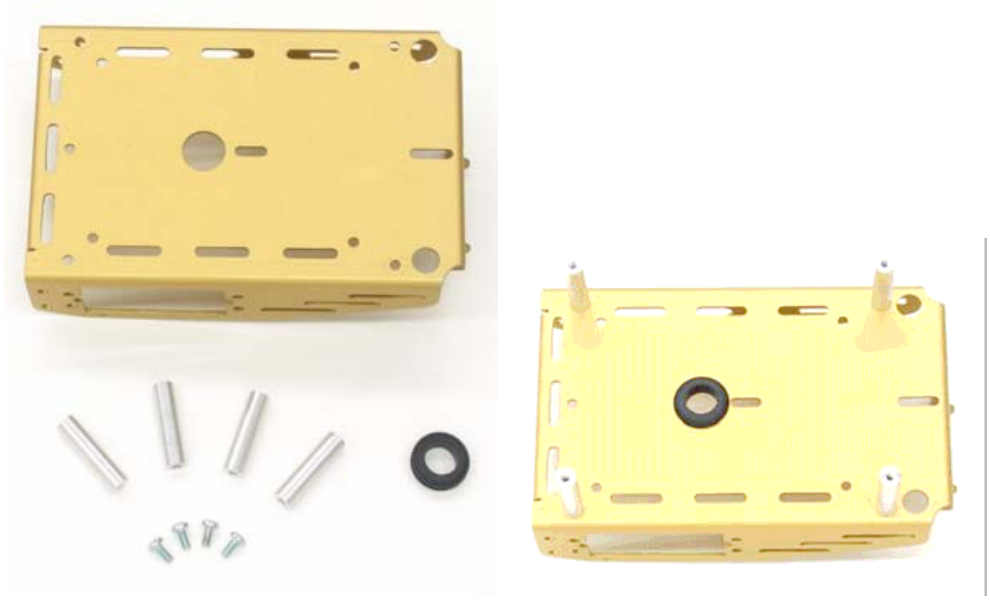


Figura 4-2.- *Chasis, separadores con sus tornillos y arandela de goma.*

Lo siguiente a montar son los servos en el chasis para lo cual debes comenzar llevando a cabo los siguientes pasos:

- Desconecta las baterías de alimentación de la Home Work y de los servos.
- Saca las pilas del portapilas.
- Desconecta los servos de la tarjeta Home Work .
- Utiliza el destornillador para soltar los tornillos que sujetan las aspas de los servos con los ejes.
- Tira hacia arriba de cada aspa hasta sacarlas de los ejes.
- Guarda los tornillos que sujetaban las aspas; se usarán en un paso posterior.

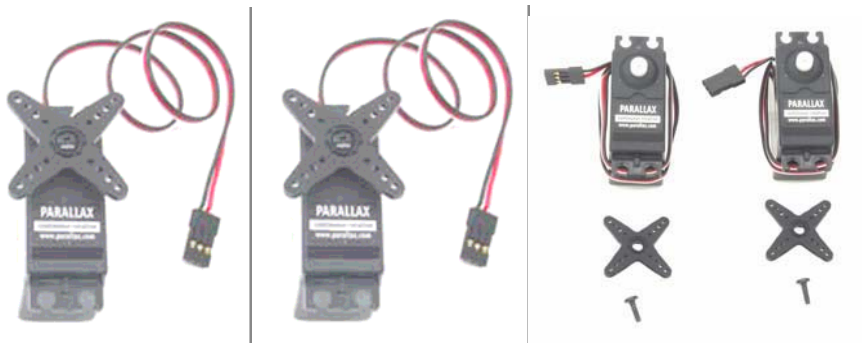


Figura 4-3.- *Los servos completos (izda.) y una vez quitadas sus aspas (dcha.).*

Para montar los servos en el chasis del robot se precisan :

(8) Tornillos de paso M3 x 6
(8) Tuercas M3

- Une los servos al chasis usando los tornillos y las tuercas. Ten en cuenta que para un mejor montaje tienes que colocar la cara de cada servo a través de la ventana rectangular del interior del chasis.
- Usa dos trozos de cinta aislante para etiquetar los servos como izquierdo (L) y como derecho (R).

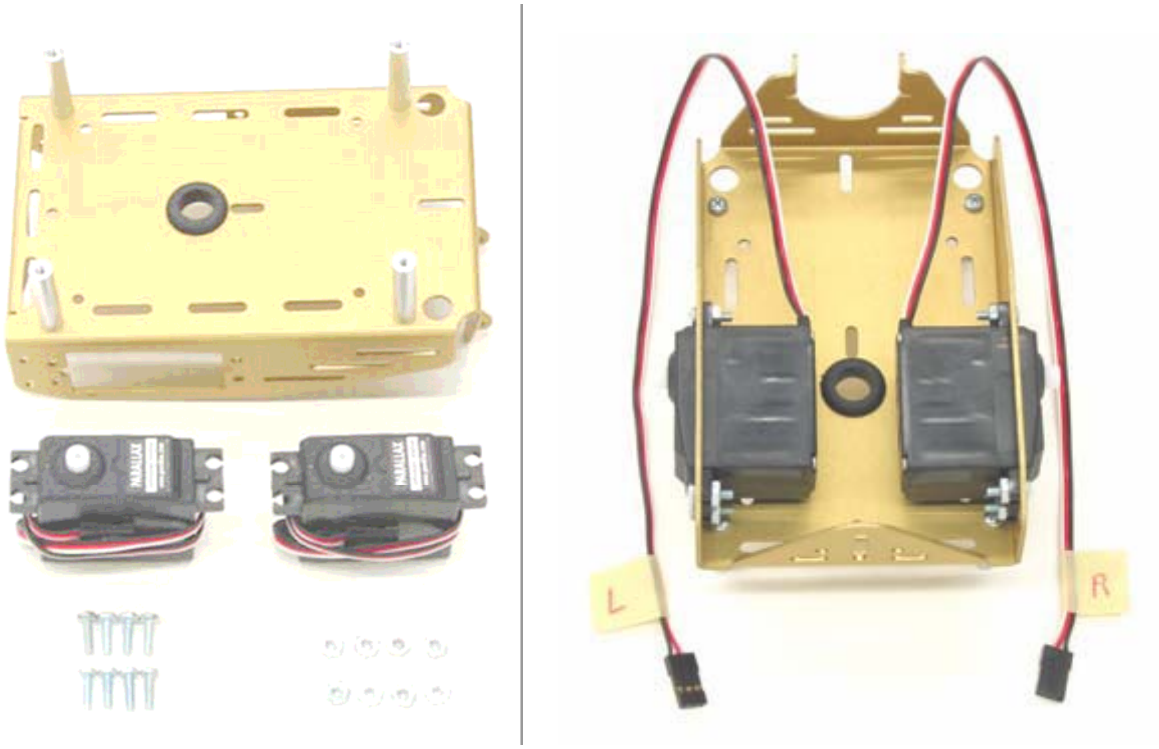


Figura 4-4.- *A la izquierda los materiales necesarios para sujetar los servos en el chasis. A la derecha fotografía de los servos una vez montados. El izquierdo lleva la etiqueta L y el derecho R.*

Para montar el porta pilas se precisan los componentes de la figura 4-5, entre los que hay dos tornillos de paso M3 x 10 con cabeza avellanada y sus correspondientes tuercas M3.

Figura 4-5.- *Materiales necesarios para sujetar el porta pilas que alimentará a los servos.*

Componentes

- Usa los tornillos y las tuercas para sujetar el portapilas a la parte inferior del chasis del Home Boe-Bot tal y como se muestra en la parte izquierda de la figura 4-6. Es posible que, con las puntas de unas tijeras, debas hacer un agujero en el porta pilas para que se pueda atornillar en el chasis.
- Asegúrate de insertar los tronillos a través del porta pilas, ajustando las tuercas en la parte superior del chasis.
- Según se muestra en la parte derecha de la figura 4-6, pasa el cable del portapilas por el agujero de la goma pasa muros colocada en el centro del chasis.
- Pasa los cables de los servo a través del mismo agujero.

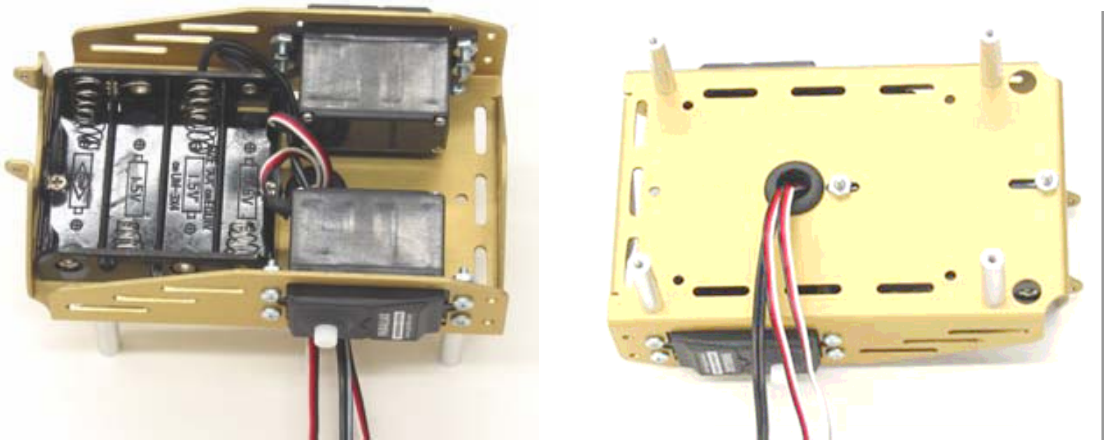


Figura 4-6.- *Aspecto del chasis con el portapilas montado y con sus cables y los de los servos pasando por la arandela de goma.*

La siguiente operación consistirá en colocar las ruedas con las que se moverá el Home Boe-Bot, para lo que se requiere:

(1) Home Boe-Bot parcialmente montado.
(1) Chaveta metálica
(1) Bola de plástico (rueda trasera)
(2) Gomas elásticas (neumáticos)
(2) Ruedas de plástico
(2) Los tornillos que se guardaron cuando se separaron las aspas de los servos

Figura 4-7.- *Materiales necesarios para colocar las ruedas del robot.*

La imagen izquierda de la figura 4-8 muestra la rueda trasera montada en el chasis. La rueda trasera es simplemente una bola de plástico con un agujero transversal atravesando su centro. La chaveta la mantiene unida al chasis y funciona como un eje para esa rueda trasera, que permitirá al robot tomar cualquier dirección determinada por el movimiento de las dos ruedas motrices delanteras.

- Alinea el agujero de la rueda trasera con los agujeros de la parte trasera del chasis.
- Pasa la chaveta a través de los 3 agujeros (izquierdo del chasis, rueda trasera, derecho del chasis).
- Dobla los extremos de la chaveta para evitar que se salga del agujero.

La imagen derecha de la figura 4-8 muestra las ruedas delanteras del Home Boe-Bot montadas en los servos.

- Estira las gomas elásticas y ajústalas a la superficie exterior de cada rueda.
- Cada rueda de plástico tiene un hueco que encaja con el eje exterior de los servos. Presiona cada una de las ruedas de plástico sobre el eje exterior del servo comprobando que el eje está alineado y encaja en el hueco.
- Utiliza los tornillos que habías guardado cuando retiraste las aspas para sujetar las ruedas delanteras al eje exterior de cada servo.

Figura 4-8.- *A la izquierda imagen de la rueda trasera. A la derecha las dos ruedas motrices delanteras acopladas a los ejes de los servos.*

Ahora toca montar la tarjeta Home Work de control sobre el chasis para lo cual nos harán falta 4 tornillos de cabeza redondeada de paso M3 x 10mm.

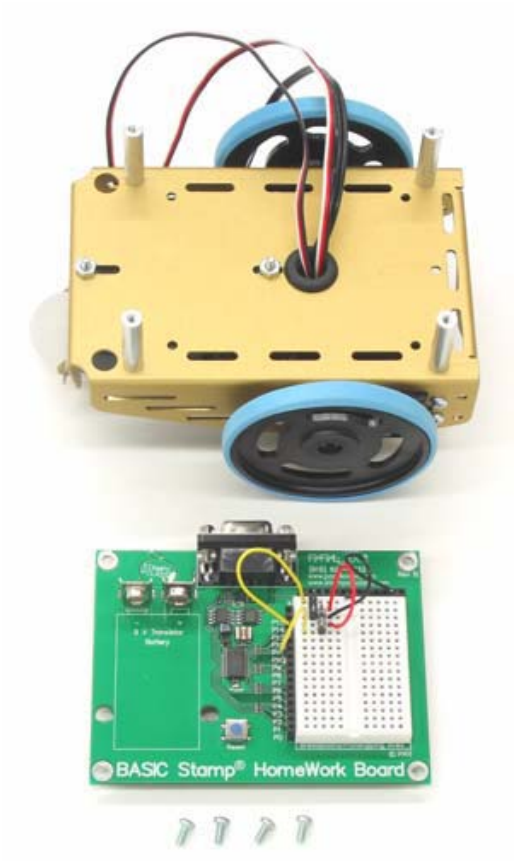


Figura 4-9.- *La tarjeta Home* Work *se sujeta sobre los separadores del chasis mediante 4 tornillos de M3 x10mm*

La figura 4-10 muestra los conectores de los servos conectados a la placa Homework.

- Conecta los terminales L y R de los servos a los conectores macho de la protoboard de la Home Work como se dispusieron en las últimas experiencias del tema anterior.
- Asegúrate de conectar el conector etiquetado como "L" a los cables procedentes de P13, y los etiquetados como "R" con los de P12.

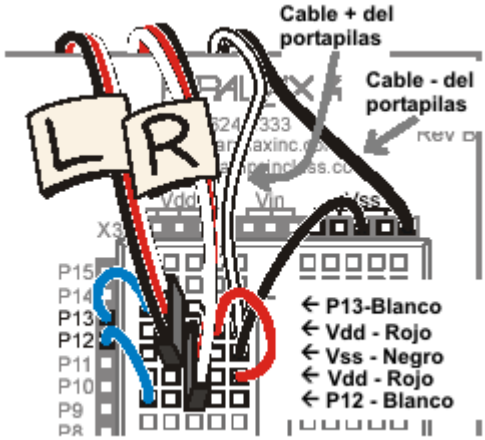


Figura 4-10.- *Conexión de los terminales de los servos con los conectores macho de la protoboard de la Home Work.*

Ten cuidado con la conexión de los cables de colores y recuerda que White es blanco, Red es rojo y Black es negro.

- Asegúrate de que la zona de montaje o protoboard de la tarjeta microcontroladora queda colocada más cerca de las ruedas delanteras que de la trasera.

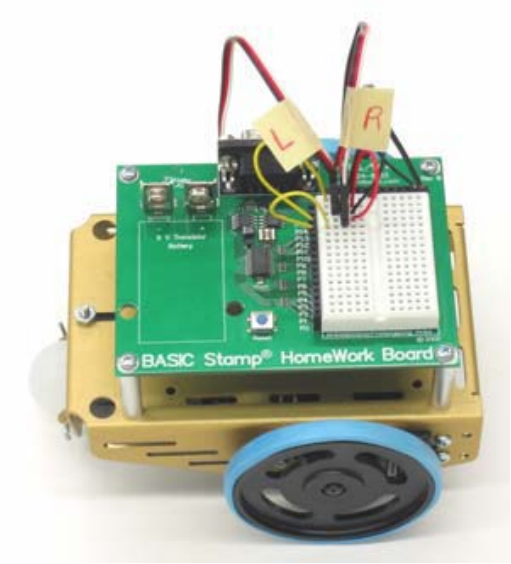


Figura43-11.- *Fotografía del robot con la Home Work montada y conectados los terminales de los servos.*

- Procura recoger los cables tanto los procedentes del porta pilas como los procedentes de los servos. Deben quedar entre el chasis y la parte inferior de la tarjeta de control Home Work, de forma que no sobresalgan ni se enreden con las ruedas.

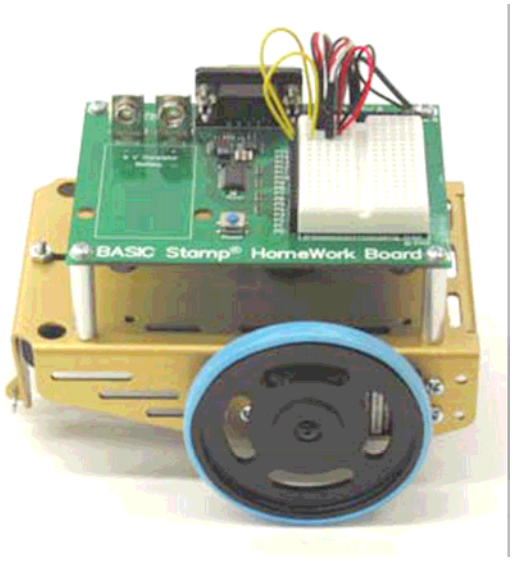


Figura 4-12.- *Una imagen del microbot.*

4.3. EXPERIENCIA #2: UNA NUEVA COMPROBACIÓN DE LOS SERVOS

Vas a comprobar que las conexiones eléctricas entre la tarjeta y los servos son correctas. Necesitamos asegurarnos que el servo de la derecha gira cuando recibe señales procedentes de P12 y que el servo de la izquierda gira cuando recibe señales de P13.

Comprobando la rueda derecha

El siguiente programa de ejemplo prueba el servo conectado a la rueda derecha (Figura 4-13). El programa hará que la rueda gire en la dirección de las agujas del reloj durante 3 segundos, parará durante un segundo y después, girará en sentido contrario a las agujas del reloj durante otros 3 segundos.

Figura 4-13.- *El programa de comprobación de la rueda derecha la hace girar 3 segundos en un sentido, luego se para un segundo y finalmente gira en sentido contrario otros tres segundos*.

Programa ejemplo: RightServoTest.bs2

- Conecta las pilas a la Home Work y a los servos.
- Teclea, salva y ejecuta el programa RightServoTest.bs2. que se presenta a continuación. Con el CD del kit se proporcionan la mayoría de los ejercicios evitando su edición.
- Verifica que la rueda derecha gira durante 3 segundos en la dirección de las agujas del reloj, para durante un segundo y gira en sentido contrario a las agujas del reloj durante otros 3 segundos.
- Si el servo derecho no se comporta como debe revisa los pasos anteriores, incluido el montaje y calibrado de los servos; si se comporta adecuadamente pasa a la siguiente sección

```
' El Robot Home Boe-Bot - RightServoTest.bs2
' El servo dcho. gira en sentido horario 3 s., se para 1 s.
' y vuelve a girar 3 s. en sentido anti horario.
' {$STAMP BS2}
' {$PBASIC 2.5}
counter VAR Word
FREQOUT 4, 2000, 3000      ' Señal de inicio/reset
FOR counter = 1 TO 122     ' Giro horario durante 3s.
  PULSOUT 12, 650
  PAUSE 20
NEXT
FOR counter = 1 TO 40      ' Parada 1 s.
  PULSOUT 12, 750
  PAUSE 20
NEXT
FOR counter = 1 TO 122     ' Giro anti horario durante 3s.
  PULSOUT 12, 850
  PAUSE 20
NEXT
END
```

Comprobando la rueda izquierda

Ahora debes hacer las mismas comprobaciones para la rueda izquierda tal y como se muestra en la figura 4-14. Hay que modificar el programa RightServoTest.bs2 de forma que los comandos PULSOUT sean enviados al servo conectado a P13 en lugar de a P12.

Figura 4-14.- *Efecto del programa para la comprobación de la rueda izquierda*

- Salva el programa RightServoTest.bs2 con el nuevo nombre LeftServoTest.bs2
- Cambia los 3 comandos **PULSOUT**, de forma que donde pone *PULSOUT 12* ponga *PULSOUT 13*.
- Salva y ejecuta el programa.
- Verifica que la rueda izquierda gira durante 3 segundos en la dirección de las agujas del reloj, para durante un segundo y que gira en sentido contrario a las agujas del reloj durante otros 3 segundos.
- Si el servo izquierdo no se comporta correctamente revisa los pasos anteriores, incluido el montaje y calibrado del servo; si se comporta adecuadamente pasa a la siguiente experiencia.

4.4. EXPERIENCIA #3: DETECTOR ACÚSTICO DE BAJA TENSIÓN Y RESET

Para que la circuitería electrónica de la tarjeta Home Work funcione correctamente es imprescindible que disponga de la suficiente tensión de alimentación. Eso supone que la tensión en Vin debe ser superior a 5,2 V, ya que si hay menos el regulador interno de la tarjeta sólo proporciona un voltaje inferior a 4,3 V, valor insuficiente que origina una anomalía denominada "brownout" y que hace que se ponga en marcha un mecanismo de autoprotección del sistema en el que tanto el procesador como la memoria pasan a un estado de congelación o reposo que detiene la ejecución de instrucciones. Al recuperar el voltaje correcto la Home Work se pone en marcha pero no donde había quedado en el programa sino desde el principio del mismo. Es decir, actúa como si se hubiese producido un Reset.

Cuando las baterías están bajas es posible que estas bajadas de tensión hagan reiniciarse al Home Boe-Bot cuando menos te lo esperas, lo cual puede hacer que el comportamiento del mismo no sea el esperado. Es decir, que tome direcciones erróneas y se mueva en las direcciones que no debe, de vueltas sobre si mismo...

Este ejercicio introduce un nuevo dispositivo llamado zumbador piezoeléctrico que puedes usar para generar diferentes tonos en función de la frecuencia de las señales que se envíen desde la Home Work.

Figura 4-15.- *Símbolo y aspecto del zumbador que emite un pitido de diferente tonalidad según la frecuencia que se le aplique.*

La figura 4-16 muestra el esquema de conexionado del terminal + del zumbador a la patita P4 de I/O, mientras que la figura 4-17 muestra el montaje de los componentes sobre la tarjeta.

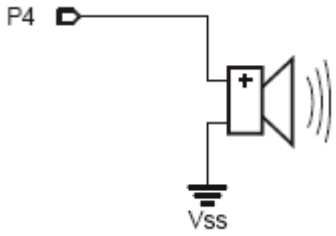


Figura 4-16.- *Conexión de un extremo (+) del zumbador a la patita P4 de la Home Work. El otro se manda a tierra.*

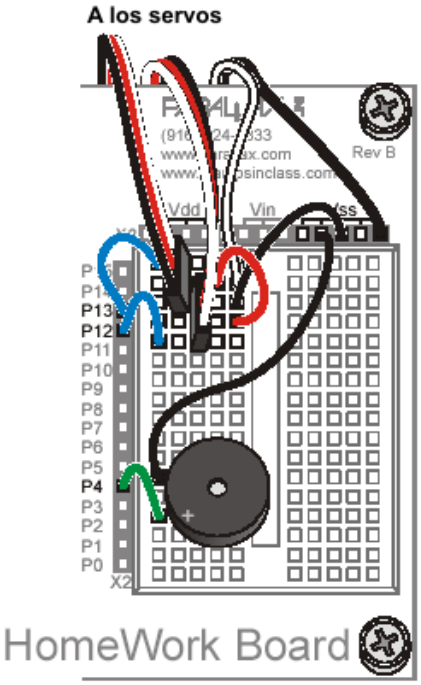


Figura 4-17.- *Montaje de los componentes del circuito sonoro con el zumbador.*

El siguiente programa produce un pitido sobre el zumbador. Utiliza el comando **FREQOUT** para enviar señales de la frecuencia que se desee por una patita de la Home Work. Su sintaxis es la siguiente:

FREQOUT Pin, Duracion, Frecuencia1 {, Frecuencia2}

Un ejemplo sería: FREQOUT 4, 2000, 3000. Envía una señal de 3000 herzios (3 KHz) durante 2000 ms (2 segundos) por el pin 4.

Programa ejemplo: StartResetIndicator.bs2

Este programa emite un pitido por el zumbador al iniciarse su ejecución y luego envía mensajes visualizadores de **DEBUG** cada medio segundo dentro de un bucle infinito. Se puede simular un "brownout" (bajada momentánea de voltaje de alimentación) presionando el reset o bien desconectando un instante la batería de tu Home Work; entonces el programa se reiniciará, emitiendo el pitido de nuevo. Cada vez que se produce el pitido en el zumbador significa que se inicia el programa desde el principio.

- Conecta la pila a tu placa.
- Teclea, salva y ejecuta el programa StartResetIndicator.bs2.

- Verifica que el zumbador emite un pitido durante 2 segundos antes de que comiencen a salir los mensajes de "Waiting for reset..." (Esperando un reset) en el Terminal de Depuración.
- Si no se oye ningún pitido comprueba las conexiones de los cables y el código.
- Si se oye el pitido puedes simular el estado de "brownout", primero presionando el botón de reset, y después desconectando y conectando la pila.

```
' El Robot Home Boe-Bot - StartResetIndicator.bs2
' Test del altavoz piezoeléctrico.
' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG CLS, "Beep!!!"            ' Visualiza mientras suena.
FREQOUT 4, 2000, 3000           ' Señal sonora.
DO                          ' Bucle DO...LOOP
  DEBUG CR, "Esperando al Reset.." ' Visualiza mensaje
  PAUSE 500                 ' cada 0.5 segundos
LOOP
```

Comportamiento del programa StartResetIndicator.bs2

Comienza mostrando el mensaje "Beep!!!" en cuanto se inicia la ejecución del programa. Inmediatamente envía una señal de 3 KHz al zumbador durante 2 segundos. Cómo las instrucciones son ejecutadas muy rápidamente por la Home Work da la sensación que el mensaje se presenta al mismo tiempo que el zumbador comienza a pitar.

Cuando termina de emitirse el pitido el programa entra en un bucle infinito mostrando una y otra vez el mensaje "Esperando al Reset..". (Esperando un reset). Cada vez que se produzca un reset, bien porque se aprieta dicho botón, bien porque se desconectan y se vuelven a conectar las baterías, el programa de reiniciará.

Añadiendo el StartResetIndicator.bs2 a otro programa

El programa anterior se usará de ahora en adelante en todos los programas. Considéralo parte de la "rutina de inicialización" de cada programa de tu Boe-Bot.

- Copia el comando **FREQOUT** del StartReserIndicator.bs2 en el HelloOnceEverySecond.bs2 encima de la sección **DO...LOOP**.
- Ejecuta el programa modificado y verifica que emite el pitido cada vez la Home Work es reseteada.

4.5. EXPERIENCIA #4: CURVAS DE TRANSFERENCIA DE LOS SERVOS

En esta práctica dibujarás las curvas que relacionan la velocidad de giro de los servos con la duración de los pulsos que se aplican desde la Home Work. Estas curvas te pueden resultar muy útiles porque cuando quieras obtener una velocidad en las ruedas gobernadas por los servos sólo tendrás que consultarlas para saber la anchura de los impulsos que debes aplicar a cada una. Usaremos el Panel de Transmisión del Terminal Debug para enviar valores a los programas que ejecuta la Home Work. Se teclean sobre dicho terminal y pasan al programa en ejecución. Sirve para poder introducir parámetros en todo momento a un programa.

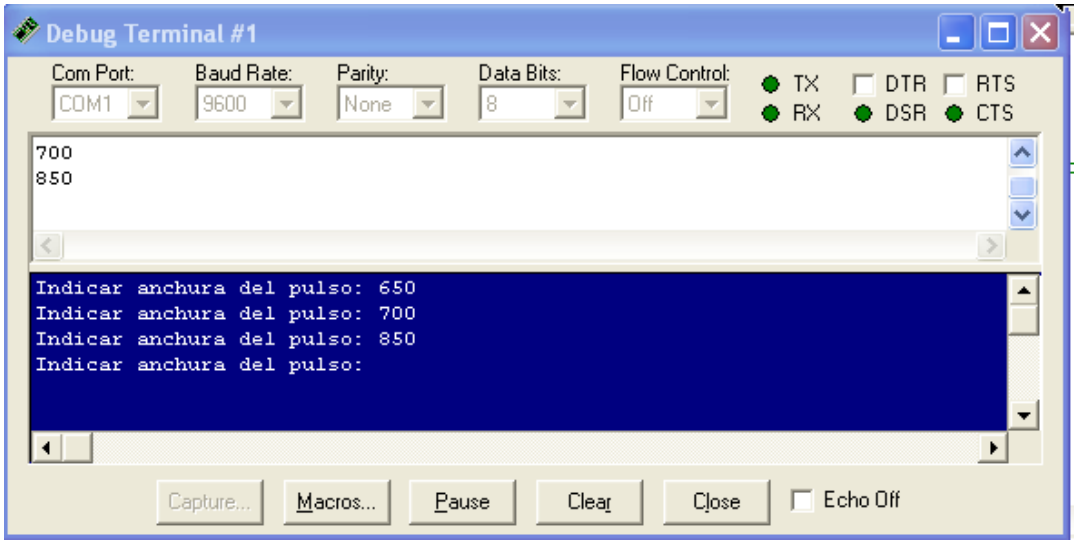


Figura 4-18.- *Sobre la ventana del Panel de Transmisión se pueden introducir valores que pasan al programa en ejecución.*

El comando DEBUGIN

Con el comando DEBUG se visualizan los mensajes que manda la Home Work al ejecutar el programa en la pantalla del PC. El comando DEBUGIN recoge el valor que se introduce con el teclado en una variable en el Panel de Transmisión y se envía al programa que está ejecutando la Home Work para que la variable quede fijada por dicho valor. Es decir, con DEBUGIN se introducen valores de variables que se usan en los programas de la Home Work.

En el siguiente programa de ejemplo la variable pulseWidth (anchura de pulso) almacena los valores que el comando **DEBUGIN** recibe. Evidentemente, habrá que declarar previamente esta variable en el programa:

PulseWidth VAR Word

El comando **DEBUGIN** captura los valores que introduzcas por el teclado a través del Panel de Transmisión y los almacena en la variable PulseWidth:

DEBUGIN DEC pulseWidth

En el siguiente ejemplo se utiliza la variable pulseWidth como el argumento ***Duración*** del comando **PULSOUT**.

PULSOUT 12, pulseWidth

Programa ejemplo: TestServoSpeed.bs2

- Teclea, salva y ejecuta el programa TestServoSpeed.bs2.
- Selecciona con el ratón el Panel de Recepción del Debug Terminal para activarlo.
- Introduce el valor 650 y presiona ENTER.
- Comprueba que el servo gira a toda velocidad en el sentido de las agujas del reloj durante 6 segundos.
- Cuando el servo haya parado, introduce el valor 850 y presiona ENTER.
- Comprueba que el servo gira a toda velocidad en el sentido contrario a las agujas del reloj.

```
' El Robot Home Boe-Bot - TestServoSpeed.bs2
' Introducir la anchura del pulso y contar nº de vueltas
' que gira la rueda durante 6 segundos.
' Multiplicando por 10 el nº de vueltas que gira conocemos
' 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 inicio/reset
DO
  DEBUG "Indicar anchura del pulso: "
  DEBUGIN DEC pulseWidth
  pulseWidthComp = 1500 - pulseWidth
  FOR counter = 1 TO 244
    PULSOUT 12, pulseWidth
    PULSOUT 4, pulseWidthComp
    PAUSE 20
  NEXT
LOOP
```

Cómo trabaja el programa TestServoSpeed.bs2

Se declaran tres variables: counter para el bucle **FOR...NEXT**, pulseWidth para los comandos **DEBUGIN** y **PULSOUT**, y pulseWidthComp, que almacena un valor que se usa en un segundo comando **PULSOUT** destinado al zumbador.

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

El comando **FREQOUT** se utiliza para indicar mediante un pitido en el zumbador que el programa se ha iniciado.

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

El resto del programa va dentro del bucle **DO...LOOP**, por lo que se ejecutará una y otra vez. El operador del DEBUGI pide que, al ejecutar DEBUGIN DEC pulseWidth, se introduzca un valor decimal que determinará la duración del pulso que se guardará en la variable pulseWidth.

```
DEBUG "Enter pulse width: "
DEBUGIN DEC pulseWidth
```

Para lograr una medición del tiempo más exacta se envían dos comandos **PULSOUT**, cuyos argumentos **Duración** sumarán 1500 entre los dos.

```
pulseWidthComp = 1500 - pulseWidth
```

Así consigues que el bucle **FOR...NEXT** tarde siempre el mismo tiempo en ejecutarse, y por tanto, que las mediciones de RPM que harás en el siguiente apartado sean más exactas.

El bucle **FOR...NEXT** envía pulsos al servo derecho (P12) durante 6 segundos. De esa forma las vueltas que se den en ese tiempo multiplicadas por 10 calculan las RPM. El valor contenido por pulseWidthComp es enviado al zumbador, y éste emite una serie de rápidos pitidos.

```
FOR counter = 1 TO 244
        PULSOUT 12, pulseWidth
        PULSOUT 4, pulseWidthComp
        PAUSE 20
NEXT
```

Dibujo de las curvas que relacionan la velocidad de los servos con la duración de los pulsos

La figura 4-19 muestra una curva de transferencia para un servo de transferencia continua. El eje horizontal mide la duración del pulso en milisegundos (ms), y el vertical la velocidad de giro en revoluciones por minuto (RPM). En dicha figura, el sentido de las agujas del reloj es el negativo, y el sentido contrario, el positivo. En dicha gráfica, los rangos de velocidad varían entre –48 y 48 RPM, y los de duración de los pulsos entre 1'3 y 1'7 ms.

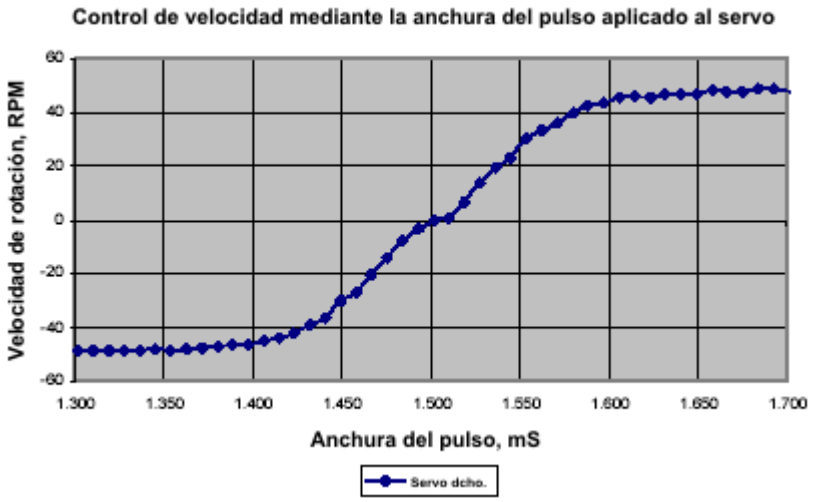


Figura 4-19.- *Gráfico que relaciona la velocidad de rotación del servo en RPM con la anchura de los pulsos que se le aplican.*

Puedes utilizar la tabla 4-1 para llevar a cabo tus propias mediciones.

Tabla 4.1 Anchura del pulso y velocidad RPM para el servo de Parallax							
Anchura de pulso (mS)	**Velocidad en RPM**	**Anchura de pulso (mS)**	**Velocidad en RPM**	**Anchura de pulso (mS)**	**Velocidad en RPM**	**Anchura de pulso (mS)**	**Velocidad en 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	

Tabla 4-1.- *Tabla para que rellenes el valor de las RPM del servo para las diferentes amplitudes de pulsos que se indican en la otra columna.*

Recuerda que el argumento **Duración** del comando **PULSOUT** se mide en unidades de 2μs .

$$Duration = 650 \times 2\ \mu s = 650 \times 0.000002\ s = 0.013\ s = 1.3\ ms$$

$$Duration = 655 \times 2\ \mu s = 655 \times 0.000002\ s = 0.0131\ s = 1.31\ ms$$

$$Duration = 660 \times 2\ \mu s = 660 \times 0.000002\ s = 0.0132\ s = 1.32\ ms$$

Figura 4-20.- *El valor del parámetro Duración viene medido en unidades de 2 microsegundos.*

- Haz una pequeña marca en la rueda para tener un punto de referencia.
- Ejecuta el programa TestServoSpeed.bs2.
- Carga el valor 650 en el Panel de Transmisión del Debug Terminal.
- Cuenta las revoluciones que da la rueda.

Ten en cuenta que el servo ha girado durante 6 segundos, por lo que si multiplicas este valor por 10, obtendrás el número de RPM.

- Escribe este valor (ya multiplicado por 10) en la tabla 4-1, junto a la celda de 1'3 ms
- Carga el valor 655.
- Cuenta las vueltas que da la rueda.
- Multiplica este valor por 10 y escríbelo junto al valor 1'31 ms de la tabla 4-1.
- Ve incrementando los valores de 5 en 5 (0'01 ms) hasta que llegues a 850 (1'7 ms).
- Repite el proceso para el otro servo.

4.6 PRUEBA DE AUTOEVALUACION

Cuestiones

1. ¿Qué es una condición de *brownout*?
2. ¿Cuáles son los síntomas de *brownout* en tu Home Boe-Bot?
3. ¿Cómo se puede utilizar el zumbador para detectar un *brownout*?
4. ¿Qué es un reset?
5. ¿Qué es una rutina de inicialización?
6. ¿Qué hace que un zumbador emita pitidos con diferentes tonos?
7. ¿Cuáles son los argumentos del comando **FREQOUT**? ¿Qué hace cada uno de ellos?
8. ¿Qué posibles errores se pueden dar al desconectar y volver a conectar los servos?
9. ¿Qué comando debes cambiar en el RightServoTest.bs2 para comprobar la rueda izquierda en vez de la derecha?
10. ¿Qué crees que ocurrirá si intentas enviar un dato utilizando el Panel de Transmisión si el comando **DEBUGIN** no está siendo ejecutado?
11. ¿Qué crees que ocurrirá si la Home Work ejecuta un **DEBUGIN** pero no se ha enviado ningún dato desde el Panel de Transmisión?

Ejercicios

1. Escribe un comando **FREQOUT** que haga que el zumbador emita un pitido distinto al del inicio de programa para identificar el fin del programa.
2. Escribe un comando **FREQOUT** que haga que el zumbador emita un pitido distinto a los de inicio y final de programa para indicar un paso intermedio. Prueba los siguientes valores: 100 ms de duración y 4kHz de frecuencia.
3. Declara una variable y mediante el comando **DEBUGIN** carga un valor en dicha variable.

Proyectos

1. Modifica el programa RightServoTest.bs2 para que emita un pitido indicando que el test se ha completado.
2. Modifica el programa RightServoTest.bs2 para que emita un pitido indicando cada vez que sale del bucle **FOR...NEXT**.
3. Modifica el programa TestServoSpeed.bs2 para que, utilizando el **DEBUGIN**, puedas introducir la duración de los pulsos para ambos servos, así como el número de repeticiones del bucle **FOR...NEXT**.
4. *Proyecto Avanzado* – Usando las curvas de transferencia, deduce qué argumentos de **PULSOUT Duración** harán que el servo izquierdo vaya a una velocidad determinada y el derecho a la misma velocidad en dirección contraria. La figura 4-21 muestra un ejemplo de cómo el servo izquierdo gira a 40 RPM en la dirección de las agujas del reloj, mientras que el derecho lo hace en la dirección contraria (a la misma velocidad, 40 RPM). Puedes multiplicar cada valor por 500 para obtener los argumentos de **PULSOUT Duración**.

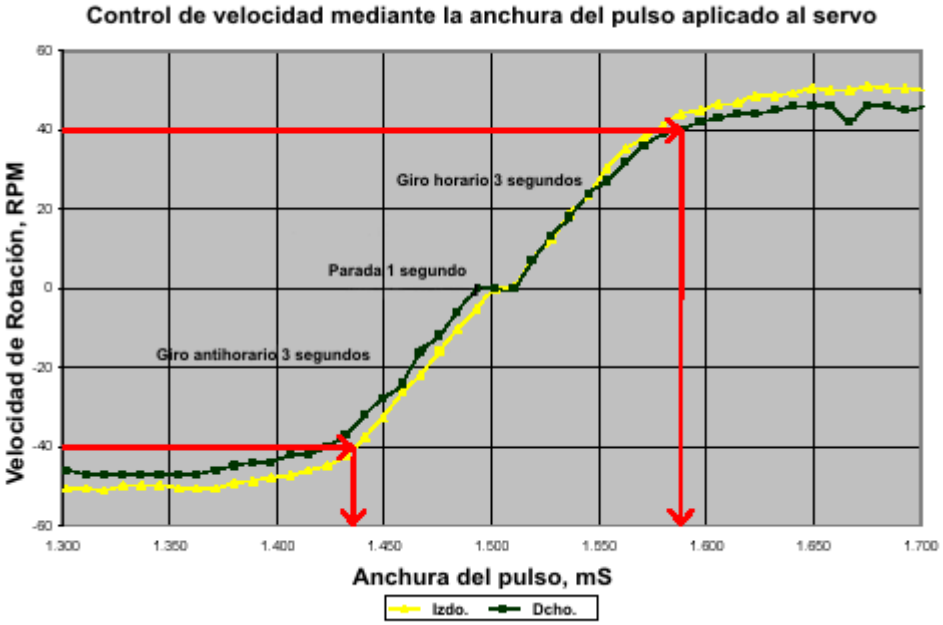


Figura 4-21.- *Los dos puntos mostrados en la gráfica indican cuando el servo izquierdo gira a 40 RPM en el sentido de las agujas del reloj y el derecho a 40 RPM en sentido contrario.*

Tema 5

Enseñando a moverse al Home Boe-Bot

5.1. EXPERIENCIA #1: MANIOBRAS BASICAS DEL HOME BOE-BOT

En este tema vamos a aprender a gobernar los movimientos fundamentales del robot: traslaciones hacia delante y atrás, giros y rotaciones. Sin embargo el Home Boe-Bot caminará "a ciegas" porque de momento no tiene sensores que le informen del entorno que le rodea, pero en los próximos temas iremos añadiendo diversos tipos de sensores que le proporcionen la información necesaria para esquivar obstáculos o seguir direcciones marcadas en el suelo.

La figura 5-1 muestra los distintos caminos que puede tomar nuestro amigo: Forward: adelante; backward: atrás; left turn: a la izquierda y right turn: a la derecha. Cuando el Home Boe-Bot va hacia delante su rueda derecha gira en el sentido de las agujas del reloj, mientras que la izquierda lo hace en sentido contrario (Figura 5-2).

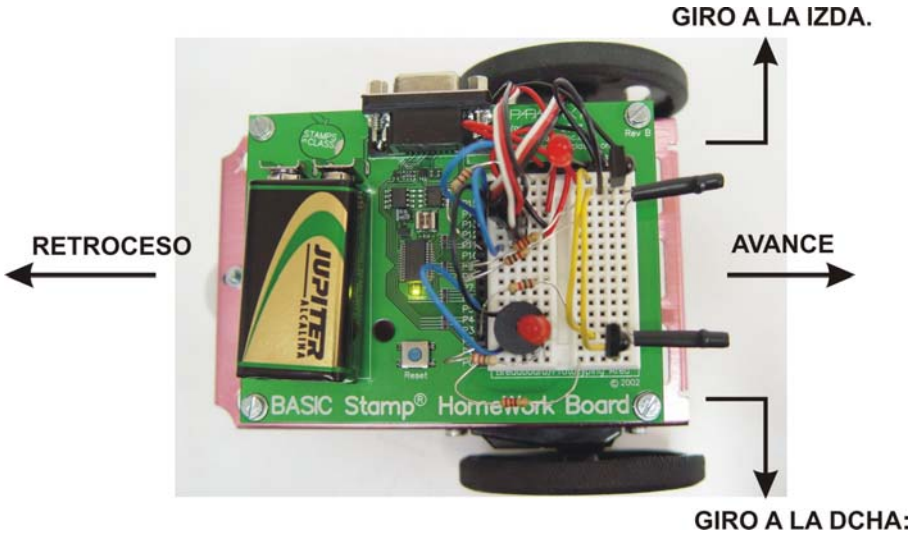


Figura 5-1.- *Los cuatro movimientos básicos: adelante, atrás, derecha e izquierda.*

Figura 5-2.- *Movimientos contrarios de las ruedas izquierda y derecha cuando avanza el robot.*

Programa BoeBotForwardThreeSeconds.bs2

Con este programa el Home Boe-Bot avanzará en línea recta durante tres segundos.

Recuerda que el argumento **Duración** del comando **PULSOUT** controla la velocidad y la dirección de los servos. Los argumentos **StartValue** y **EndValue** de un bucle **FOR...NEXT** controlan el número de pulsos que son enviados. Si cada pulso tiene la misma duración, el argumento **EndValue** también controla el tiempo que funcionan los servos. Se propone un programa que hará que el robot se mueva hacia delante unos tres segundos.

- Asegúrate de conectar las pilas
- Teclea, guarda y ejecuta el programa BoeBotForwardThreeSeconds.bs2

```
' El Robot Home Boe-Bot - BoeBotForwardThreeSeconds.bs2
' Conseguir que el robot avance durante 3 seg.
' {$STAMP BS2}
' {$PBASIC 2.5}

counter VAR Word
FREQOUT 4, 2000, 3000          ' Señal de inicio/reset.
FOR counter = 1 TO 122
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
END
```

Controlando la distancia y la velocidad

- Si cambias el valor **EndValue** del bucle **FOR…NEXT** de 122 a 61 harás que el Home Boe-Bot se desplace hacia delante en la mitad de tiempo, y que se mueva por tanto la mitad de distancia.
- Guarda el programa BoeBotForwardThreeSeconds.bs2 con otro nombre.
- Cambia el **EndValue** del bucle **FOR…NEXT** de 122 a 61.
- Ejecuta el programa y comprueba que el robot recorre la mitad de la distancia y durante la mitad de tiempo.
- Repite los pasos anteriores, pero en este caso cambia el valor **EndValue** a 244.

Si se ajusta el parámetro **Duración** de **PULSOUT** cerca de 650 o 850 los servos girarán a su máxima velocidad Si estos valores se aproximan a 750 harás que se muevan más lentos.

- Modifica en tu programa con los siguientes comandos:

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

- Ejecuta el programa y comprueba que el Boe-Bot avanza más lento.

Mover el Home Boe-Bot hacia atrás, rotarlo y "pivotarlo".

Todos estos movimientos se realizan modificando los argumentos **PULSOUT Duración**. Por ejemplo estas dos instrucciones **PULSOUT** hacen que el robot vaya hacia atrás:

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

Estas dos para que el Home Boe-Bot gire a la izquierda:

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

Y estas dos a la derecha:

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

Puedes combinar ahora estos comandos para que el Home Boe-Bot vaya hacia delante, detrás o gire en cualquiera de los sentidos, como se propone en el siguiente programa.

Programa ForwardLeftRightBackward.bs2

- Teclea, guarda y ejecuta el programa ForwardLeftRightBackward.bs2

```
' El Robot Home Boe-Bot - ForwardLeftRightBackward.bs2
' Conseguir que el robot realice los cuatro movimientos básicos.
' {$STAMP BS2}
' {$PBASIC 2.5}
counter VAR Word

FREQOUT 4, 2000, 3000                'Señal de inicio/reset
FOR counter = 1 TO 64                ' Avance
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT

PAUSE 200                            'Espera
FOR counter = 1 TO 24                ' Rotación izda 1/4 de vuelta
  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20
NEXT

PAUSE 200                            'Espera
FOR counter = 1 TO 24                ' Rotación dcha 1/4 de vuelta
  PULSOUT 13, 850
  PULSOUT 12, 850
  PAUSE 20
NEXT

PAUSE 200                            ' Espera
FOR counter = 1 TO 64                ' Retroceso
  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20
NEXT
END
```

Haciendo pivotar al robot

Puedes hacer que el Home Boe-Bot pivote sobre una rueda. Para ello hay que lograr que sólo gire una rueda mientras la otra se queda quieta. Por ejemplo si quieres que la rueda izquierda permanezca quieta y la derecha gire hacia delante para que el robot pivote deberás utilizar las siguientes líneas de código:

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

Si por lo contrario quieres que pivote hacia delante y a la derecha, simplemente detén la rueda derecha y haz que la rueda izquierda gire hacia delante.

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

Los siguientes comandos **PULSOUT** consiguen que el robot pivote hacia atrás y a la derecha

PULSOUT 13, 650
PULSOUT 12, 750

Finalmente, para que pivote hacia atrás y la izquierda se emplea:

PULSOUT 13, 750
PULSOUT 12, 850

- Guarda el programa ForwardBackwardLeftRight.bs2 como PivotTests.bs2.
- Sustituye los comandos **PULSOUT** que acabamos de mencionar en el mismo sitio donde estaban las rutinas de ir hacia delante, derecha, izquierda y marcha atrás.
- Ajusta el tiempo de ejecución de cada maniobra cambiando el valor **EndValue** de cada **FOR...NEXT** a 30.
- Ejecuta el nuevo programa generado.

5.2. EXPERIENCIA #2: RETOCANDO LAS MANIOBRAS BÁSICAS

Cuando el Home Boe-Bot está programado para avanzar en línea recta es frecuente que se vaya desviando ligeramente hacia uno de los lados. Se puede conseguir mantener la trayectoria mediante software. Comienza modificando el programa BoeBotForwardThreeSeconds.bs2 para que en vez de 3 segundos, se mueva durante 10, tiempo necesario para comprobar si se desvía o no. Para ello hay que cambiar el valor de **EndValue** del FOR Counter de 122 a 407, quedando el programa como se presenta a continuación. (Guárdalo como Boe-BotForwardTenSeconds.bs2).

```
' El Robot Home Boe-Bot - BoeBotForwardTenSeconds.bs2
' Conseguir que el Robot avance durante 10 seg.
' {$STAMP BS2}
' {$PBASIC 2.5}
counter VAR Word

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

FOR counter = 1 TO 407       ' Número de pulsos = tiempo de ejecución.
  PULSOUT 13, 850            ' Servo izdo.sentido antihorario máxima velocidad
  PULSOUT 12, 650             ' Servo dcho.sentido horario máxima velocidad
  PAUSE 20
NEXT
END
```

Ejecuta el programa y comprueba si el robot se desvía o no.

Ajuste de la velocidad de los servos para que el Boe-Bot siga una trayectoria recta.

Vamos a suponer que el Home Boe-Bot se desvía hacia la izquierda. Hay dos posibilidades. Puede ser porque la rueda izquierda gire más lenta o porque la rueda derecha gire más rápida. Como el Home Boe-Bot está configurado para que vaya a su máxima velocidad, lo más lógico sería pensar que la rueda derecha debe girar un poco más despacio.

Recuerda que la velocidad de los servos está determinada por el argumento **Duración** del comando **PULSOUT**. Cuanto más se acerque este valor a 750, más lento girará el servo. Esto quiere decir que hay que cambiar el valor 650 del comando **PULSOUT** 12 a un valor más cercano a 750. Si el robot no se desvía demasiado con un valor de 663 sería suficiente. Si por el contrario el grado de desvío es grande habrá que poner como mínimo un valor de 690. Para poder regular bien el Home Boe-Bot será necesario que hagas varias pruebas hasta alcanzar el objetivo deseado.

Una vez se ha controlado el movimiento hacia delante hay que conseguir que el movimiento hacia atrás tampoco produzca desviaciones de la trayectoria. Aplica el método que se acaba de describir para la marcha atrás y utiliza el programa BoeBotForwardTenSeconds.bs2 como referencia.

Ajustando los giros.

Es posible también que regules los giros del Home Boe-Bot mediante software. El tiempo que se pasa el Boe-Bot girando es el que determina el ángulo de giro. Como el que controla el tiempo de giro en el programa es el bucle **FOR…NEXT**, lo que hay que hacer es modificar el argumento **EndValue** para que gire más o menos según el caso. Aquí tienes la rutina de giro a la izquierda sacada de ForwardLeftRightBakcward.bs2.

```
FOR counter = 1 TO 24 ' Rotación izda ¼ de vuelta
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
NEXT
```

Vamos a suponer que el robot gira un poco más de 90º. Prueba primero a modificar el contador del bucle FOR Counter = 1 TO 23 ó 1 TO 22. Si sigue sin girar los 90º exactos, modifica el argumento de **PULSOUT** a un valor más cercano al 750, como has hecho cuando iba recto. Repite al acción hasta que gire los 90º exactos.

Confecciona un programa que haga girar al robot exactamente 90º. Una vez que lo consigas guárdalo como ForwardLeftRightBackward.bs2. Te dirá si deseas sobrescribir el fichero, a lo que responderás "si".

5.3. EXPERIENCIA #3: CÁLCULO DE DISTANCIAS

Suele ser muy frecuente que los robots tengan que realizar un recorrido desde un punto inicial hasta un destino situado a cierta distancia, para luego regresar al punto de partida.

Para calcular distancias con el Home Boe-Bot se aplica la fórmula general de la velocidad:

Tiempo = Distancia / Velocidad

Deberás calcular la velocidad del robot. La forma más fácil de hacerlo es poner un metro a su lado y ver que distancia recorre. Conociendo los cm que ha recorrido y el tiempo en segundos empleado en ello, sabrás la velocidad del Home Boe-Bot (cm/s).

- Teclea, guarda y ejecuta el programa ForewardOneSecond.bs2.
- Pon el Home Boe-Bot junto a un metro como se muestra en la figura 5-3.

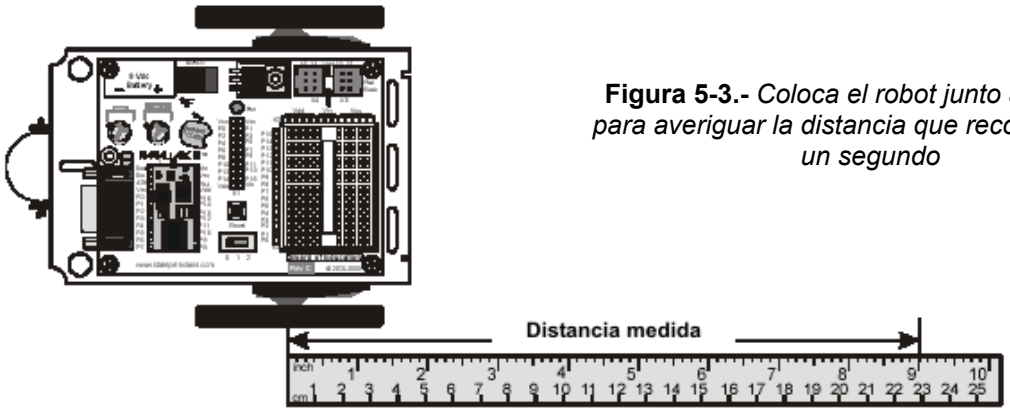


Figura 5-3.- *Coloca el robot junto a un metro para averiguar la distancia que recorre durante un segundo*

- Pulsa el botón de reset de tu placa y arranca el programa. Mide lo que ha recorrido el Home Boe-Bot y apúntalo aquí:_____

Programa BoeBotForwardOneSecond.bs2

```
' El Robot Home Boe-Bot - ForwardOneSecond.bs2
' Conseguir que el robot avance durante 1 seg.
' {$STAMP BS2}
' {$PBASIC 2.5}
counter VAR Word
FREQOUT 4, 2000, 3000                ' Señal de inicio/reset.
FOR counter = 1 TO 41
  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20
NEXT
END
```

El mencionado programa controla el movimiento del robot durante un segundo, por lo que si ha recorrido 23 cm en su ejecución, la velocidad será de 23 cm/s.

Ejemplo: Calcular el tiempo necesario para recorrer una distancia de 51cm.

Tiempo = 51 cm / 23 cm/s = 2,22 s

Ahora se calculan los pulsos que han de ser enviados a los servos. Para ello deberemos multiplicar el tiempo obtenido por 40,65 pulsos/segundo.

Pulsos = 2,22 s x 40,65 pulsos/s = 90 pulsos

Control de la distancia recorrida

- Si todavía no lo has hecho, utiliza un metro y el programa BotForwardOneSecond.bs2 para determinar la velocidad del Home Boe-Bot en cm/s.
- Selecciona la distancia que quieres que recorra el robot.
- Utiliza la ecuación de los pulsos para determinar el número de pulsos que debes mandarle para la distancia elegida.
- Modifica el programa BotForwardOneSecond.bs2 indicando el nuevo número de pulsos.
- Ejecuta el programa y comprueba que realmente recorre la distancia correcta.

5.4. EXPERIENCIA #4: MANIOBRAS DE ACELERACIÓN Y DECELERACIÓN.

Se trata que el Home Boe-Bot acelere y decelere (frene) de forma gradual. Así conseguiremos que los servos y las baterías duren más y que el arranque y parada no sean tan bruscos.

Programa de aceleración

La clave para acelerar es ajustar el argumento **Duración** de **PULSOUT**. La figura 5-4 muestra cómo el bucle **FOR…NEXT** puede hacer que el Home Boe-Bot acelere. Cada vez que se ejecuta el bucle **FOR…NEXT** la variable **pulseCount** se incrementa en 1. A medida de que el valor de **pulseCount es mayor**, la velocidad de los servos también. A la centésima vez que hemos realizado el bucle la variable **pulseCount** vale 100 que es lo mismo que usar los comandos **PULSOUT** 13, 850 y **PULSOUT** 12, 650. Estos comandos son los que se utilizan para mover el Home Boe-Bot a su máxima velocidad.

```
PulseCount     VAR     Word

FOR pulseCount = 1 TO 100

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

NEXT
```

Si por lo contrario hacemos que el bucle **FOR…NEXT** decremente la variable **pulseCount**, el robot decelera hasta pararse. Se propone un programa que hace que el Home Boe-Bot acelere y decelere desde su inicio hasta su parada.

Programa StartAndStopWithRamping.bs2

- Teclea, guarda y ejecuta el programa **StartAndStopWithRamping.bs2**
- Comprueba que el robot acelera hasta su máxima velocidad, mantiene esta máxima velocidad y posteriormente decelera gradualmente hasta pararse.

```
' El Robot Home Boe-Bot - StartAndStopWithRamping.bs2
' Acelera avanzando y luego decelera hasta detenerse.
' {$STAMP BS2}
' {$PBASIC 2.5}
pulseCount VAR Word                    ' Contador del bucle FOR...NEXT.
' -----[ Inicialización ]----------------------------------------------------
FREQOUT 4, 2000, 3000                  ' Señal de Inicio/reset.
' -----[ Rutina principal ]--------------------------------------------------

' Aceleración gradual.
FOR pulseCount = 1 TO 100              ' Bucle de aceleración de 100 pulsos.
  PULSOUT 13, 750 + pulseCount         ' Pulso = 1.5 ms + pulseCount.
  PULSOUT 12, 750 - pulseCount         ' Pulso = 1.5 ms – pulseCount.
  PAUSE 20                             ' Espera de 20 ms.
NEXT

' Avance constante de 75 pulsos.
FOR pulseCount = 1 TO 75               ' Avance constante de 75 pulsos.
  PULSOUT 13, 850                      ' Pulso de 1.7 ms al servo izdo.
  PULSOUT 12, 650                      ' Pulso de 1.3 ms al servo dcho.
  PAUSE 20                             ' Espera de 20 ms.
NEXT

' Deceleración gradual hasta detenerse.
FOR pulseCount = 100 TO 1                  ' Bucle de deceleración de 100 pulsos.
  PULSOUT 13, 750 + pulseCount             ' Pulso = 1.5 ms + pulseCount.
  PULSOUT 12, 750 - pulseCount             ' Pulso = 1.5 ms - pulseCount.
  PAUSE 20                                 ' Espera de 20 ms.
NEXT

END                                        ' Stop hasta un nuevo Reset.
```

Ahora puedes crear rutinas para combinar las acciones de aceleración y deceleración junto con el resto de maniobras. Se ofrece un ejemplo de cómo ir acelerando el Home Boe-Bot yendo hacia atrás en vez de hacia delante. La única diferencia entre que vaya hacia delante o hacia atrás es que en este caso el valor de **pulseCount** es el sustraendo de 750 en el comando **PULSOUT** 13, donde antes actuaba como un sumando.

```
' Aceleración al máximo en retroceso
FOR pulseCount = 1 TO 100

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

Se sugiere otro ejercicio para que el robot gire a la derecha mientras va acelerando para después decelerar hasta pararse otra vez:

```
'Aceleración rotación derecha
FOR pulseCount = 1 TO 30

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

'Deceleración rotación derecha
FOR pulseCount = 30 TO 0

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

Modifica el programa ForwardLeftRightBackward.bs2 para que acelere y decelere en cada una de las maniobras que realiza. Finalmente guárdalo como ForwardLeftRightBackwardRamping.bs2 y ejecútalo en el Home Boe-Bot para comprobar que realmente funciona.

5.5. EXPERIENCIA #5: FACILITAR LOS MOVIMIENTOS DEL ROBOT CON SUBRUTINAS

El Home Boe-Bot va a ser capaz, en el siguiente tema de este manual, de realizar maniobras con el fin de evitar obstáculos. Una de las formas más eficientes de esquivar obstáculos es realizando maniobras pre-programadas mediante la utilización de subrutinas. En esta práctica aprenderás a utilizar subrutinas y a crearlas con el fin de programar maniobras predefinidas.

Las subrutinas

Una subrutina es una secuencia de instrucciones que se repite en diversas ocasiones a lo largo del programa principal. Para no tenerla que repetir e insertar en el programa cada vez que se necesita, sólo se pone una vez como un programa independiente y cada vez que se precisa se la llama.

Hay dos partes dentro de una subrutina en PBASIC. La primera es la "llamada a la subrutina", es decir, la instrucción que hace que una vez llegue el programa allí, se ejecute la parte de código que contiene dentro de la subrutina. La otra parte es la propia subrutina en sí. La subrutina comienza cuando se referencia con GOSUB su nombre (también llamado etiqueta) y finaliza con el comando RETURN, que devuelve el control a la siguiente instrucción a GOSUB. El código que hay entre la etiqueta del nombre de la subrutina y RETURN, es lo que se ejecutará en cada llamada a esa subrutina.

La figura 5-5 nos presenta un trozo de código en PBASIC que contiene una llamada a subrutina y la propia subrutina. La llamada a subrutina se realiza con el comando **GOSUB** y el nombre de la subrutina, en nuestro caso My_Subroutine. Un vez se inicia la subrutina, se va ejecutando línea a línea hasta encontrar el comando RETURN, que obliga a salir de la subrutina y seguir ejecutando el programa en el punto en que se abandonó.

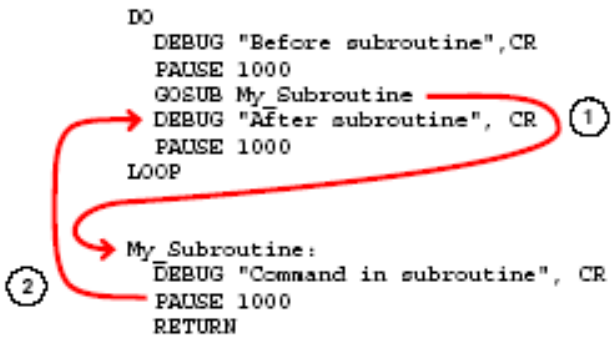


Figura 5-5.- *La llamada a la subrutina con GOSUB inicia la ejecución de las líneas de la misma hasta encontrar RETURN que devuelve el flujo de control a la siguiente instrucción a GOSUB.*

Programa OneSubroutine.bs2

- Teclea, guarda y ejecuta el programa **OneSubroutine.bs2**

```
' El Robot Home Boe-Bot - OneSubroutine.bs2
' Demostración de una simple llamada a subrutina.
' {$STAMP BS2}
' {$PBASIC 2.5}
DEBUG "Antes de la subrutina",CR
PAUSE 1000
GOSUB My_Subroutine
DEBUG "Después de la subrutina", CR
END

My_Subroutine:
  DEBUG "Subrutina en ejecución", CR
  PAUSE 1000
  RETURN
```

- Mira la ventana del Debug Terminal y pulsa reset varias veces. Deberás obtener el mismo resultado en los distintos casos.

Una vez que conoces el uso y funcionamiento de las subrutinas, vamos a usarlas para manejar los movimientos del robot.

Programa MovementsWithSubroutines.bs2

- Teclea, guarda y ejecuta el programa MovementsWithSubroutines.bs2

```
' El Robot Home Boe-Bot - MovementsWithSubroutines.bs2.
' Realizar los movimientos básicos mediante subrutinas.
' {$STAMP BS2}
' {$PBASIC 2.5}
counter VAR Word

FREQOUT 4, 2000, 3000                    ' Señal de inicio/reset.
GOSUB Forward
```

```
GOSUB Left
GOSUB Right
GOSUB Backward
END

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

  ' Giro a la izda.
Left:
  FOR counter = 1 TO 24
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  PAUSE 200
  RETURN

  ' Giro a la dcha.
Right:
  FOR counter = 1 TO 24
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  PAUSE 200
  RETURN

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

Se propone un programa más avanzado donde se incluyen las variables para poder configurar las maniobras del Home Boe-Bot. Fíjate antes en estos dos trozos de código:

```
' Forward full speed

FOR counter = 1 TO 64

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

NEXT
```

```
' Ramp down from full speed backwards

FOR pulseCount = 100 TO 1

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

NEXT
```

Lo que hace que estos dos trozos de código ejecuten distintas maniobras es el cambio en los argumentos **FOR StartValue**, **EndValue** y **PULSOUT**. Estos argumentos pueden ser variables y éstas pueden ser modificadas a lo largo del programa. En vez de utilizar para cada maniobra una subrutina puedes hacer más sencillo el programa con el uso de variables a las que se introduce ciertos valores para cada tipo de movimiento.

```
' El Robot HomeBoe-Bot - MovementWithVariablesAndOneSubroutine.bs2
' Realizar una rutina de navegación que acepte parámetros externos.
' {$STAMP BS2}
' {$PBASIC 2.5}

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

FREQOUT 4, 2000, 3000                          ' Señal de inicio/reset.
' Avance
pulseLeft = 850: pulseRight = 650: pulseCount = 64: GOSUB Navigate
' Giro a la izda.
pulseLeft = 650: pulseRight = 650: pulseCount = 24: GOSUB Navigate
' Giro a la dcha.
pulseLeft = 850: pulseRight = 850: pulseCount = 24: GOSUB Navigate
' Retroceso
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
```

Ahora ejecuta el programa anterior y comprueba la secuencia delante-izquierda-derecha y hacia atrás.

- Modifica el ejemplo anterior para que el Home Boe-Bot se mueva describiendo un cuadrado.

5.6. EXPERIENCIA #6: PROGRAMAR MANIOBRAS COMPLEJAS CON LA EEPROM

Cuando se introduce un programa el editor BASIC Stamp lo convierte en valores numéricos llamados "tokens". Los tokens son lo que utilizan los módulos de Parallax como instrucciones para ejecutar el programa. Estos datos son guardados en un chip de memoria EEPROM de 2 KB de la Home Work, modelo 24LC16B. También se pueden guardar en las posiciones de esta memoria EEPROM los datos o valores que nos interesen y posteriormente leerlos.

Es posible ver el mapa de la memoria EEPROM en el editor BASIC STAMP seleccionando RUN → MEMORY MAP. La figura 5-6 nos muestra el mapa de memoria para el programa MovementsWithSubroutines.bs2

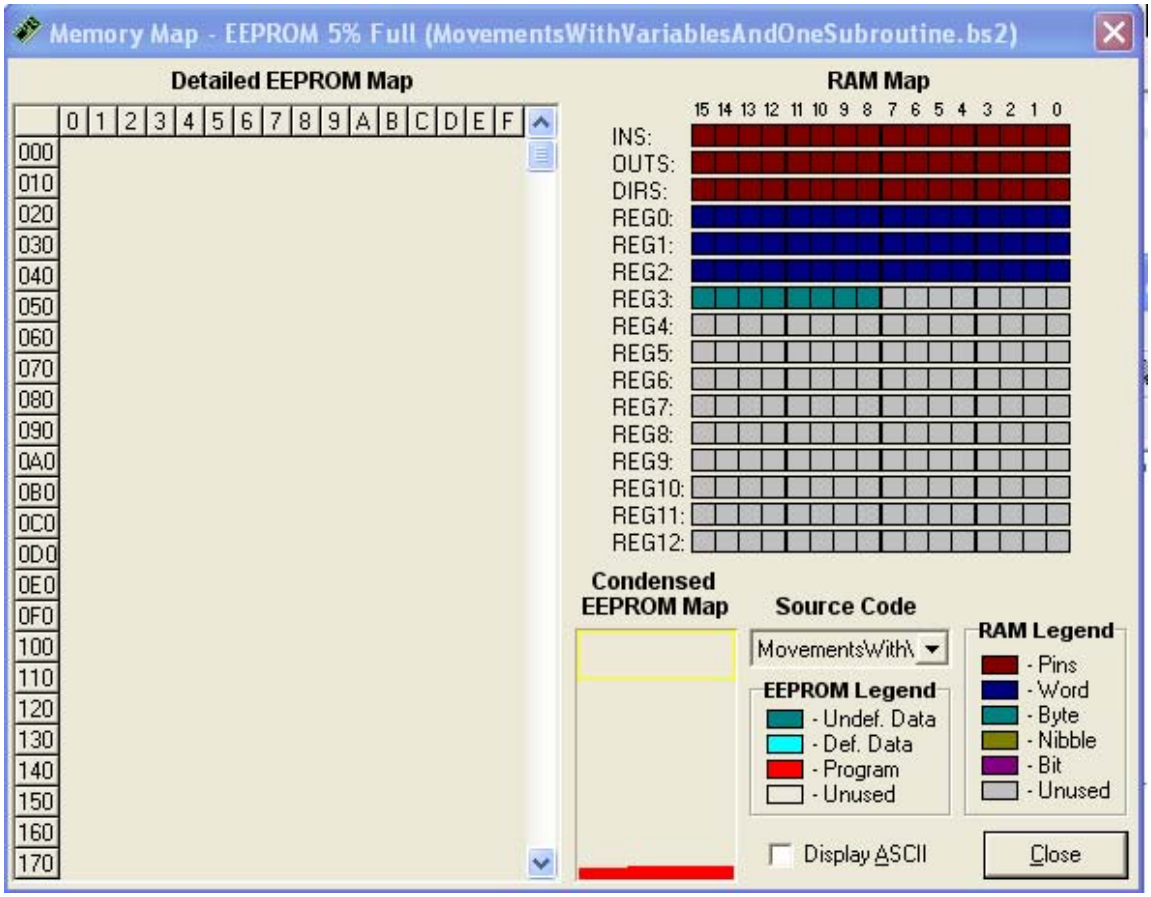


Figura 5-6.- *Mapa de memoria de la EEPROM y la RAM para el programa MovementsWithSubroutines.bs2.*

Para la programación (grabación y lectura de datos) de la EEPROM se utiliza la directiva **DATA** y los comandos **READ** y **SELECT…CASE…ENDSELECT**. Vamos a explicar un poco estos comandos antes de entrar en la confección de programas para mover al Boe-Bot por caminos más largos y difíciles.

Cada maniobra básica del Home Boe-Bot estará referenciada por una letra (F: Hacia delante, B: Hacia Atrás, L: Izquierda, R: Derecha) y esa letra a su vez será una subrutina con lo que podremos hacer rutas complicadas y guardarlas como conjuntos de letras en la EEPROM. Para ello crearemos una cadena ("String") con todos los movimientos que queremos que haga sin olvidar que debe acabar con la letra Q que significa "Salir". Para guardar la ruta en la EEPROM utilizaremos la directiva DATA, que se muestra a continuación:

```
DATA "FLFFRBLBBQ"
```

Cada letra es guardada en un byte de le EEPROM, comenzando desde la dirección 0 (a menos que le indiquemos nosotros la dirección dónde queremos que empiece). El comando **READ** se utiliza para obtener o leer la lista que antes hemos guardado en la EEPROM mientras se ejecuta el programa. Estos valores los podemos obtener con un bucle **DO…LOOP** de este estilo:

```
DO
        READ address, instruction
        address = address + 1
        ' PBASIC code block omitted here.
LOOP
```

La variable **address** corresponde a la ubicación de cada byte en la EEPROM. La variable **instruction** guardará el valor actual de ese byte, en nuestro caso, cada letra. Lo que conseguimos con el bucle entonces es incrementar el valor de **address** cada vez que pasamos por él, con lo que iremos leyendo sucesivas posiciones de la EEPROM en cada vuelta del bucle.

Realizaremos un bloque usando el comando **SELECT...CASE...ENDSELECT** para que cada letra haga que se ejecute un procedimiento distinto. El funcionamiento de esta instrucción es la misma que en los demás lenguajes de programación y consiste en que según el valor que tome la variable ejecutará la operación (subrutina) correspondiente a lo que introduzcamos para esa variable (instrucción). En nuestro caso sería lo siguiente:

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

Tenga en cuenta que finaliza el programa de acceso a la subrutina cuando se encuentra la letra Q ("salir").

```
DO
        ' PBASIC code block omitted here.
LOOP UNTIL instruction = "Q"
```

Ahora juntamos todos los conceptos en un nuevo programa que llamaremos EepromNavigation.bs2

Programa EepromNavigation.bs2

- Lee todos los comentarios con detenimiento para entender todo lo que el programa realiza.
- Teclea, guarda y ejecuta el programa para ver que realmente funciona bien.

```
' El Robot Home Boe-Bot - EepromNavigation.bs2
' Navegación mediante el empleo de caracteres almacenados
' en la memoria EEPROM.
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]
pulseCount VAR Word             ' Almacena nº de pulsos.
address VAR Byte                ' Almacena dirección de EEPROM.
instruction VAR Byte            ' Almacena instrucción en EEPROM.

' -----[ Datos EEPROM ]
' Dirección: 0123456789          ' Estas 2 líneas muestran la
'            ||||||||||         ' dirección EEPROM de cada dato.
DATA     "FLFFRBLBBQ"           ' Instrucciones de navegación.
' -----[ Inicialización ]

FREQOUT 4, 2000, 3000           ' Señal de inicio/reset.
' -----[ Rutina principal ]

DO
  READ address, instruction     ' Lee una instrucción de la dirección.
  address = address + 1         ' Incrementa la dirección para la siguiente lectura.
  SELECT instruction
    CASE "F": GOSUB Forward
    CASE "B": GOSUB Backward
```

```
   CASE "L": GOSUB Left_Turn
   CASE "R": GOSUB Right_Turn
  ENDSELECT
LOOP UNTIL instruction = "Q"
END                             ' Detiene  la ejecución hasta un Reset.

' -----[ Subrutina - avance ]
Forward:                        ' Subrutina de avance.
  FOR pulseCount = 1 TO 64      ' Envía 64 pulsos de avance.
    PULSOUT 13, 850             ' Pulso de 1.7ms al servo izdo.
    PULSOUT 12, 650             ' Pulso de 1.3ms al servo dcho.
    PAUSE 20                    ' Espera de 20 ms.
  NEXT
  RETURN                        ' Retorno al bucle principal.

  ' -----[ Subrutina - retroceso ]
Backward:                       ' Subrutina de retroceso.
  FOR pulseCount = 1 TO 64      ' Envía 64 pulsos de retroceso.
    PULSOUT 13, 650             ' Pulso de 1.3ms al servo izdo.
    PULSOUT 12, 850             ' Pulso de 1.7ms al servo dcho.
    PAUSE 20                    ' Espera de 20mS.
  NEXT
  RETURN                        ' Retorno al bucle principal.

  ' -----[ Subrutina - giro izdo ]
Left_Turn:                      ' Subrutina de giro a la izda.
  FOR pulseCount = 1 TO 24      ' Envía 24 pulsos de giro a la izda.
    PULSOUT 13, 650             ' Pulso de 1.3ms al servo izdo.
    PULSOUT 12, 650             ' Pulso de 1.3ms al servo dcho.
    PAUSE 20                    ' Espera de 20ms.
  NEXT
  RETURN                        ' Retorno al bucle principal.

' -----[ Subrutina - giro dcha ]
Right_Turn:                     ' Subrutina de giro a la dcha.
  FOR pulseCount = 1 TO 24      ' Envía 24 pulsos de giro a la dcha.
    PULSOUT 13, 850             ' Pulso de 1.7ms al servo izdo.
    PULSOUT 12, 850             ' Pulso de 1.7ms al servo dcho.
    PAUSE 20                    ' Espera de 20ms.
  NEXT
  RETURN                        ' Retorno al bucle principal.
```

Al ejecutar el programa, tu Home Boe-Bot ¿anda describiendo rectángulos? Si describe un trapecio lo que debes hacer es ajustar los argumentos **FOR pulseCount *EndValue*** en las subrutinas con el fin de que realice giros de 90º

Con el programa anterior en ejecución, dentro del editor BASIC STAMP selecciona RUN→ MEMORY MAP.

Las letras indicativas de subrutinas de movimiento que has guardado aparecerán remarcadas (en azul) en el mapa detallado de la EEPROM como se muestra en la figura 5-7. Los números se muestran en Hexadecimal por lo que deberás pulsar en Display ASCII para poder ver los caracteres que has introducido antes en una cadena con la instrucción DATA (Figura 5-8).

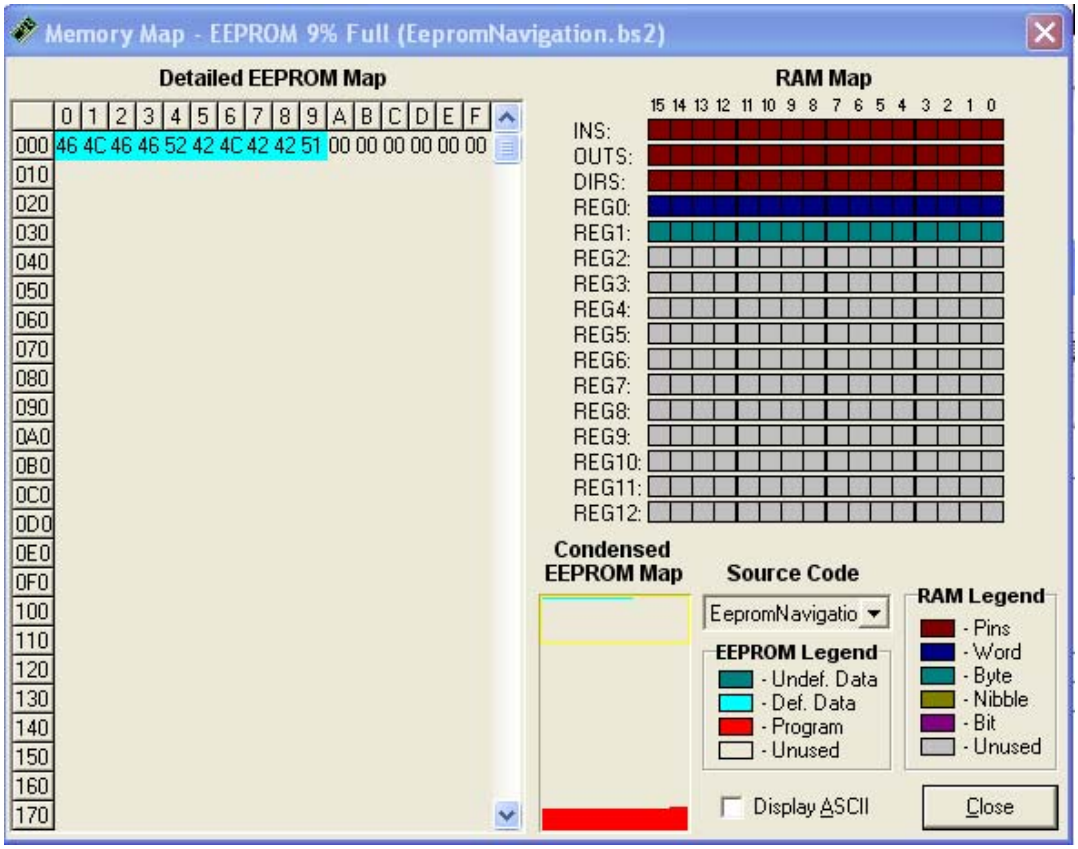


Figura 5-7.- *Las letras indicativas de subrutinas de movimiento guardadas en la EEPROM se visualizan en el mapa de memoria con sus valores hexadecimales correspondientes.*

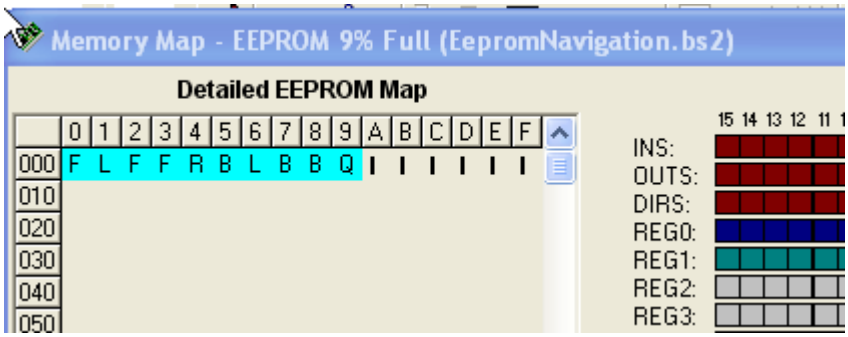


Figura 5-8.- *Cuando se convierte a ASCII los números hexadecimales se traducen a las letras indicativas de las subrutinas.*

El programa que estás ejecutando guarda un total de 10 caracteres en la EEPROM. Estos 10 caracteres son accesibles con la variable **address** del comando **READ**. Esta variable está declarada como byte, con lo que puede acceder hasta a 256 posiciones. Si quisieras guardar más posiciones deberías declararla como word, así conseguirías acceder a 65535 posiciones

Ahora puedes modificar la cadena que antes hemos puesto por otra para que el Home Boe-Bot haga distintas rutas. Es posible añadir más directivas **DATA**, pero recuerda que estas guardarán los datos inmediatamente después de lo anterior almacenado (de la directiva **DATA** anterior).

Programa EepromNavigationWithWordValues.bs2

A primera vista, el siguiente programa parece complicado, pero es la forma más eficiente de realizar movimientos totalmente personalizados con el Home Boe-Bot. El programa que tenemos a continuación utiliza la EEPROM, pero no así las subrutinas ya que se opta por guardar también en la EEPROM las variables que harán que el robot se mueva en distintas direcciones.

Por defecto, la directiva **DATA** guarda bytes de información en la EEPROM. Para guardar datos de tamaño word (palabra) deberás utilizar el modificador word. Cuando se utiliza más de una directiva **DATA** es recomendable ponerlas nombres a las mismas. Fíjate en este trozo de código, recordando que cada word consta de dos bytes.

```
' 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 tres sentencias **DATA** empiezan con su propia etiqueta. El modificador word va antes de cada uno de los datos, los cuales van separados por comas. Las tres cadenas se van a guardar una detrás de otra en la EEPROM. No necesitas calcular nada ya que la etiquetas y la variable **addressOffset** lo harán automáticamente. El comando **READ** usa cada etiqueta para determinar donde empieza cada cadena y después añadirá el valor de la variable **addressOffset** para saber dónde debe localizar la cadena. Los datos obtenidos los guardará entonces en una variable de tipo word al final de la sentencia **READ**. Fíjate en este trozo de código:

```
DO
        READ Pulses_Count + addressOffset, Word pulseCount
        READ Pulses_Left + addressOffset, Word pulseLeft
        READ Pulses_Right + addressOffset, Word pulseRight
        addressOffset = addressOffset + 2
        ' PBASIC code block omitted here.
LOOP
```

La primera vez que se pasa por el bucle, **addressOffset** = 0. El primer comando **READ** lo que hará es obtener el valor de 64 y lo asignará a la variable **pulseCount**. El segundo comando **READ** obtiene el valor 850 y lo asignará en este caso a la variable **pulseLEFT**. El tercer comando **READ** hará algo similar con la variable **pulseRight**. Cuando estos valores están ya cargados en el programa tendríamos que:

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

Que al tomar valores queda:

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

¿Te suena el código generado?

Ahora escribe, guarda y ejecuta el programa EepromNavigationWithWordValues.bs2.

```
' El Robot Home Boe-Bot - EepromNavigationWithWordValues.bs2
' Almacena una lista de valores WORD.
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]
counter VAR Word
pulseCount VAR Word                                  ' Almacena nº de pulsos.
addressOffset VAR Byte                               ' Almacena desplazamiento.
instruction VAR Byte                                 ' Almacena instrucción.
pulseRight VAR Word                                  ' Almacena anchura de pulso.
pulseLeft VAR Word

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

' -----[ Inicialización ]
FREQOUT 4, 2000, 3000                                ' Señal de 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                                                  ' Detiene la ejecución hasta el Reset.
```

Después de ejecutarlo, ¿te suena la ruta que ha descrito? Ahora es el momento para que tú consigas que el Home Boe-Bot describa caminos más complejos.

El Home Boe-Bot es capaz de moverse por caminos complicados

- Guarda EepromNavigationWithWordValues.bs2. con otro nombre.
- Reemplaza las directivas DATA como se indica a continuación:

```
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
```

- Ejecuta el programa y comprueba lo que hace el robot
- Haz una tabla con 3 filas, una por cada directiva **DATA** y una columna por cada maniobra que quieres que haga el Home Boe-Bot, más una para el item de "WORD 0" en la fila **Pulses_Count**.

- Usa esta tabla para controlar lo que va a hacer el Home Boe-Bot rellenado cada argumento **FOR…*EndValue*** y **PULSOUT *Duration*** que vas a necesitar para cada movimiento.
- Modifica tu programa con las nuevas tablas para aplicarles las directivas DATA personalizadas para tu proyecto.
- Ejecuta el programa y comprueba que el Home Boe-Bot describe la trayectoria prevista.

5.7 PRUEBA DE AUTOEVALUACION

Preguntas

1. ¿Que sentido de giro debe tener la rueda izquierda para que el Home Boe-Bot pueda ir hacia delante?
2. ¿Cuál debe tener la rueda derecha para que el robot gire?
3. ¿Cuál debe tener la rueda derecha para que el Home Boe-Bot pueda retroceder?
4. ¿Que dirección debe seguir la rueda izquierda y la derecha para que el robot t gire?
5. ¿Qué es lo que controla la velocidad y dirección de los servos?, ¿qué comando PBASIC utilizamos para ello?
6. ¿Cuando el robot pivota a la derecha, que hace cada rueda?
7. ¿Cómo se resuelve el problema originado cuando quieres que el Home Boe-Bot vaya recto hacia delante y se desvíe hacia un lado?
8. Si el Home Boe-Bot se mueve a 60cm/s, ¿cuántos pulsos necesitará para recorrer 1 metro?
9. ¿Qué relación hay entre el argumento **counter** de un bucle **FOR…NEXT** y el argumento **Duración** del comando **PULSOUT** ?
10. ¿Qué diferencia hay entre una subrutina y una llamada a ésta?
11. ¿Qué comando puedes utilizar para guardar valores en la EEPROM de la BASIC Stamp antes de ejecutar un programa?

Ejercicios

1. Escribe una rutina que haga que el Home Boe-Bot retroceda durante 350 pulsos.
2. Escribe una rutina que haga que el robot pivote hacia atrás y la izquierda durante 50 pulsos.
3. Vamos a pensar que has calculado que para que el Home Boe-Bot gire 90º hay que mandarle 48 pulsos. Con esta información calcula los pulsos que hay que mandarle para que gire 30, 45 y 60 grados.
4. Escribe un rutina que haga que vaya hacia delante, que acelere y decelere en un giro con la operación de pivotar y después siga andando hacia delante.
5. Hay cuatro formas de pivotar. Escribe una rutina para cada una de ellas.

Proyectos

1. Confecciona un programa para realizar un recorrido personalizado con el Home Boe-Bot valiéndote de las tablas que hemos manejado en el último ejercicio de este tema.
2. En la figura 5-9 vemos varios recorridos posibles. Haz que tu robot describa cada uno de los recorridos que se muestran.

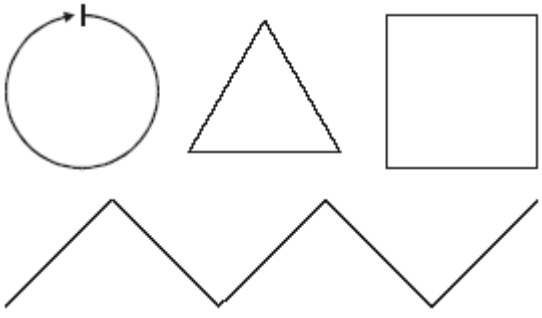


Figura 5-9.- *Posibles recorridos para el Home Boe-Bot.*

3. Modifica los programas del Proyecto para obtener distintos resultados.
4. Escribe una subrutina para cada maniobra del Proyecto 1.

Tema 6

Navegación con antenas táctiles

6.1. UNOS BIGOTES PARA NUESTRO ROBOT

En el tema anterior enseñamos a caminar al Home Boe-Bot, pero "a ciegas". Si tropezaba con un obstáculo allí se quedaba. En esta ocasión vamos a colocarle unos sensores que le avisen de la presencia de obstáculos en su camino, además de conocer si dicho obstáculo se encuentra a la derecha o a la izquierda. Conocida esta información podemos mandar realizar un movimiento de desvío que lo evite y pueda proseguir hacia su destino.

Usaremos dos "bumpers", nombre técnico que se asigna a ciertos interruptores que cierran sus contactos cuando se presiona uno de ellos. Los bumper que emplea el Home Boe-Bot están construidos mediante un alambre de acero y tienen aspecto de antenas. Les llamaremos "bigotes" porque sirven para detectar objetos como lo hacen los bigotes de un gato o las antenas de una hormiga. Figura 6-1.

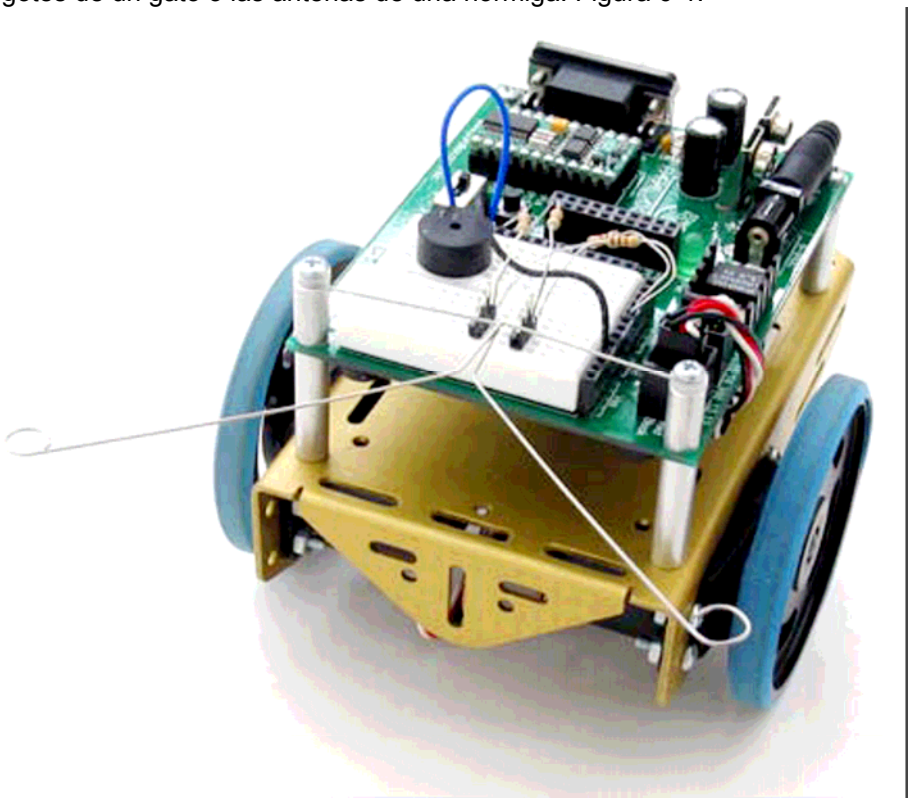


Figura 6-1.- *Los "bigotes" del Home Boe-Bot le advierten la presencia de obstáculos en su camino.*

6.2. EXPERIENCIA #1: MONTANDO Y PROBANDO LOS BIGOTES

Para montar los bigotes al robot se precisan los siguientes materiales, que se muestran en la figura 6-2.

(2) "Bigotes"
(2) Tornillos de cabeza plana de M3 x 15mm
(2) Separadores metálicos H-H de M3 x 10 mm
(2) Arandelas de baquelita de 3 mm
(2) Conectores de 3 pines
(2) Resistencias de 220 Ω
(2) Resistencias de 10 kΩ

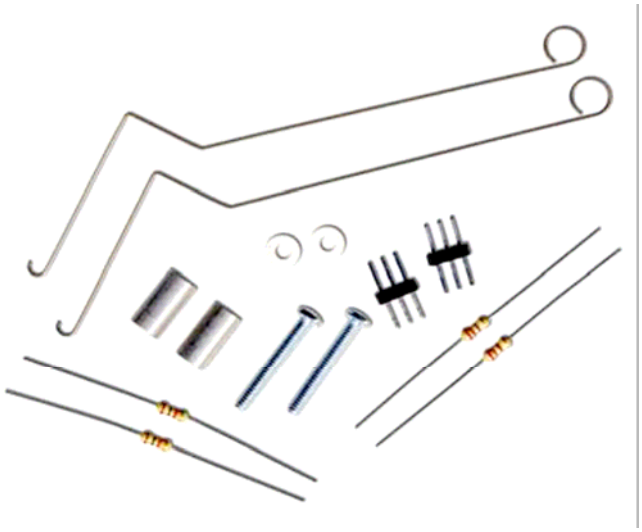


Figura 6-2.- *Materiales necesarios para montar los bigotes.*

Las operaciones para el montaje son:

- Quita los dos tornillos frontales que unen la tarjeta Home Work con los dos separadores frontales.
- Fíjate en la figura 6-3 para comprender las siguientes operaciones.
- Coloca una arandela de baquelita y un separador de M3x10 en cada uno de los tornillos de M3x15.
- Coloca los tornillos sobre los agujeros de la tarjeta y atorníllalos a los separadores que hay debajo, pero no los aprietes completamente todavía.
- Engancha los bigotes a los tornillos. Coloca cada uno sobre una de las arandelas, de tal forma que no se toquen entre si.
- Ahora termina de apretar los tornillos.

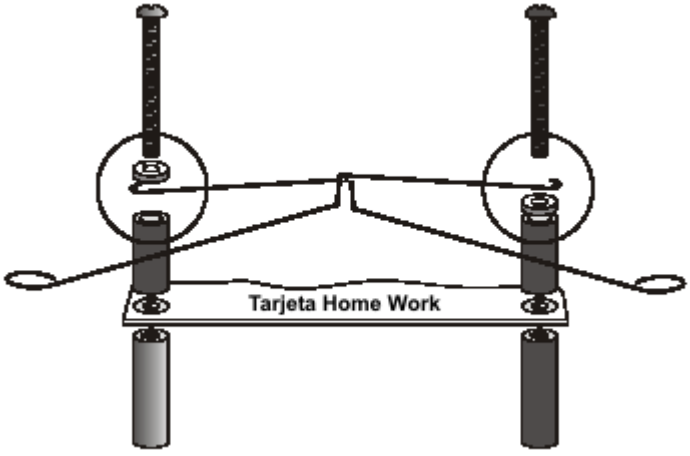


Figura 6-3.- *Detalle del montaje de los bigotes.*

Ahora construiremos el circuito eléctrico de los bigotes para añadirlo a los circuitos del zumbador y los servos que se montaron en el tema anterior. Figura 6-4.

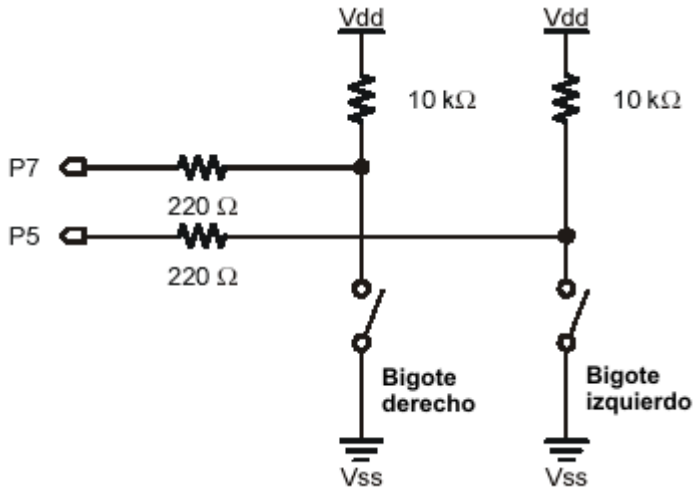


Figura 6-4.- *Circuito eléctrico para el conexionado del bigote derecho y del izquierdo.*

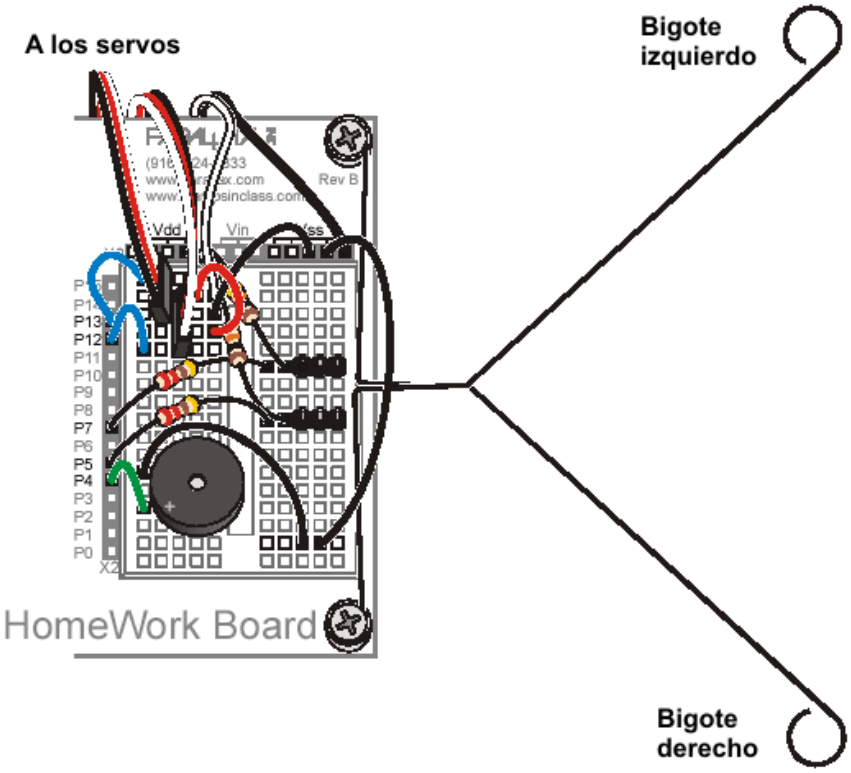


Figura 6-5.- *Aspecto de la protoboard de la Home Work una vez montados los circuitos de los bigotes, del zumbador y de los servos.*

Probando los bigotes

Echa otro vistazo al esquema de los bigotes (figura 6-6). Cada bigote es una extensión mecánica de un interruptor normalmente abierto que tiene un extremo conectado a tierra. La razón por la que los bigotes están conectados a tierra (Vss) es que los agujeros en los bordes exteriores de la tarjeta están conectados a tierra. Los separadores y tornillos metálicos conectan la señal de tierra con los bigotes.

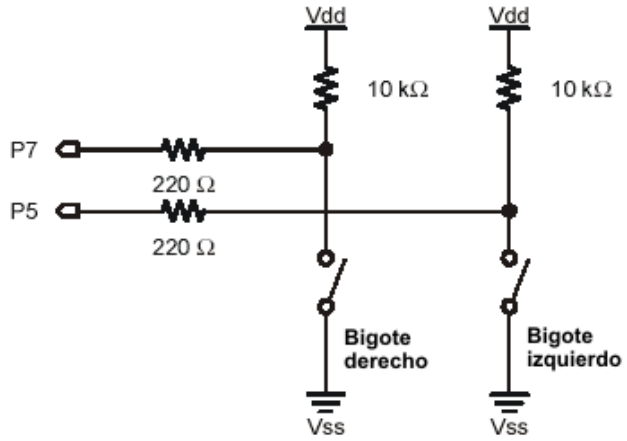


Figura 6-6.- *El bigote no es más que una extensión mecánica de un interruptor normalmente abierto que tiene un extremo conectado a tierra.*

El módulo microcontrolador puede ser programado para detectar cuando es presionado un bigote. Los pines o patitas de E/S conectados a cada interruptor están configurados como entradas y reciben un voltaje (Figura 6-7). Cuando los bigotes están sin presionar el voltaje que reciben los pines de E/S es de 5 V (1 lógico). Pero si son presionados, el circuito se cierra con tierra y los pines reciben 0 V (0 lógico).

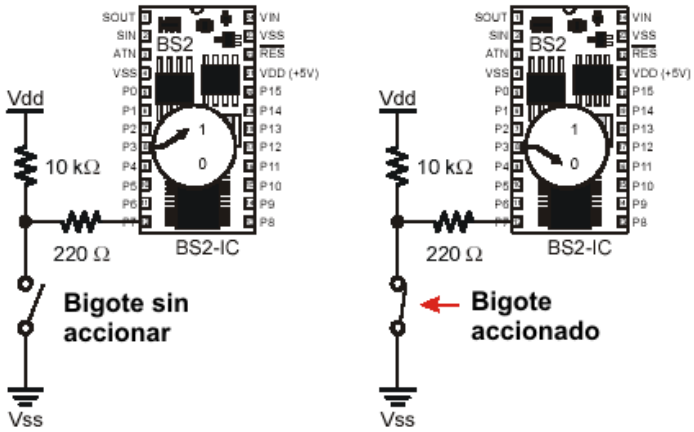


Figura 6-7.- *Si el bigote está sin presionar (izquierda) el pin de E/S recibe un nivel lógico alto (1), pero si se presiona y se cierra, el contacto del bigote que va a tierra e introduce un nivel bajo (0).*

Programa TestWhiskers.bs2

Este programa está diseñado para que puedas probar los bigotes y asegurarte que funcionan correctamente. Lo que hace es mostrar el estado lógico de las entradas que corresponden a los pines de E/S conectados a los bigotes (IN7 e IN5).

Todos los pines de E/S son de entrada por defecto, a menos que se programe lo contrario. Por lo tanto, los pines conectados a los bigotes tendrán un "1" si el voltaje que reciben es 5 V (el bigote no está presionado) o un "0" si lo que reciben es 0 V (el bigote está presionado). Puedes utilizar el Debug Terminal para mostrar estos valores.

- Conecta las pilas a tu placa y a los servos.
- Teclea, salva y ejecuta el programa TestWhiskers.bs2
- Este programa utiliza el Debug Terminal, por lo que deberás dejar conectado el cable serie con el PC mientras se está ejecutando.

```
' El Robot Home Boe-Bot - TestWhiskers.bs2
' Visualizar las líneas de E/S conectadas a los "bigotes".
' {$STAMP BS2}.
' {$PBASIC 2.5}.
DEBUG "Estado Bigotes", CR,"Izda Dcha.", CR,"------ ------"
DO
  DEBUG CRSRXY, 0, 3,"P5 = ", BIN1 IN5," P7 = ", BIN1 IN7
  PAUSE 50
LOOP
```

- Fíjate en los valores que muestra el Debug Terminal; deberían indicar que P7 y P5 son igual a 1.
- Antes de continuar, asegúrate de cual es la patita de entrada conectada a cada uno de los bigotes.
- Presiona el bigote derecho contra el cabezal de 3 pines hasta que haga buen contacto y fíjate lo que se visualiza en la ventana del Debug Terminal. Debería mostrar: P5 = 1, P7 = 0.
- Presiona el bigote izquierdo contra el cabezal de 3 pines hasta el contacto y fíjate en lo que muestra el Debug Terminal. Debería ser: P5 = 0, P7 = 1.
- Presiona los dos bigotes el mismo tiempo. Ahora lo que se debería mostrar es: P5 = 0, P7 = 0.
- Si todo es correcto, pasa a la siguiente actividad.

6.3. EXPERIENCIA #2: OTRA FORMA DE PROBAR LOS BIGOTES

¿Qué ocurriría si no tuvieras a tu disposición un ordenador con el que ver el estado de las entradas en el Debug Terminal? ¿Cómo comprobarías si has montado correctamente los bigotes? Una posible solución es programar la tarjeta Home Work para que saque un valor determinado en función de cuál es la entrada que está activada. Esto puede materializarse con un par de diodos LED que se enciendan y se apaguen según los bigotes estén presionados o no.

Para montar este circuito que refleje en dos LED el estado de los bigotes se necesita.

(2) Resistencias de 220 Ω
(2) Diodos LED

Para montar el circuito de los LED:

- Desconecta las pilas de la tarjeta y los servos.
- Monta sobre la protoboard los esquemas de las figuras 6-8 y 6-9.

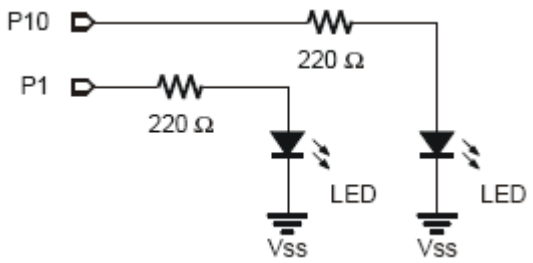


Figura 6-8.- *Esquema eléctrico de los LED que monitorizan el estado de los bigotes.*

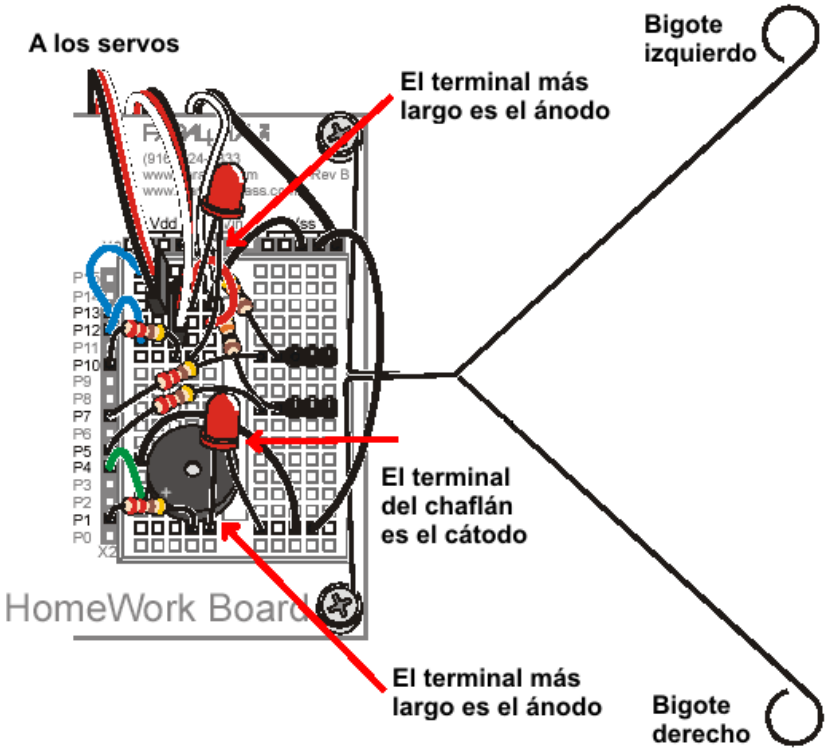


Figura 6-9.- *Aspecto de la protoboard una vez montados los circuitos de los bigotes, del zumbador, de los servos y de los dos LED de monitorización del estado de los bigotes.*

Una vez montado el circuito mostrado en la figura 6-9:

- Conecta pila a la tarjeta Home Work.
- Salva el programa TestWhiskers.bs2 como TestWhiskersWithLeds.bs2 (cambia su nombre).
- Añade estas dos instrucciones entre el segundo comando DEBUG y el comando PAUSE 50.

```
IF (IN7 = 0) THEN
        HIGH 1
ELSE
        LOW 1
ENDIF
IF (IN5 = 0) THEN
        HIGH 10
ELSE
        LOW 10
ENDIF
```

Las declaraciones IF...THEN serán explicadas más extensamente en posteriores apartados. En PBASIC se utilizan para tomar decisiones. La primera declaración pone P1 a nivel alto, de tal forma que el LED se iluminará cuando el bigote conectado a P7 esté presionado (IN7 = 0). La parte del ELSE hace que el LED se apague cuando el bigote no está presionado. La segunda declaración hace lo mismo para el otro bigote (conectado a P5, y su LED conectado a P10).

- Ejecuta el programa TestWhiskersWithLeds.bs2
- Comprueba el comportamiento del programa presionando los bigotes y comprobando que se enciende el LED correspondiente.

6.4. EXPERIENCIA #4: LA RESPUESTA AL ESTADO DE LOS BIGOTES

En esta experiencia vamos a tratar de que el robot reaccione a la información que introducen los bigotes. Cuando el Home Boe –Bot esté moviéndose y uno de sus bigotes sea presionado, significa que ha tropezado con un obstáculo. El programa de exploración deberá comprobar esta entrada y decidir que maniobra hay que ejecutar para evitar dicho obstáculo y dirigir al robot en otra dirección.

El siguiente programa hace que el Home Boe-Bot vaya hacia delante hasta encontrarse con un obstáculo. En el momento en que el obstáculo sea detectado por los bigotes las rutinas y subrutinas escritas en el tema anterior harán que el Home Boe-Bot retroceda y gire. Después, seguirá avanzando hasta que se encuentre con otro obstáculo.

El microcontrolador que gobierna al Home Boe-Bot tiene que ser programado para que tome decisiones cuando uno de los bigotes sea presionado. El lenguaje PBASIC dispone de un comando llamado "IF...THEN". La sintaxis es la siguiente:

IF (condición) THEN...{ELSEIF (condición)}…{ELSE}…ENDIF

Los puntos suspensivos significan que se puede meter un trozo de código en su lugar. El siguiente programa hace que el Home Boe-Bot tome decisiones en función de las entradas de los bigotes, llamando a subrutinas de movimiento ya conocidas.

```
IF (IN5 = 0) AND (IN7 = 0) THEN
        GOSUB Back_Up           'Si los dos bigotes están activados
        GOSUB Turn_Left         'el robot retrocede y realiza un giro
        GOSUB Turn_Left         ' en forma de U
ELSEIF (IN5 = 0) THEN
        GOSUB Back_Up           'Si el bigote izdo. Está activado se
        GOSUB Turn_Right        'retrocede y giro a la dcha.
ELSEIF (IN7 = 0) THEN
        GOSUB Back_Up           'Si el bigote dcho. Está activado se
        GOSUB Turn_Left         'retrocede y giro a la izda.
ELSE
        GOSUB Forward_Pulse     'Si ningún bigote esstá activado
ENDIF                           'se avanza
```

Programa RoamingWithWhiskers.bs2

Este programa muestra una forma de evaluar las entradas proporcionadas por los bigotes para decidir a qué subrutina de movimiento llamar, utilizando la declaración IF...THEN.

- Conecta las pilas a la tarjeta y a los servos.
- Teclea , salva y ejecuta el programa RoamingWithWhiskers.bs2.

```
' -----[ Title ]-------------------------------------------------------------
' El Robot Home Boe-Bot - RoamingWithWhiskers.bs2
' El Home Boe-Bot emplea los bigotes para detectar objetos y navegar en
' función del estado de los mismos.
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]
pulseCount VAR Byte          ' Contador para el bucle FOR...NEXT

' -----[ Inicialización ]
FREQOUT 4, 2000, 3000        ' Señal de inicio/reset.
```

```
' -----[ Rutina principal ]
DO
  IF (IN5 = 0) AND (IN7 = 0) THEN      ' Ambos bigotes detectan obstáculo
    GOSUB Back_Up                      ' Retrocede y gira a la izda.
    GOSUB Turn_Left
    GOSUB Turn_Left
  ELSEIF (IN5 = 0) THEN                ' Bigote izdo. activado
    GOSUB Back_Up                      ' Retroceso y giro a la dcha.
    GOSUB Turn_Right
  ELSEIF (IN7 = 0) THEN                ' Bigote dcho. activado
    GOSUB Back_Up                      ' Retroceso y giro a la izda.
    GOSUB Turn_Left
  ELSE                                 ' Ambos bigotes desactivados (no hay obstáculos)
    GOSUB Forward_Pulse                ' Avance
  ENDIF                                ' Chequear de nuevo estado de los bigotes
LOOP

' -----[ Subrutinas ]
Forward_Pulse:                         ' Envía un simple pulso de avance.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                             ' Giro a la izda. 90º.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20             ' Giro a la dcha. 90º.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN

Back_Up:                               ' Retroceso.
  FOR pulseCount = 0 TO 40
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN
```

Las declaraciones IF...THEN en el programa principal comprueban el estado de los bigotes. Si los dos bigotes están presionados (IN5 = 0, IN7 = 0), el robot realizará un giro en U llamando a la subrutina Back_Up seguida de la rutina Turn_Left (dos veces). Si sólo el bigote izquierdo está presionado (IN5 = 0), el programa ejecutara la subrutina Back_Up seguido por la subrutina Turn_Right. Si es el bigote derecho el que está presionado, el programa ejecutará la rutina Back_Up y seguidamente llamará a la rutina Turn_Left. La única posibilidad que no está resuelta en el programa es cuando los dos bigotes no están presionados (IN5 = 1, IN7 = 1). En ese caso el comando ELSE llama a la rutina Forward_Pulse.

```
IF (IN5  = 0) AND (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 ya son conocidas, pero la subrutina Forward_Pulse tiene un detalle que debe ser explicado: sólo envía un pulso, después devuelve el control al programa principal. Esto es importante, porque significa que el Home Boe-Bot puede comprobar sus bigotes entre cada pulso que le hace avanzar.

```
Forward_Pulse:
  PULSOUT 12,650
  PULSOUT 13,850
  PAUSE 20
  RETURN
```

Si no se hiciera así, en caso de que el robot entrara en la rutina de Forward_Pulse, ya no volvería a comprobar los bigotes. Con ese único pulso, el Home Boe-Bot avanza medio centímetro y vuelve a comprobar los bigotes. En caso de no detectar nada, seguirá avanzando según las instrucciones del programa principal.

Los parámetros del bucle FOR...NEXT en las rutinas Back_Right y Back_Left pueden ser modificados para variar el ángulo de giro del Home Boe_Bot.

Modifica el parámetro EndValue del bucle FOR...NEXT en las rutinas de movimiento del programa RoamingWithWhiskers.bs2.

También puedes cambiar las declaraciones IF...THEN para que los LED empleados en la experiencia anterior indiquen la maniobra que está realizando el Boe-Bot. Aquí tienes un ejemplo:

```
IF (IN5  = 0) AND (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
```

6.5. EXPERIENCIA #4: INTELIGENCIA ARTIFICIAL. DECIDIENDO QUÉ HACER CUANDO SE BLOQUEA EL HOME BOE-BOT EN LAS ESQUINAS

Probablemente ya te habrás dado cuenta de que el Home Boe-Bot se queda bloqueado en las esquinas. Cuando encuentra una esquina y el bigote izquierdo choca contra la pared, gira a la derecha, pero entonces es el bigote derecho el que choca contra la otra pared, quedándose maniobrando en un bucle infinito del que no sabe como salir.

El programa RoamingWithWhiskers puede ser modificado para detectar este problema de bloqueo en las esquinas y actuar en consecuencia. El truco está en contar y recordar cuantas veces han chocado los bigotes de forma alternativa . El valor de este contaje es comparado con un valor previamente decidido, y si el número de choques es menor al valor, se añade uno más. Si dicha cuenta es igual o mayor que el valor decidido previamente, es el momento de hacer un giro en U y resetear o inicializar el contador de choques alternos.

El siguiente programa te enseña como se jerarquizan las declaraciones IF...THEN. En otras palabras, el programa comprueba una de las condiciones, y si es verdadera, pasa a comprobar la otra condición dentro de la primera. Aquí tienes un ejemplo en pseudo código.

```
ELSE
        Comandos que son
        ejecutados cuando no
        se cumple la condición
ENDIF
```

En el siguiente programa tienes un ejemplo real.

Programa EscapingCorners.bs2

- Teclea, salva y ejecuta el programa EscapingCorners.bs2, que hará que tu Home Boe-Bot ejecute un giro en U al cuarto o quinto choque alterno de los bigotes, dependiendo de cual es el que se ha presionado primero.

```
' El Robot Home Boe-Bot - EscapingCorners.bs2
' El robot navega tratando de salir de las esquinas que se encuentre,
' detectando los bigotes que se activan alternativamente
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ Variables ]
pulseCount VAR Byte          ' Contador del bucle For...Next...
counter VAR Nib              ' Contador de contactos.
old7 VAR Bit                 ' Almacena estado previo de IN7.
old5 VAR Bit                 ' Almacena estado previo de IN5.

' -----[ Inicialización ]
FREQOUT 4, 2000, 3000        ' Señal de inicio/reset.
counter = 1                  ' Inicia contador de contactos.
old7 = 0                     ' Valor inicial.
old5 = 1
```

```
' -----[ Rutina principal ]
DO

' --- Detecta esquinas alternativas y consecutivas
  IF (IN7 <> IN5) THEN                      ' Uno de los bigotes se ha activado.
    IF (Old7 <> IN7) AND (Old5 <> IN5) THEN ' Es diferente al previo.
      counter = counter + 1                 ' Incrementa contador contactos count + 1.
      old7 = IN7                            ' Registra el estado del bigote
      old5 = IN5                            ' for next comparison.
      IF (counter > 4) THEN                 ' Si se han registrado 4 contactos
        counter = 1                         'el contador se pone a 0
        GOSUB Back_Up                       ' y ejecuta un giro en forma de U.
        GOSUB Turn_Left
        GOSUB Turn_Left
      ENDIF          ' ENDIF counter > 4.
    ELSE             ' ELSE (old7=IN7) o (old5=IN5),
      counter = 1                           ' no son alternativos, inicia contador.
    ENDIF            ' ENDIF (old7<>IN7) and
                                            ' (old5<>IN5).
  ENDIF              ' ENDIF (IN7<>IN5).

' --- Las mismas rutinas de navegación que en RoamingWithWhiskers.bs2
  IF (IN5 = 0) AND (IN7 = 0) THEN           ' Ambos bigotes detectan obstáculo
    GOSUB Back_Up                           ' Retroceso y giro en U
    GOSUB Turn_Left
    GOSUB Turn_Left
  ELSEIF (IN5 = 0) THEN                     ' Contacto en el bigote izdo.
    GOSUB Back_Up                           ' Retroceso y giro a la dcha.
    GOSUB Turn_Right
  ELSEIF (IN7 = 0) THEN                     ' Contacto en el bigote dcho.
    GOSUB Back_Up                           ' Retroceso y giro a la izda.
    GOSUB Turn_Left
  ELSE                                      ' Ningún bigote detecta obstáculo
    GOSUB Forward_Pulse                     ' Avanza un pulso
  ENDIF                                     ' Volver a chequear
LOOP

' -----[ Subroutinas ]
Forward_Pulse:                              ' Envía un pulso de avance.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                                  ' Giro a la izda. 90º.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20                  ' Giro a la dcha 90º.
    PULSOUT 13, 850
    PULSOUT 12, 850
```

```
  PAUSE 20
 NEXT
 RETURN

Back_Up:                                    ' Retroceso.
 FOR pulseCount = 0 TO 40
  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20
 NEXT
 RETURN
```

El programa EscapingCorners.bs2 es una modificación del RoamingWithWhiskers.bs2 por lo que sólo se explicarán las variaciones relacionadas con la detección y desbloqueo en las esquinas.

Se han creado tres nuevas variables para detectar las esquinas. La variable counter (de tipo nibble) puede almacenar valores entre 0 y 15 porque es una variable de 4 bits cuyo valor máximo es 15 en decimal.. Es un tamaño razonable, ya que nuestro valor máximo para detectar que estamos en una esquina es 4. Las otras dos variables (old7 y old8) son de tipo bit, es decir, que sólo pueden almacenar los valores 1 o 0. Tienen el tamaño adecuado, ya que servirán para almacenar los valores "viejos" de IN7 e IN5, que también son de tipo bit.

```
counter     VAR   Nib
old7        VAR   Bit
old5        VAR   Bit
```

Estas variables deben ser inicializadas. La variable counter se inicializa a 1, y cuando llegue a 4 se reinicializará de nuevo a 1. Las otras dos variables deben ser inicializadas de tal forma que parezca que uno de los dos bigotes fue presionado antes de iniciar el programa. Da igual cuál sea la que esté a 1, siempre que la otra esté a 0.

```
Counter = 1
old7    = 0
old5    = 1
```

Ahora pasamos a la sección de "Detectar Esquinas Alternas Consecutivas"!. Lo primero que debemos comprobar es que uno de los bigotes ha sido presionado. Una manera muy simple es preguntar si el valor de IN7 es diferente al de IN5.

```
IF (IN7 <> IN5) THEN
```

Si es cierto que uno de los bigotes ha sido presionado, a continuación se debe comprobar si dicho bigote es el mismo que la vez anterior o si es el otro. Es aquí donde participan las variables old7 y old5.

```
IF (old7 <> IN7) AND (old5 <> IN5) THEN
    counter = counter + 1
    old7 = IN7
    old5 = IN5
```

Cuando se cumple lo anterior significa que ya se han producido cuatro contactos consecutivos de los bigotes, por lo que hay que resetear el contador, o sea, ponerlo a 1 y ejecutar un giro en U.

```
IF (counter > 4) THEN
    counter = 1
    GOSUB Back_Up
    GOSUB Turn_Left
    GOSUB Turn_Left
```

El siguiente ELSE va en la declaración IF (old7 <> IN7) AND (old5 <> IN5) THEN y cubre la posibilidad de que la condición del IF no sea cierta. En otras palabras, que no se hayan producido los cuatro choques alternativos y por tanto el Home Boe-Bot no está en una esquina.

```
ELSE
    counter = 1
```

Te proponemos que practiques con las siguientes operaciones.

- Incrementa el valor de repetición a 5 o 6 y prueba el efecto.
- Prueba también a reducirlo y comprueba si se comporta igual que en una exploración normal.

6.6 PRUEBA DE AUTOEVALUACION

Cuestiones

1. ¿Qué clase de dispositivo es un bigote?
2. ¿Qué hace el formateador CRSRXY del DEBUG? ¿Qué dos valores deben acompañar a este formateador?
3. Cuando un bigote es presionado, ¿qué voltaje recibe el pin de E/S que lo está monitorizando? ¿Qué valor binario recibe el correspondiente biestable interno?. Si el pin de E/S P8 se utiliza para monitorizar el pin de entrada, ¿qué valores tiene IN8 cuando un bigote esta presionado o no?
4. ¿Qué bigote está conectado a IN5? ¿Y a IN7? Si IN7 = 1, ¿qué significa? ¿Y si IN7 = 0? ¿Y que hay de IN5 = 1 e IN5 = 0?
5. ¿Qué comando se utiliza para saltar a las subrutinas dependiendo del valor de una variable? ¿Qué comando se utiliza para decidir a que subrutina se debe saltar? ¿En que se basan estas decisiones?
6. ¿Qué tres técnicas de programación en PBASIC se usan en este capítulo para capturar eventos y tomar decisiones basadas en los mismos?

Ejercicios

1. Escribe en una nueva línea un comando DEBUG para el TestWhiskers.bs2 que actualice el estado de los bigotes. Ajusta el comando PAUSE de 50 a 250.
2. ¿Cuál es el nuevo ratio de muestreo que pusiste en el ejercicio 1? Pista: El ratio de muestreo es el número de veces por segundo que se comprueba el estado de los bigotes. Se puede calcular dividiendo 1 entre el tiempo que pasa entre cada muestreo.
3. En el programa RoamingWithWhiskers.bs2 determina el ratio de muestreo cuando el Home Boe-Bot está avanzando y cuando está maniobrando.
4. Utilizando el programa RoamingWithWhiskers.bs2 como referencia, escribe una subrutina llamada Turn_Away que llame una vez a Back_Up y dos veces a Turn_Left. Escribe también las modificaciones que deberías hacer al programa principal de RoamingWithWhiskers.bs2.

Proyectos

1. Modifica el programa RoamingWithWhiskers.bs2 para que el Home Boe-Bot emita un pitido de 4 kHz durante 100 ms antes de ejecutar una maniobra evasiva. Haz que el pitido sea doble si han sido los dos bigotes los que han tropezado con algún obstáculo.
2. Modifica RoamingWithWhiskers.bs2 para que los LED parpadeen cuando el Home Boe-Bot esté ejecutando una maniobra.
3. Lo ideal sería que el Home Boe-Bot recorriera la mayor distancia en el menor tiempo posible. Modifica el RoamingWithWhiskers.bs2 para que el robot ejecute giros de 45º en una habitación con unos pocos obstáculos de tamaño variable. Repite el proceso con giros de 30º, 60º, 90º y 120º. Prueba también a ver cuanta distancia es capaz de recorrer el Home Boe-Bot marcha atrás.

4. *Proyecto avanzado* – Modifica el programa RoamingWithWhiskers.bs2 de forma que el robot describa un círculo de 1 metro de diámetro. Cuando se active uno de los bigotes, el Home Boe-Bot describirá un círculo de menor diámetro, mientras que si se activa el otro el círculo será mayor.

Tema 7

Caminando hacia la luz

7.1. SENSORES DE LUZ Y POSIBLES APLICACIONES

La luz tiene muchas aplicaciones en la robótica y el control industrial. Se citan algunos ejemplos como la detección del borde de un rollo de tela en la industria textil, el encendido de las farolas según la época del año, el nivel de luminosidad al tomar una fotografía, el momento adecuado para regar las plantas de un invernadero...

Hay muchos tipos de sensores de luz dedicados a funciones concretas. El sensor de luz que utiliza tu Home Boe-Bot está diseñado para detectar la luz visible y puede servir para medir el nivel de luminosidad. Con esta capacidad, tu robot puede ser programado para reconocer áreas con luz o perímetros oscuros, informando de los niveles de luminosidad que detecte, y buscar así la salida de una habitación navegando hacia la luz que entra por su puerta.

Las resistencias manejadas hasta ahora tenían un valor fijo que venía marcado en su cuerpo mediante franjas de colores. Las foto resistencias, también llamadas LDR (Resistencias Dependientes de la Luz), tienen un valor óhmico que varía en función de la luz (brillo o luminosidad) que incide sobre su superficie en cada momento. Ver la figura 7-1. La LDR tiene poca resistencia cuando está muy iluminada, mientras que si está en la oscuridad puede superar los 50 KΩ.

Figura 7-1.- *Símbolo eléctrico y apariencia de una LDR, que es una resistencia cuyo valor depende de la luz que recibe.*

7.2. EXPERIENCIA #1: MONTANDO Y PROBANDO LOS CIRCUITOS DE LAS FOTO RESISTENCIAS

Los circuitos con foto resistencias LDR serán capaces de diferenciar entre una sombra y la luz normal. Los comandos de PBASIC que utilizarás son similares a los manejados para comprobar si un bigote estaba presionado o no.

Para el montaje de esta experiencia se necesitan los siguientes materiales.

(2) Foto resistencias (LDR)
(2) Resistencias de 2 kΩ
(2) Resistencias de 220 Ω
(4) Cables
(2) Resistencias de 470 Ω
(2) Resistencias de 1 kΩ
(2) Resistencias de 4k7 Ω
(2) Resistencias de 10 kΩ

Montando los ojos foto sensitivos del robot

La figura 7-2 muestra el esquema, y la 7-3 la apariencia del conexionado de los circuitos de las foto resistencias que utilizarás en esta y en las dos próximas experiencias.

- Desconecta las pilas de la placa y los servos.
- Monta el circuito de la figura 7-2 tomando como referencia la figura 7-3.

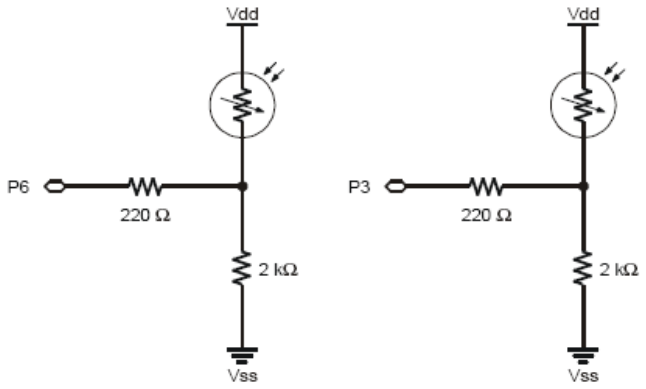


Figura 7-2.- *Esquema eléctrico para el conexionado de las LDR.*

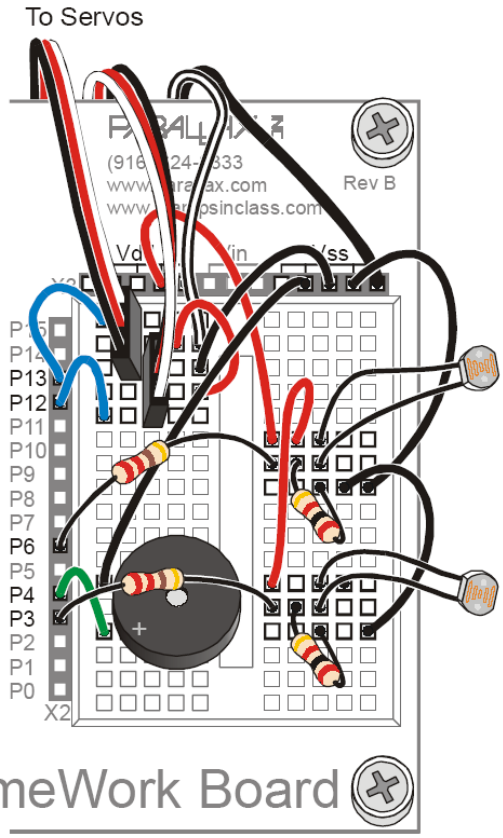


Figura 7-3.- *Apariencia del montaje de los circuitos LDR.*

Cómo funciona el circuito con las foto resistencias

Un pin de E/S de la Home Work puede funcionar tanto como entrada como salida. Cuando lo hace como salida, el pin puede enviar señales de nivel alto (5 V) o bajo (0 V). Estas señales pueden utilizarse para controlar los LED, los servos, el zumbador, etc. Cuando el pin funciona como entrada lo que haces es "escuchar". Si el pin de entrada detecta un voltaje superior a 1'4 V almacena un 1 lógico en el biestable correspondiente. Si el voltaje es inferior almacenará un 0. En el tema anterior estos biestables de entrada almacenaban valores que indicaban cuando los bigotes eran presionados.

En la figura 7-4 se muestra el circuito que se conecta a una patita de entrada de la Home Work cuando hay una LDR. Este circuito es un "divisor de tensión" formado por dos resistencias en serie: la LDR y otra de 2kΩ. Si la LDR está recibiendo la luz de una fluorescente su resistencia interna tiene un valor R = 1 kΩ, aproximadamente. Si por el contrario se halla situada debajo de la sombra de un objeto, R = 25 kΩ. La tensión Vo que se aplica a la patita de entrada de la Home Work depende del valor de la LDR. Cuanto más resistencia tenga la LDR menor valor tendrá Vo y viceversa.

Si Vo se conecta a la patita 6 de E/S (IN6) cuando Vo tiene un voltaje superior a 1,4 v, el biestable de IN6 almacena un nivel lógico 1 y si es inferior a 1,4 V almacena un 0.

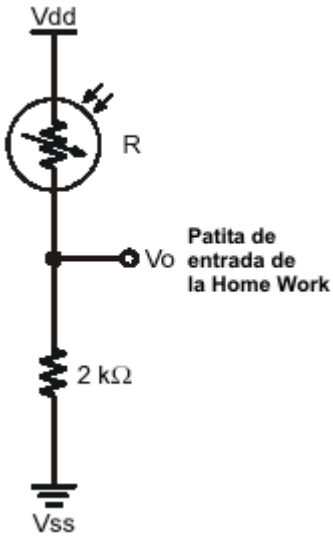


Figura 7-4.- *Circuito de conexionado de la LDR con una patita de entrada de la Home Work.*

Detectando sombras

Al pasar bajo una sombra, la resistividad de la foto resistencia (R) aumenta, lo que hace que el valor de Vo disminuya. La resistencia de 2 kΩ hace que el valor de Vo esté ligeramente por encima del umbral de 1'4 V incluso cuando se halla dentro de una habitación bien iluminada. Si haces sombra sobre el Home Boe-Bot con tu mano Vo tomará un valor inferior al umbral de 1'4 V. Los dos circuitos con las LDR se conectan a las patitas 6 y 3 de la Home Work.

En una habitación bien iluminada tanto IN6 como IN3 almacenarán el nivel 1. Si existe una sombra sobre la fotorresistencia conectada a P6, entonces almacenarán un 0. Asimismo, si la sombra está sobre la fotorresistencia conectada a P3, sólo IN3 almacenará un 0.

Programa TestPhotorresistorsDividers.bs2

Este programa es el TestWhiskers.bs2 adaptado a las foto resistencias. En vez de monitorizar P5 y P7 como hacíamos para los bigotes, monitorizaremos P3 y P6, que están conectados a los circuitos de las foto resistencias. Este programa debería visualizar un 1 en ambas patitas si estamos en una habitación bien iluminada. Si hay una sombra sobre alguna de las foto resistencias, su correspondiente valor deberá cambiar a 0.

- Conecta la alimentación a la placa.
- Teclea, salva y ejecuta TestPhotorresistorsDividers.bs2.
- Verifica que cuando no hay sombras tanto IN6 como IN3 almacenan el valor 1.
- Verifica que si pasas la mano sobre una fotorresistencia creando sombra, el registro correspondiente almacena un 0.

```
' El robot Home Boe-Bot - TestPhotoresistorDividers.bs2
' Visualizar el valor del voltage detectado por las foto
' resistencias conectadas a las líneas de E/S.
' {$Stamp bs2}
' {$PBASIC 2.5}

DEBUG "ESTADO DE LAS FOTO RESISTENCIAS", CR,
    "Izda.       Dcha.",   CR,
    "-------     --------"
DO
  DEBUG CRSRXY, 0, 3,
     "P6 = ", BIN1 IN6,
     "      P3 = ", BIN1 IN3
  PAUSE 100
LOOP
```

Dependiendo de la iluminación en el área de acción del robot, deberás poner unas resistencias superiores o inferiores a 2 kΩ para que el Home Boe-Bot pueda detectar las sombras.

- Recuerda que debes desconectar la alimentación de la tarjeta cuando estés modificando los circuitos.
- Cambia las resistencias de 2 kΩ por otras con los siguientes valores: 470 Ω, 1 kΩ, 4'7 kΩ y 10 kΩ.
- Prueba el TestDigitalPhotorresistors.bs2 para averiguar con que resistencia trabaja mejor el Home Boe-Bot bajo esas condiciones de iluminación.
- Utiliza la combinación de resistencias que tu creas más conveniente en las dos siguientes experiencias.

7.3. EXPERIENCIA #2: DETECTANDO Y ESQUIVANDO SOMBRAS COMO SI FUERAN OBJETOS

Vamos a adaptar el programa RoamingWithWhiskers.bs2 del tema anterior para que funcione con las foto resistencias. Todo lo que tienes que hacer es ajustar las condiciones del IF...THEN para que monitoricen IN6 e IN3 en vez de IN7 e IN5. La figura 7-5 muestra como hacerlo:

```
' From RoamingWithWhiskers.bs2

IF (IN5  = 0) AND (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
```

```
' Modified for
' RoamingWithPhotoresistor
' Dividers.bs2

IF (IN6  = 0) AND (IN3 = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
  GOSUB Turn_Left
ELSEIF (IN6  = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Right
ELSEIF (IN3  = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
ELSE
  GOSUB Forward_Pulse
ENDIF
```

Figura 7-5.- ***Adaptación del programa RoamingWithWhiskers.bs2 para las LDR.***

Programa RoamingPhotoresistorDividers.bs2

- Abre el programa RoamingWithWhiskers.bs2 y sálvalo como
- RoamingPhotoresistorDividers.bs2.
- Haz la modificaciones indicadas en la figura 7-5.

- Conecta las pilas a la placa y los servos.
- Ejecuta y prueba el programa.
- Verifica que el **Home** Boe-Bot evita las sombras (Utiliza las manos para provocar las mismas). Haz las pruebas sin sombras, poniendo una sobre la fotorresistencia derecha (conectada a P3), poniéndo otra sobre la izquierda (conectada P7), y colocando dos sombras sobre ambas foto resistencias.
- Vuelve a escribir los comentarios, título, etc de tal forma que quede bien reflejado que este programa es para las foto resistencias y no para los bigotes. Compila el código de nuevo cuando hayas terminado.

```
' -----[ Título ]-------------------------------------------------------------
' El robot Home Boe-Bot - RoamingPhotoresistorDividers.bs2
' El Boe-Bot detecta sombras mediante la tensión que proporcionan las foto
' resistencias y actúa en función de ellas

' {$Stamp bs2}
' {$PBASIC 2.5}

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

pulseCount VAR Byte                     ' Contador para el bucle For..next.

' -----[ Inicialización ]------------------------------------------------------

FREQOUT 4, 2000, 3000                   ' Señal de Inicio/reset

' -----[ Rutina principal ]----------------------------------------------------
DO
IF (IN6 = 0) AND (IN3 = 0) THEN          ' Ambas LDR detectan sombras
  GOSUB Back_Up                         ' retroceso y giro a la izda.
  GOSUB Turn_Left
  GOSUB Turn_Left
ELSEIF (IN6 = 0) THEN                   ' La LDR izda. detecta sombra,
  GOSUB Back_Up                        ' retroceso y giro a la dcha.
  GOSUB Turn_Right
ELSEIF (IN3 = 0) THEN                   ' La LDR dcha detecta sombra,
  GOSUB Back_Up                        ' retroceso y giro a la izda.
  GOSUB Turn_Left
ELSE                                   ' Ninguna LDR detecta sombras,
  GOSUB Forward_Pulse                  ' avance
ENDIF

LOOP

' -----[ Subrutinas ]----------------------------------------------------------

Forward_Pulse:                         ' Envía un pulso de avance.
  PULSOUT 12,650
  PULSOUT 13,850
  PAUSE 20
  RETURN

Turn_Left:                             ' Giro a la izda. 90º.
  FOR pulseCount = 0 TO 20
    PULSOUT 12, 650
    PULSOUT 13, 650
    PAUSE 20
    NEXT
```

```
  RETURN

Turn_Right:
 FOR pulseCount = 0 TO 20              ' Giro a la dcha. 90º.
  PULSOUT 12, 850
  PULSOUT 13, 850
  PAUSE 20
  NEXT
  RETURN

Back_Up:                               ' Retroceso.
 FOR pulseCount = 0 TO 40
  PULSOUT 12, 850
  PULSOUT 13, 650
  PAUSE 20
  NEXT
  RETURN
```

Puedes mejorar el rendimiento de tu robot colocando algunas de las llamadas a subrutina diseñadas para ayudar al Home Boe-Bot a retroceder ante obstáculos y girar para evitarlos. La figura 7-6 muestra un ejemplo en el que dos llamadas a la subrutina Turn_Left son usadas en el cuerpo del IF...THEN donde la condición es que ambas foto resistencias detecten una sombra. De esta forma, si solo una de las foto resistencias detecta la sombra, las llamadas a la subrutina Back_Up son aplicadas y el Home Boe-Bot girará en respuesta a la sombra.

```
' Excerpt from
' RoamingWithPhotoresistor
' Dividers.bs2

IF (IN6  = 0) AND (IN3 = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
  GOSUB Turn_Left
ELSEIF (IN6  = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Right
ELSEIF (IN3  = 0) THEN
  GOSUB Back_Up
  GOSUB Turn_Left
ELSE
  GOSUB Forward_Pulse
ENDIF
```

```
' Modified excerpt from
' RoamingWithPhotoresistor
' Dividers.bs2

IF (IN6  = 0) AND (IN3 = 0) THEN
  GOSUB Back_Up
'   GOSUB Turn_Left
'   GOSUB Turn_Left
ELSEIF (IN6  = 0) THEN
'   GOSUB Back_Up
  GOSUB Turn_Right
ELSEIF (IN3  = 0) THEN
'   GOSUB Back_Up
  GOSUB Turn_Left
ELSE
  GOSUB Forward_Pulse
ENDIF
```

Figura 7-6.- *Programa para hacer girar al robot frente a las sombras.*

- Modifica RoamingWithPhotoresistorDividers.bs2 como se muestra en la parte derecha de la figura 7-6.
- Ejecuta el programa y comprueba el resultado.

7.4. EXPERIENCIA #3: PERSIGUIENDO A LAS SOMBRAS

Eliminando los bucles FOR...NEXT de las subrutinas de movimiento puedes conseguir que tu Home Boe-Bot sea más eficiente. Esto no era posible con los bigotes, ya que el robot tenía que retroceder antes de girar cuando se había encontrado con un obstáculo físico. Cuando utilizas las sombras para guiar al Home Boe-Bot, puedes comprobar entre cada pulso si la sombra ha sido ya detectada, sin importar si está moviéndose hacia delante o ejecutando una maniobra.

Una manera de controlar remotamente el Home Boe-Bot es hacer que esté parado bajo la luz normal, pero que siga una sombra en el momento que la detecte. Si creas una sombra con tu mano sobre la fotorresistencia podrás hacer que el robot te siga los movimientos que hagas con ella.

Programa ShadowGuideBoeBot.bs2

Cuando ejecutes el siguiente programa, el Home Boe-Bot debería quedarse quieto cuando no haya ninguna sombra sobre sus foto resistencias. Si creas una sombra que afecte a las dos foto resistencias el Home Boe-Bot debería moverse hacia delante. Si la sombra está sólo sobre una de las foto resistencias, el Home Boe-Bot debería girar en la dirección en que la fotorresistencia ha detectado la sombra.

- Teclea, salva y ejecuta ShadowGuidedBoeBot.bs2
- Usa tus manos para crear sombras sobre las foto resistencias.
- Estudia el programa atentamente y asegúrate que entiendes como trabaja. Es muy corto pero muy potente.

```
' El robot Home Boe-Bot - ShadowGuidedBoeBot.bs2
' El robot detecta sombras y trata de seguirlas.

' {$Stamp bs2}
' {$PBASIC 2.5}

FREQOUT 4, 2000, 3000              ' Señal de Inicio/reset.

DO                                 ' Ambas LDR detectan sombras, avance.

 IF (IN6 = 0) AND (IN3 = 0) THEN   ' Ambas LDR detectan sombras, avance.
   PULSOUT 13, 850
   PULSOUT 12, 650
 ELSEIF (IN6 = 0) THEN             ' Sombra a la izda.,
   PULSOUT 13, 750                 ' giro a la izda.
   PULSOUT 12, 650
 ELSEIF (IN3 = 0) THEN             ' Sombra a la dch.,
   PULSOUT 13, 850                 ' giro a la dcha.
   PULSOUT 12, 750
 ELSE
   PULSOUT 13, 750                 ' No hay sombras, stop
   PULSOUT 12, 750
 ENDIF

 PAUSE 20                          ' Pausas entre pulsos.

LOOP
```

El bloque IF...THEN dentro del DO...LOOP busca una de las cuatro posibles condiciones de sombra: ambas, izquierda, derecha o ninguna. Dependiendo de que condición es detectada, los comandos PULSOUT determinan los pulsos a enviar para ejecutar la maniobra correspondiente: avanzar, girar a la derecha, girar a la izquierda o quedarse quieto. Después del bloque IF...THEN, es importante acordarse de incluir la instrucción PAUSE 20 para asegurarse de que se da a tiempo a los servos a recibir todos los pulsos que les son enviados.

Este programa no necesita la condición del ELSE ni los dos comandos PULSOUT que la siguen. Si decides no enviar pulsos, el Home Boe-Bot estará parado igual que si ejecutaras un PULSOUT 750.

- Intenta borrar o comentar este bloque de código:

```
ELSE
        PULSOUT 13, 750
        PULSOUT 12, 750
```

- Ejecuta el programa modificado.
- ¿Notas alguna diferencia en el comportamiento del Home Boe-Bot?

7.5. EXPERIENCIA #4: MIDIENDO EL NIVEL DE LUZ

La única información que la Home Work es capaz de tomar de las foto resistencias es si el nivel de luz está por encima o debajo del umbral predeterminado. Se propone un nuevo circuito con el que la Home Work puede monitorizar y determinar niveles de iluminación relativos. El valor que producirá este circuito oscilará entre números muy pequeños (luz muy brillante) y muy grandes (poca luz). Esto significa que no tendrás que volver a cambiar manualmente las resistencias dependiendo del nivel de luz. En vez de eso, podrás ajustar tu programa para buscar diferentes umbrales de transición de 0 a 1.

El Condensador

Un condensador es un dispositivo que almacena carga eléctrica y es indispensable en la mayoría de los circuitos. La cantidad de carga que un condensador es capaz de almacenar se mide en faradios (F). Un faradio es un valor muy grande. Los condensadores que utilizarás en esta actividad almacenan fracciones de millonésimas partes de faradio. Una millonésima de faradio es un microfaradio, y su abreviatura es μF. El condensador que utilizarás en este ejercicio almacena una centésima de millonésima de faradio, es decir, 0.01 μF.

La figura 7-7 muestra el símbolo esquemático para un condensador de 0.01 μF junto a un dibujo del condensador que viene con tu kit del Home Boe-Bot. La marca 103 indica su valor. Dicha marca expresa el valor del condensador en nanofaradios, que representan la millonésima parte del μF. Así, 103 indica que al 10 inicial hay que añadir tres ceros para obtener el valor de 10.000 nF ó 0.01 μF.

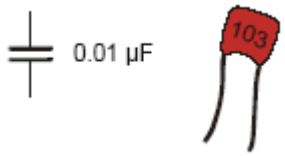


Figura 7-7.- *Símbolo eléctrico y apariencia de un condensador de 0,01 μF.*

Para este nuevo circuito se precisan los siguientes materiales:

(2) Foto resistencias (LDR)
(2) Condensadores – 0'01 μF (103)
(2) Resistencias – 220 Ω
(2) Cables

Reconstruyendo los ojos sensibles a la luz

El circuito básico que se utiliza para determinar el nivel de luz se llama circuito resistencia/condensador (RC). La figura 7-8 muestra los esquemas de los circuitos RC para detección de luz del Home Boe-Bot, y la figura 7-9 muestra una imagen del conexionado.

Para medir el nivel de luz que existe sobre la LDR nos basamos en el tiempo que un condensador tarda en descargarse por una resistencia que se coloca en paralelo con él. Dicha resistencia será la LDR y cuanto mayor valor tenga más tardará en descargarse el condensador.

- Desconecta las pilas de la placa y los servos.
- Monta los circuitos que se muestran en la figura 7-8 utilizando como referencia la figura 7-9.

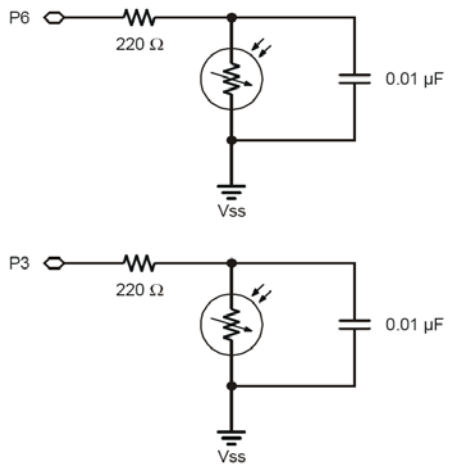


Figura 7-8.- *Esquema eléctrico de los circuitos de detección del nivel de luz. El condensador se descarga a través de la LDR en paralelo*

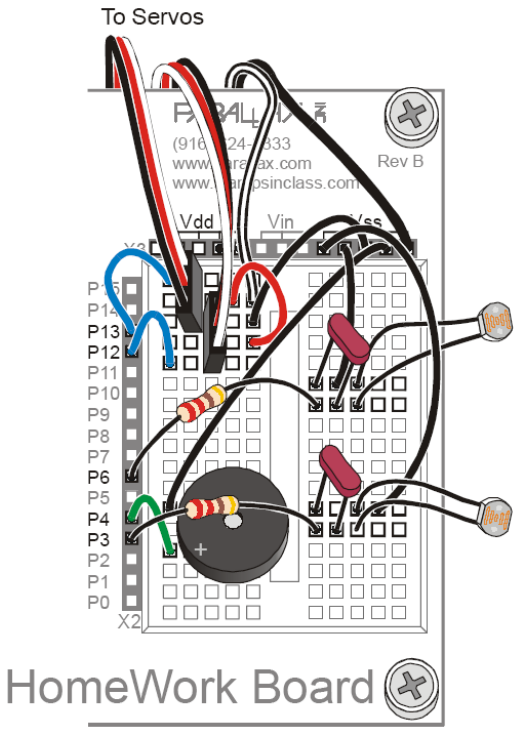


Figura 7-9.- *Apariencia del montaje de los circuitos detectores del nivel de luz.*

Piensa que el condensador del circuito mostrado en la figura 7-10 funciona como una pequeña batería recargable. Cuando P6 envía una señal de nivel alto, carga el condensador al aplicarle 5 V. Tras unos pocos ms, el condensador alcanza casi 5 V. Si el programa cambia en este momento la configuración de la patita P6 y la hace entrada, el condensador irá perdiendo su carga a través de la fotorresistencia. Debido a esto, el voltaje decrece. La cantidad de tiempo que tarda el voltaje que recibe IN6 en quedar por debajo de 1'4 V depende del valor óhmico de la fotorresistencia por la que se descarga el condensador. Si la fotorresistencia tiene una alta resistividad debido a unas condiciones de luz muy débiles, el condensador tardará más en descargarse. Si la fotorresistencia tiene una baja resistividad por recibir una luz muy brillante, el condensador se descargará rápidamente. El tiempo de descarga del condensador depende de la luz que incide sobre la LDR.

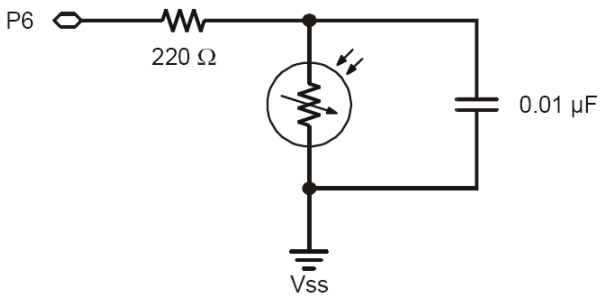


Figura 7-10.- *Primero P6 es salida y saca un 1 para cargar al condensador. Luego P6 es entrada y recibe el voltaje del condensador que va decreciendo al irse descargando a través de la LDR.*

Midiendo el tiempo de descarga del circuito RC

La Home Work puede ser programada para cargar el condensador y después medir cuanto tiempo tarda en descargarse hasta llegar a 1'4 V. Esta medición del tiempo de descarga puede usarse para determinar el nivel o intensidad de la luz detectada por la fotorresistencia. Para esta medición se utilizan los comandos HIGH, PAUSE, y el nuevo RCTIME. Este último, el RCTIME, se utiliza para medir el tiempo de descarga en un circuito como el mostrado en la figura 7-10. La sintaxis del RCTIME es:

RCTIME Pin, State, Duration

El atributo Pin es el número de pin de E/S que quieres medir. Por ejemplo, si quieres medir P6, el atributo Pin sería 6. El atributo State puede ser 1 o 0. Sería 1 si el voltaje del condensador comienza siendo mayor que 1'4 V y va bajando, o 0 si es menor que 1'4 V y va subiendo. Para el circuito de la figura 7-10 el voltaje empezará siendo prácticamente 5 V e irá bajando hasta 1'4 V, por lo que State sería 1. El atributo Duration es una variable que almacena el tiempo de descarga medido en unidades de 2 µs. En el siguiente ejemplo, mediremos el tiempo de descarga del circuito RC formado por el condensador y la LDR conectada a P6 (la foto resistencia izquierda del Home Boe-Bot). Para guardar dicho tiempo usaremos una variable llamada timeLeft.

Lo primero que debes hacer para medir el tiempo de descarga del circuito RC es asegurarte de que has declarado la variable que almacenará dicho tiempo:

```
timeLeft        VAR     Word
```

Las tres siguientes líneas de código cargan el condensador, miden el tiempo de descarga y lo almacenan en la variable timeLeft:

```
HIGH 6                  'Inicio carga del condensador
PAUSE 3                 'Retardo dar tiempo a la carga
RCTIME 6,1,timeLeft     `Medición del tiempo de descarga
```

Para tomar la medida debes implementar el código en estos tres pasos:

- Comienza por cargar el condensador conectando el circuito a 5 V (utiliza para ello el comando HIGH).
- Utiliza el comando PAUSE para darle suficiente tiempo a HIGH para que cargue el condensador del circuito RC.
- Ejecuta el comando RCTIME, que configura el pin de E/S como entrada, mide el tiempo de descarga (desde casi 5 V hasta 1'4 V), y lo almacena en la variable timeLeft.

Programa TestP6Photoresistors.bs2

- Conecta la pila a tu placa.
- Teclea, salva y ejecuta TestP6Photorresistor.bs2.

- Crea una sombra sobre la fotorresistencia conectada a P6 y verifica que el tiempo de descarga del condensador es mayor cuanto más oscura es la luz que rodea al Home Boe-Bot.
- Aplica una luz brillante sobre la LDR. El tiempo de descarga del condensador debe ser pequeño.

```
' El robot Home Boe-Bot - TestP6Photoresistor.bs2
' Chequea la LDR conectada a P6 y visualiza el tiempo de descarga

' {$Stamp bs2}                    ' Stamp directive.
' {$PBASIC 2.5}                   ' PBASIC directive.

timeLeft VAR Word

DO
  HIGH 6
  PAUSE 2
  RCTIME 6,1,timeLeft

  DEBUG HOME, "timeLeft = ", DEC5 timeLeft
  PAUSE 100

LOOP
```

- Salva el TestP6Photoresistor.bs2 como TestP3Photoresistor.bs2.
- Modifica el programa para que mida el tiempo de descarga del condensador conectado a la fotorresistencia de la derecha (P3).
- Repite las pruebas hechas antes y verifica que el circuito funciona correctamente.

7.6. EXPERIENCIA #5: SIGUIENDO UN FOCO DE LUZ

En esta práctica probarás y calibrarás los sensores de tu Home Boe-Bot para que distingan entre luz ambiental y un foco de luz directa. Programarás tu robot para que siga un foco de luz que se mueva frente a él. Para esta experiencia se precisa una linterna.

Ajustando los sensores para que busquen el foco de luz

Esta práctica funcionará mejor si las foto resistencias apuntan hacia puntos un poco separados del Home Boe-Bot (unos 5 cm).

- Coloca las foto resistencias como se indica en la figura 7-11.

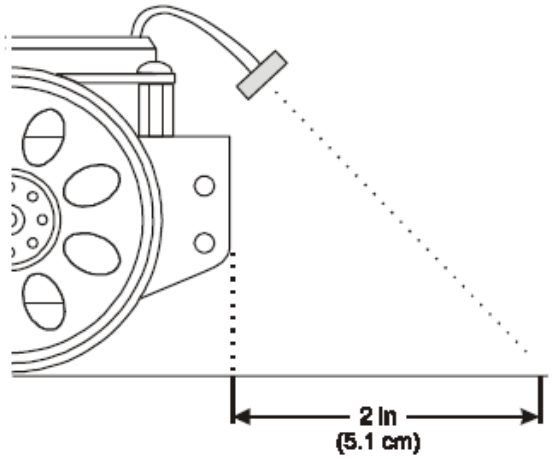


Figura 7-11.- *Procura que las LDR apunten a una distancia de unos 5 cm frente al Home Boe-Bot.*

Probando la respuesta de los sensores ante el foco de luz

Antes de programar el Home Boe-Bot para que siga el foco de luz, debes conocer las diferencias entre las lecturas de luz que existentes con y sin el foco de luz apuntando sobre su zona de detección.

Programa TestBothPhotoresistors.bs2

- Teclea, salva y ejecuta el programa TestBothPhotorresistors.bs2.
- Coloca el Home Boe-Bot en una superficie sobre la que pueda seguir el foco de luz. Asegúrate de que el cable serie está conectado y que las mediciones se muestran en el Debug Terminal.
- Anota los valores de las mediciones en la tabla 7-1.
- Enciende la linterna y apunta el foco de luz justo delante del robot.
- Tus mediciones ahora deberían ser significativamente menores que antes. Anota los nuevos valores en la tabla 7-1.

DURACION		DESCRIPCION
TimeLeft	TimeRight	
		Tiempo de descarga medido sin la linterna (luz ambiente)
		Tiempo de descarga medido aplicando la luz de la linterna frente al robot

Tabla 7-1.- *Tabla para rellenar con los tiempos de descarga corespondientes a la LDR izquierda y a la derecha cuando hay luz ambiente y cuando la luz procede de una linterna.*

```
' El robot Home Boe-Bot - TestBothPhotoresistors.bs2
' Chequea el circuito RC de las LDR del Boe-Bot.

' {$Stamp bs2}
' {$PBASIC 2.5}

timeLeft  VAR Word                      ' Declaración de variables.
timeRight VAR Word

DEBUG "VALORES DE LAS LDR", CR,    ' Inicialización.
      "LDR Izda. LDR Dcha.", CR,
      "--------- ---------"

DO                                      ' Rutina principal.

  HIGH 6                                ' Medida del tiempo RC izdo.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                ' Medida del tiempo RC dcho.
  PAUSE 3
  RCTIME 3,1,timeRight

  DEBUG CRSRXY, 0, 3,             ' Visualizar las medidas.
        DEC5 timeLeft,
    "     ",
    DEC5 timeRight

  PAUSE 100

LOOP
```

- Coloca el Home Boe-Bot para en diferentes posiciones y repite las mediciones.
- Para mejorar los resultados, haz la media de estos resultados con los anteriores y apúntalo en la tabla 7-1.

Siguiendo el foco de luz

Hasta ahora has estado utilizando declaraciones de variables. Por ejemplo, counter VAR Nib referencia con el nombre counter a un determinado espacio en la memoria RAM del microcontrolador. Después de declarar la variable, cada vez que utilizas counter en un programa PBASIC, estás usando ese determinado valor almacenado en la RAM.

También puedes declarar constantes. En otras palabras, si tienes un número que vas a emplear muy a menudo a lo largo del programa, puedes darle un nombre representativo. En lugar de utilizar la directiva VAR, deberás usar la directiva CON. Aquí tienes algunos ejemplos que se utilizarán para referenciar constantes:

```
leftAmbient     CON     108
rightAmbient    CON     114
leftBright      CON     20
rightBright     CON     22
```

Ahora, cada vez que en el programa aparezca el nombre leftAmbient, la Home Work tomará el valor 108. Actuará igual cuando aparezca rightAmbient, leftBright y rightBright, sustituyéndolos por 114, 20 y 22 respectivamente. Antes de ejecutar, sustituye estos valores por los que apuntaste en la tabla 7-1.

Las constantes también pueden ser utilizadas para calcular otras constantes. Aquí tienes un ejemplo de dos constantes llamadas leftThershold y rightThershold que se calculan a partir de las constantes que acabamos de declarar. Estas nuevas constantes se utilizan en el programa para saber cuando el foco de luz ha sido detectado.

```
'                               Media                       Factor de escala

LeftThreshold   CON     LeftBright + LeftAmbient        / 2 * 5 / 8
RightThreshold  CON     RightBright + RightAmbient      / 2 * 5 / 8
```

La operación matemática realizada es una media y posteriormente un escalado. El cálculo de la media viene dado por (leftBright + leftAmbient) / 2. El resultado obtenido es multiplicado por 5 y dividido por 8. Esto quiere decir que leftThershold es una constante cuyo valor es 5/8 de la media de leftBrigt y leftAmbient.

Programa FlashlightControlledBoeBot.bs2

- Carga el FlashlightControlledBoeBot.bs2 en el BASIC Stamp Editor.
- Sustituye tu medida timeLeft sin foco de luz (de la tabla 7-1) por el valor 108 de la directiva leftAmbient CON.
- Sustituye tu medida timeRight sin foco de luz (de la tabla 7-1) por el valor 114 de la directiva rightAmbient CON.
- Sustituye tu medida timeLeft con foco de luz (de la tabla 7-1) por el valor 20 de la directiva leftBright CON.
- Sustituye tu medida timeRight con foco de luz (de la tabla 7-1) por el valor 22 de la directiva rightBright CON.
- Conecta la alimentación a la placa y los servos.
- Salva y ejecuta FlashlightControlledBoeBot.bs2.
- Prueba y fíjate bien en que el Home Boe-Bot hace los giros y la maniobras en los momentos en que detecta el foco de luz.
- Utiliza la linterna para guiar al robot a través de un circuito con varios obstáculos.

```
' -----[ Título ]-------------------------------------------------------------
' El Robot Home Boe-Bot - FlashlightControlledBoeBot.bs2
' El robot trata de seguir el foco de luz.

' {$Stamp bs2}                                          ' Stamp directive.
' {$PBASIC 2.5}                                         ' PBASIC directive.

' -----[ Constantes ]----------------------------------------------------------

' CAMBIAR ESTOS VALORES POR LOS QUE TU DETERMINES E INCLUIRLOS EN LA TABLA 6-1.

LeftAmbient     CON   108
RightAmbient    CON   114
LeftBright      CON   20
RightBright     CON   22

'     Calcular el factor de escala

LeftThreshold   CON   LeftBright + LeftAmbient / 2 * 5 / 8
RightThreshold  CON   RightBright + RightAmbient / 2 * 5 / 8

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

' Declarar variables para almacenar los tiempos RC medidos en las LDR izda.
' y dcha.

timeLeft VAR Word
timeRight VAR Word

' -----[ Inicialización ]------------------------------------------------------

FREQOUT 4, 2000, 3000

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

DO

  GOSUB Test_Photoresistors
  GOSUB Navigate

LOOP

' -----[ Subroutina - Test_Photoresistors ]------------------------------------

Test_Photoresistors:

  HIGH 6                                                ' Medir el tiempo RC izdo.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                               ' Medir el tiempo RC dcho.
  PAUSE 3
  RCTIME 3,1,timeRight

  RETURN
```

```
' -----[ Subroutina - Navigate ]-------------------------------------------

Navigate:

 IF (timeLeft < LeftThreshold) AND (timeRight < RightThreshold) THEN
   PULSOUT 13, 850                                  ' Luz en ambas LDR,
   PULSOUT 12, 650                                  ' avance a toda velocidad.
 ELSEIF (timeLeft < LeftThreshold) THEN             ' Luz en la LDR izda,
   PULSOUT 13, 700                                  ' mover a izda..
   PULSOUT 12, 700
 ELSEIF (timeRight < RightThreshold) THEN           ' Luz en la LDR dcha.,
   PULSOUT 13, 800                                  ' mover a dcha.
   PULSOUT 12, 800
 ELSE
   PULSOUT 13, 750                                  ' No hay luz, esperar.
   PULSOUT 12, 750
 ENDIF

 PAUSE 20                                           ' Pausa entre pulsos.

 RETURN
```

Cómo funciona el programa FlashlightControlledBoeBot.bs2

Estas son las cuatro declaraciones de constantes que tu deberás hacer con los valores que apuntaste en la tabla 7-1:

```
leftAmbient     CON     108
rightAmbient    CON     114
leftBright      CON     20
rightBright     CON     22
```

Ahora que has declarado las constantes, las siguientes dos líneas son para calcular las dos nuevas constantes que comparándolas con los valores de timeLeft y timeRigut te ayudarán a determinar cuando las foto resistencias están detectando luz ambiental o un foco de luz.

```
leftThreshold   CON    ((leftBright+leftAmbient)/2)*5/8
rightThresholdCON      ((rightBright+rightAmbient)/2)*5/8
```

Estas variables se utilizan para guardar las mediciones de RCTIME.

```
timeLeft        VAR    Word
timeRight       VAR    Word
```

Este es el indicador de reset que has utilizado en muchos de los programas anteriores.

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

La sección del programa principal sólo contiene dos llamadas a subrutina. Test_Photoresistors toma las mediciones de RCTIME de los dos circuitos RC y Navigate determina que pulsos se deben enviar a los servos.

```
DO
        GOSUB Test_Photoresistors
        GOSUB Navigate
LOOP
```

Esta es la subrutina que toma las medidas de los circuitos RC. La medición del circuito izquierdo se almacena en timeLeft y la del circuito derecho en timeRight.

```
Test_Photoresistors:
        HIGH 6
        PAUSE 3
        RCTIME 6,1,timeLeft

        HIGH 3
        PAUSE 3
        RCTIME 3,1,timeLeft

        RETURN
```

La subrutina Navigate utiliza un bloque IF…THEN para comparar las variables timeLeft y timeRight con las constantes leftThreshold y rightThreshold respectivamente. Recuerda, cuando la medida de RCTIME es pequeña, eso significa que una luz brillante ha sido detectada. Así, al comparar esta medición (guardada en la variable correspondiente), si es menor que la constante, significa que se ha detectado un foco de luz.

```
Navigate:

  IF(timeLeft<LeftThreshold)AND(timeRight<RightThreshold) THEN
    PULSOUT 13, 850
    PULSOUT 12, 650
  ELSEIF (timeLeft < LeftThreshold) THEN
    PULSOUT 13, 700
    PULSOUT 12, 700
  ELSEIF (timeRight < RightThreshold) THEN
    PULSOUT 13, 800
    PULSOUT 12, 800
  ELSE
    PULSOUT 13, 750
    PULSOUT 12, 750
  ENDIF

  PAUSE 20

  RETURN
```

Ajustando el funcionamiento y cambiando el comportamiento

Puedes ajustar el funcionamiento del programa cambiando el valor de escalado cuando calculas las constantes:

```
leftThreshold  CON    ((leftBright+leftAmbient)/2)*5/8
rightThresholdCON     ((rightBright+rightAmbient)/2)*5/8
```

Si aquí cambias el factor de escalado de 5/8 a 1/2 tu Home Boe-Bot será menos sensible ante las variaciones de luz, por lo que será mucho más difícil que detecte los focos de luz.

- Prueba distintos factores de escalado, como ¼, ½, 1/3, 2/3, y ¾, y anota las diferencias cuando el robot detecta un foco de luz.

Modificando el bloque IF...THEN en el programa de ejemplo puedes cambiar el comportamiento del robot para que en vez de seguir el foco de luz trate de evitarlo.

- Modifica el bloque IF...THEN para que el Home Boe-Bot retroceda cuando detecte el foco de luz con ambas foto resistencias y que gire cuando lo detecte sólo con una de ellas.

7.7. EXPERIENCIA #6: AVANZANDO HACIA LA LUZ

El programa de ejemplo utilizado en esta práctica puede utilizarse para guiar al Home Boe-Bot, situado en una habitación oscura, hacia una puerta entreabierta por la que entra una luz. Permite también un control mucho mayor que cuando generas sombras con tus manos sobre las foto resistencias.

Reajustando las foto resistencias

Esta experiencia funciona mejor con la superficie sensible de las fotorresistencia apuntando hacia fuera y arriba.

- Recoloca las foto resistencias como se muestra en la figura 7-12.

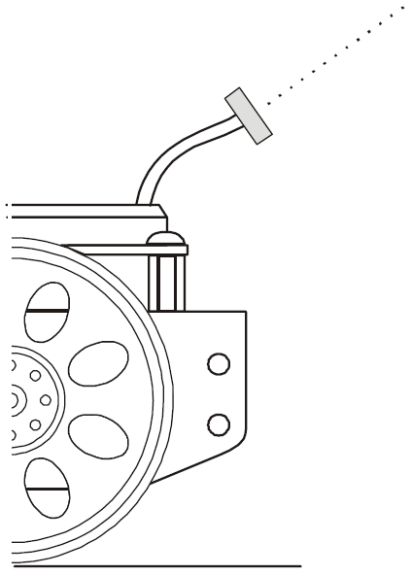


Figura 7-12.- *Orientación de las LDR hacia arriba.*

Programando el Home Boe-Bot para que siga la luz

La clave de mover el Home Boe-Bot para que vaya hacia la luz es hacer que avance cuando las mediciones de las dos foto resistencias sean pequeñas y hacer que gire hacia el lado de la fotorresistencia que haya obtenido la medición más pequeña cuando éstas sean muy diferentes. Esto hará que el robot gire hacia la luz.

Inicialmente parece una tarea fácil de programar; un razonamiento con IF...THEN como el del siguiente ejemplo debería funcionar. El problema está en que el Home Boe-Bot gire continuamente de izquierda a derecha y viceversa porque la variación de los valores de timeLeft y timeRight sea muy grande. Cada vez que el robot gira un poco, las variables timeLeft y timeRight cambian y el Home Boe-Bot intenta corregirlo retrocediendo.

```
IF (timeLeft > timeRight) THEN          ' Turn right
        PULSOUT 13, 850
        PULSOUT 13, 850
ELSEIF (timeRight > timeLeft) THEN  ' Turn left
        PULSOUT 13, 650
        PULSOUT 13, 650
ELSE                                    ' Go forward
        PULSOUT 13, 850
        PULSOUT 12, 650
ENDIF
```

Ahora se propone otro bloque de código que trabaja un poco mejor. En este caso se fija el problema del retroceso y el avance bajo ciertas condiciones. Ahora la variable timeLeft tiene que ser mayor que la timeRight en un margen de al menos 15 unidades antes de que el Home Boe-Bot gire hacia la izquierda. Igualmente ocurre cuando el Boe-Bot gira hacia la derecha. Esto le da al robot la oportunidad de retroceder antes de ejecutar el giro.

```
IF (timeLeft > timeRight + 15) THEN          'Turn right
        PULSOUT 13, 850
        PULSOUT 13, 850
ELSEIF (timeRight > timeLeft + 15) THEN      'Turn left
        PULSOUT 13, 650
        PULSOUT 13, 650
ELSE                                         'Go forward
        PULSOUT 13, 850
        PULSOUT 12, 650
ENDIF
```

El problema de este último bloque de código es que sólo trabaja bajo condiciones de una oscuridad media. Si pones tu Home Boe-Bot en un área más oscura retrocederá y avanzará continuamente. Si lo pones en un área muy brillante, avanzará continuamente, sin girar a izquierda o derecha.

¿Por qué ocurre esto?

Cuando el Home Boe-Bot está en una parte oscura de la habitación, el valor de cada fotorresistencia es muy grande. Para que decida avanzar hacia una zona más iluminada, la diferencia entre ambas mediciones tiene que ser amplia. Cuando el robot está en una zona más iluminada, las mediciones de cada fotorresistencia serán menores, pero para que el Home Boe-Bot gire, la diferencia entre ambas mediciones debe ser mucho menor que cuando estaba en la parte oscura de la habitación. La manera de conseguir que esta diferencia responda a las condiciones de iluminación es utilizando una variable que sea una fracción de la media entre timeLeft y timeRight. Así, dará igual si la luz es brillante o no.

```
average = timeRight + timeLeft / 2
difference = average / 6
```

Ahora la variable difference puede usarse en el bloque IF...THEN, y será un valor alto cuando la luz esté baja, y un valor bajo cuando la luz sea brillante.

```
IF (timeLeft > timeRight + difference) THEN  ' Turn right.
  PULSOUT 13, 850
  PULSOUT 12, 850
ELSEIF (timeRight > timeLeft + difference) THEN ' Turn left.
  PULSOUT 13, 650
  PULSOUT 12, 650
ELSE                                          ' Go forward.
  PULSOUT 13, 850
  PULSOUT 12, 650
ENDIF
```

Programa RoamingTowradTheLight.bs2

Al contrario que RoamingPhotoresistorDividers.bs2, este programa hará que el Home Boe-Bot sea mucho más sensible a las sombras que crees con tus manos sobre las foto resistencias, sea el nivel de luz ambiental alto o bajo. Este programa no necesita que cambies la posición de las foto resistencias en función de las condiciones de iluminación. En cambio, el almacenamiento de las condiciones de iluminación y los ajustes de sensibilidad se realizan por software utilizando las variables average y difference.

Este programa utiliza la media total de timeLeft y timeRight y la utiliza para calcular difference, valor que empleará para determinar los pulsos de giro.

- Teclea, salva y ejecuta el programa RoamingTowardTheLight.bs2.
- Pon en marcha tu Home Boe-Bot en diferentes áreas, y asegúrate de que sigue las sombras que crees sobre las foto resistencias dándole igual las condiciones de iluminación.
- Prueba a dejar tu Home Boe-Bot en una habitación oscura y con la puerta entreabierta por la que entra algo de luz para ver si es capaz de salir.

```
' -----[ Título ]-------------------------------------------------------------
' El Robot HomeBoe-Bot - RoamingTowardTheLight.bs2

' {$Stamp bs2}              ' Stamp directive.
' {$PBASIC 2.5}             ' PBASIC directive.

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

' Declara variables para almacenar las medidas de tiempo RC de las LDR
' izda. y dcha.

timeLeft      VAR Word
timeRight     VAR Word
average       VAR Word
difference    VAR Word

' -----[ Inicialización ]-----------------------------------------------------

FREQOUT 4, 2000, 3000

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

DO

  GOSUB Test_Photoresistors
  GOSUB Average_And_Difference
  GOSUB Navigate
```

```
LOOP

' -----[ Subroutina - Test_Photoresistors ]----------------------------------

Test_Photoresistors:

  HIGH 6                                        ' Medida del tiempo RC izdo.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                         ' Medida del tiempo RC dcho.
  PAUSE 3
  RCTIME 3,1,timeRight

  RETURN

' -----[ Subroutina - Average_And_Difference ]-------------------------------

Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / 6

  RETURN

' -----[ Subroutina - Navigate ]-----------------------------------------

Navigate:


  IF (timeLeft > timeRight + difference) THEN
    PULSOUT 13, 850
    PULSOUT 12, 850

  ELSEIF (timeRight > timeLeft + difference) THEN
    PULSOUT 13, 650
    PULSOUT 12, 650

  ELSE
    PULSOUT 13, 850
    PULSOUT 12, 650
  ENDIF

PAUSE 10

RETURN
```

Ajustando la sensibilidad según la luz

La variable difference es la average media (average) dividida entre 6. Puedes dividir average por un valor más pequeño si quieres que el Home Boe-Bot sea menos sensible a las diferencias de iluminación, o mayor si lo quieres hacer más sensible.

- Prueba a dividir la variable average por 3, 4, 5, 7 y 9.

- Ejecuta el programa y comprueba si el Home Boe-Bot es capaz de salir de la habitación oscura con los distintos valores.
- Decide cuál es el denominador óptimo.

```
Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / 6

  RETURN
```

Puedes meter el denominador en una constante:

```
Denominator          CON          6
```

Entonces, en la subrutina Average_And_Difference deberás cambiar el valor 6 por la constante Denominator:

```
Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / Denominator

  RETURN
```

También puedes eliminar una variable. Fíjate que la única vez que utilizas la variable average es para almacenar temporalmente el valor de la media que después dividirás por Denominator para almacenarlo en la variable difference. La variable difference vuelve a ser utilizada después, no así average:

```
Average_And_Difference:

  difference = timeRight + timeLeft / 2
  difference = difference / Denominator

  RETURN
```

De todas formas, por legibilidad lo dejaremos como estaba antes:

```
Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / Denominator

  RETURN
```

Haz ahora el siguiente cambio en las declaraciones de las variables:

```
' Unchanged code                    ' Changed to save Word of RAM

average       VAR     Word          average       VAR     Word
difference    VAR     Word          difference    VAR     average
```

Ahora, tanto average como difference referencian a la misma palabra en la RAM. Comprueba que todo funciona correctamente después de hacer los cambios.

7.8 PRUEBA DE AUTOEVALUACION

Cuestiones

1. ¿Qué significan las siglas LDR?
2. ¿Cómo se comporta el valor de la resistencia interna de las foto resistencias ante una luz muy brillante o muy débil? ¿Y si es una luz intermedia?
3. ¿Tiene un pin de E/S algún efecto sobre el circuito cuando está configurado como entrada? ¿Qué tiene que ocurrir para que un registro asociado a un pin de E/S configurado como entrada almacene un 1 o un 0?
4. ¿Qué quiere decir voltaje límite o umbral? ¿Cuál es el voltaje límite para un pin de E/S de la Home Work?
5. ¿Qué es un divisor de voltaje?
6. Según la figura 7-2, ¿qué hace que Vo suba por encima o baje por debajo del voltaje límite de un pin de E/S de la Home Work? ¿Cuál es el circuito que hace que se produzca esta variación de Vo?
7. ¿Qué ajuste puedes hacer en el circuito de la figura 7-2 para que el Home Boe-Bot pueda funcionar en un área más iluminada? ¿Y para que lo haga en un área menos iluminada?
8. ¿Cuáles son los mínimos cambios que debes hacer en RoamingWithWhiskers.bs2 para que funcione con las foto resistencias?
9. ¿Cuáles son las diferencias entre ShadowGuidedBoeBot.bs2 y RoamingWithPhotoresistorDividers.bs2? ¿En que varía el funcionamiento del Home Boe-Bot?
10. ¿Cuál es la diferencia entre un faradio y un μfaradio?
11. Según la figura 7-10, ¿qué efectos tiene HIGH 6 en el voltaje que atraviesa el condensador? ¿Qué hace PAUSE 3? ¿Y RCTIME?
12. ¿Qué es una declaración de constantes? ¿Qué es lo que hace? ¿Cómo lo puedes usar en un programa?
13. ¿Cómo se evalúan las expresiones matemáticas en PBASIC?
14. ¿Cuál es la diferencia entre utilizar un número y una constante en un bloque IF...THEN? ¿Cuál es la diferencia entre usar una medición RCTIME en un bloque IF...THEN y usar un valor obtenido de un pin de E/S?
15. ¿Qué dos ejemplos en este capítulo utilizan el PBASIC para calcular una media? ¿Por qué son distintos? ¿Por qué son iguales?
16. Puedes utilizar estos modificadores: Bit, Nib, Byte y Word cuando declaras una variable pero, ¿puedes declarar un nombre de variable para referirte a otra variable que ya ha sido declarada? Si sí se puede, ¿cómo se hace? Si no, ¿por qué no?

Ejercicios

1. Para la figura 7-2, calcula Vo si R = 10 kΩ. Repítelo para R = 30 kΩ.
2. Para la figura 7-2, calcula Vo si R = 20 kΩ. Repite este cálculo, pero sustituye la resistencia de 2 kΩ de la figura por los siguientes valores: 220 Ω, 470 Ω, 1 kΩ, 4'7 kΩ, 10 kΩ.
3. Si en le figura 7-2 el valor de Vo es 1'4 V, ¿cuánto vale R?. Repítelo para Vo = 1 V y Vo = 3 V.
4. Escribe un bloque de sentencias IF...THEN que haga que el Home Boe-Bot se mueva hacia delante si detecta una luz brillante y hacia atrás si la luz es débil. También debes conseguir que el robot se aparte de las sombras.
5. Escribe una declaración de constantes que pueda utilizarse con el comando PAUSE para el tiempo que pasa entre los pulsos de los servos. Reescribe el comando PAUSE para que utilice esta constante.
6. Repite el anterior ejercicio con los valores 13 y 12 para referirte a los pines de E/S. Escribe algunos ejemplos utilizando PULSOUT con tus constantes de pin.
7. Supón que tienes tres valores almacenados en tres variables: firstValue, secondValue y thirdValue. Escribe la sentencia que calcule la media de estas tres variables y deje el resultado en una variable llamada myAverage. Escribe una sentencia que calcule 7/8 de la media anterior y deje el resultado en una nueva variable llamada myScaleAverage. Escribe las declaraciones de las variables necesarias para que esto funcione en un programa real, primero por separado, y luego asociando myAverage con myScaleAverage.

Proyectos

1. Con las foto resistencias de tu Home Boe-Bot mirando hacia delante y abajo, escribe un programa que haga que el Boe-Bot diferencie entre blanco y negro. Busca una superficie blanca esparce trozos de papel negro sobre ella. Escribe un programa para que el Home Boe-Bot evite estos papeles. Pistas: Asegúrate de probar y entender cuando el robot detecta el color blanco y cuando el negro. Utiliza los programas de las tres últimas actividades de este capítulo. Asegúrate de que los obstáculos sigan un curso uniforme y que no hay luces brillantes en la habitación donde haces las pruebas.
2. Si has hecho bien el proyecto 1, prueba a meter al Home Boe-Bot entre tiras de papel negro.

Tema 8

Navegación guiada por Infrarrojos IR

8.1. PRINCIPIOS Y APLICACIONES DE LOS RAYOS INFRARROJOS

Las ondas electromagnéticas emitidas por los cuerpos de color rojo tienen una longitud de onda de 780 nm (nanómetros) y son visibles al ojo humano. Por encima de dicha longitud de onda las ondas se denominan "infrarrojas" (IR) y ya no son visibles. Existen ciertos modelos de diodos LED que emiten ondas infrarrojas y se llaman LED IR. También existen detectores (receptores) de IR. En nuestras aplicaciones usaremos pares de emisores/receptores que funcionan con una longitud de onda de 980 nm.

Para seguir un camino hay procedimientos sofisticados que utilizan Visión Artificial o Radar. Una forma más sencilla y barata es mediante los rayos IR.

El sistema de infrarrojos (IR a partir de ahora) que vamos a montar en el Boe-Bot se parece bastante a los focos de un coche. Cuando las luces del coche enfocan los obstáculos, el conductor puede detectarlos y desviar el vehículo para no chocar. El Home Boe-Bot usa LED IR como el que se muestra en la figura 8-1. Estos LED emiten luz infrarroja que se refleja en los obstáculos y vuelve hacia el robot. Los ojos del Home Boe-Bot son los detectores IR que cuando detectan la luz que refleja el obstáculo envían señales a la Home Work. El cerebro del robot, toma decisiones y maneja los servomotores en función de los datos recibidos desde los detectores.

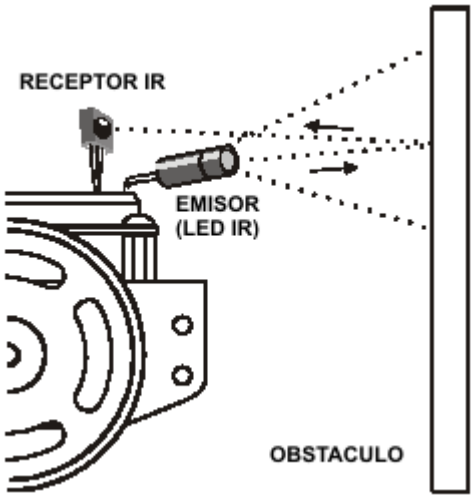


Figura 8-1.- *El LED emite rayos infrarrojos que al reflejarse en el obstáculo los recibe el sensor receptor.*

Los detectores IR que usaremos llevan incorporados filtros ópticos que únicamente permiten el paso de luz infrarroja de 980 nm. También incorporan un filtro electrónico que impide el paso de señales cuya frecuencia no sea 38.5 kHz, es decir, los detectores seleccionan una fuente de luz que parpadea 38.500 veces por segundo, así se consigue que otras fuentes de luz no interfieran en la lectura. La luz del sol tiene una frecuencia de 0 Hz porque no parpadea ninguna vez por segundo y la luz artificial tiende a parpadear con una frecuencia comprendida entre los 100 y los 120 Hz. Como estas frecuencias se encuentran muy alejadas de los 38.5 kHz que filtran los detectores IR, tanto la luz natural como la artificial utilizada para iluminación serán ignoradas.

8.2. EXPERIENCIA #1: MONTAJE Y PUESTA A PUNTO DE LOS SENSORES DE IR

Vamos a montar y probar los pares emisor/detector de IR. Se necesitan los materiales mostrados en la figura 8-2, que son los siguientes:

(2) Detectores de infrarrojos
(2) LED infrarrojos
(2) Carcasas para los LED infrarrojos
(2) Resistencias 220 Ω
(2) Resistencias 1 kΩ

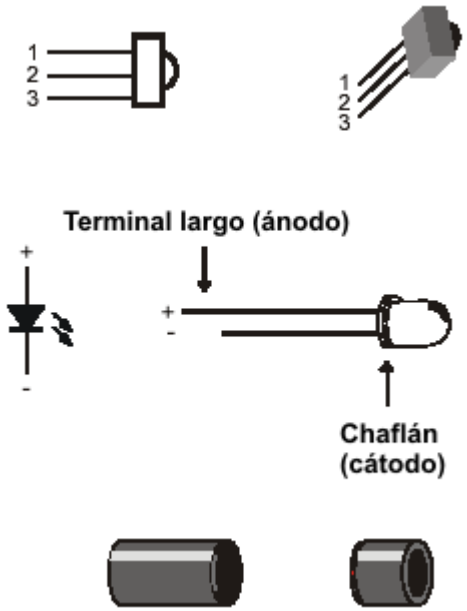


Figura 8-2.- *Detectores IR (arriba), en el centro LED emisor y abajo carcasa para este último.*

Sigue los siguientes pasos para el montaje de los emisores IR, de acuerdo con la figura 8-3.

- Introduce los LED IR en las carcasas como se muestra en la figura 8-3
- Asegúrate de que los LED encajan en la parte grande de las carcasas
- Encaja la parte más pequeña de cada carcasa sobre cada LED que se ha montado en la parte grande.

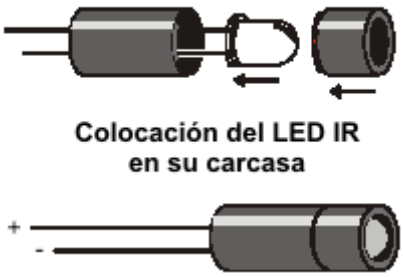


Figura 8-3.- *Forma de introducir el LED IR en la carcasa*

Cada pareja de emisor/receptor IR será montada en una esquina de la protoboard de la Home Work. La figura 8-4 muestra el circuito eléctrico y la figura 8-5, la apariencia final del montaje. Desconecta las pilas de la placa y los servomotores durante el montaje de los elementos. Monta el circuito de la figura 8-5.

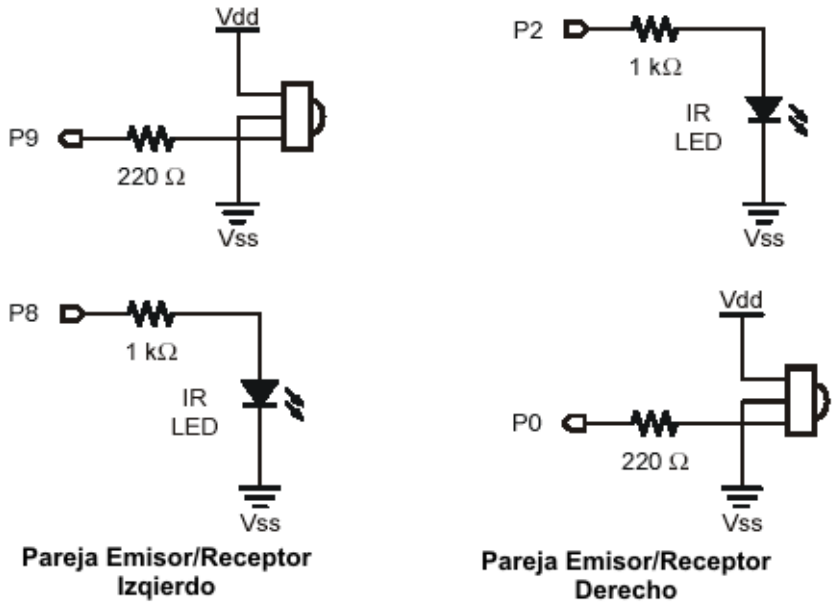


Figura 8-4.- *Esquemas eléctricos del conexionado de las parejas emisor/receptor de IR para el lado izquierdo y el derecho del robot.*

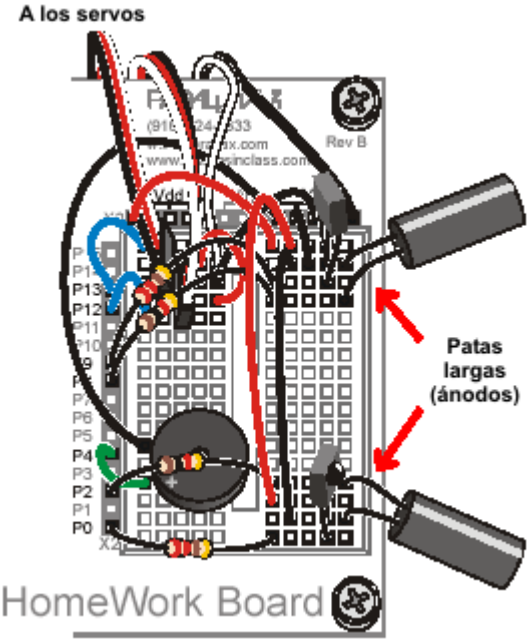


Figura 8-5.- *Apariencia del montaje de las parejas emisor/receptor de IR.*

El comando FREQOUT

El comando FREQOUT fue diseñado principalmente para sintetizar y generar distintos tonos de sonido. El rango de actuación de FREQOUT se encuentra entre 1 y 32.768 Hz. Un interesante fenómeno del audio sintetizado es que contiene unas señales llamadas armónicos. Un armónico es una frecuencia múltiplo superior mezclada con la frecuencia del tono real que se está sintetizando. El umbral de percepción humana se encuentra comprendido, aproximadamente, entre los 20 Hz y los 20 kHz. Los armónicos generados por el comando FREQOUT son de frecuencias superiores a 32.769 Hz. Se puede controlar la frecuencia de los armónicos a través del parámetro Freq1 con valores superiores a 32.768. En esta práctica se va a usar la instrucción FREQOUT 8, 1, 38.500 para enviar un armónico de 38.5 kHz a la patita P8 durante 1 ms. El LED IR conectado a P8 emitirá una

señal infrarroja a dicha frecuencia y si encuentra un objeto en el camino a una distancia adecuada la refleja con cierto nivel hasta el detector IR (que tiene un filtro a 38.500 Hz). Éste captará la onda reflejada de IR y producirá una señal de reconocimiento a la Home Work. Si el objeto se halla a mucha distancia el nivel de la onda reflejada no será suficiente para ser captado por el receptor de IR.

La manera de funcionar cada par emisor/receptor IR es enviar una señal infrarroja de 38.5 kHz por el emisor durante 1 ms e inmediatamente almacenar la salida del detector/receptor IR correspondiente en una variable. He aquí un ejemplo que envía una señal de 38.5 kHz al LED IR conectado a P8 durante 1 ms y después almacena la salida del detector IR que está conectado a P9 en una variable de tipo bit llamada irDetectLeft.

```
FREQOUT 8, 1, 38500
lrDetectLeft = IN9
```

Cuando el detector/receptor IR no percibe ninguna señal su salida permanece en estado alto. Cuando el detector IR capta el armónico de 38.500 Hz reflejado por un objeto, su salida pasa a nivel bajo. La salida del detector sólo está a nivel bajo durante un milisegundo después de que el comando FREQOUT haya enviado el armónico, así que es vital almacenar el contenido de la salida del detector/receptor justo a continuación de haber enviado el armónico.

Programa TestLeftIrPair.bs2

- Vuelve a conectar la pila a la placa
- Teclea, guarda y ejecuta el programa TestLeftIrPair.bs2

```
' El robot Home Boe-Bot - TestLeftIrPair.bs2
' Comprobar los circuitos IR de detección de objetos, el LED IR está conectado a P8
' el detector/receptor  IR está conectado a P9

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

irDetectLeft VAR Bit

DO
  FREQOUT 8, 1, 38500
  irDetectLeft = IN9

  DEBUG HOME, "Detector IR izdo. = ", BIN1 irDetectLeft
  PAUSE 100

LOOP
```

Deja conectado al PC el Home Boe-Bot por el cable serie porque seguirás usando el terminal de depuración para probar la pareja de IR. Sitúa un objeto delante de la pareja IR a un par de centímetros como se ve en la figura 8-1. Comprueba que cuando sitúas un objeto ante la pareja IR, el terminal de depuración muestra un 0 y cuando lo retiras, muestra un 1

Tu turno

- Guarda el programa TestLeftIrPair.bs2 con el nombre TestRightIrPair.bs2
- Cambia la sentencia DEBUG, el título y los comentarios para hacer referencia al par IR derecho.
- Cambia el nombre de la variable irDetectLeft por irDetectRight. Deberás hacerlo en tres líneas del programa.
- Cambia el parámetro Pin del comando FREQOUT de 8 a 2
- Modifica el registro que monitoriza la variable irDetectRight de IN9 a IN10
- Repite los pasos de prueba para el par IR de la derecha con el LED IR conectado a P2 y el detector conectado a P0

8.3 EXPERIENCIA #2: DETECTANDO OBJETOS E INTERFERENCIAS

En esta práctica montaremos unos LED indicadores que señalizarán cuando se detecta un objeto sin necesidad del terminal de depuración. Escribiremos también un programa para "escuchar" las interferencias provenientes de las lámparas fluorescentes. El dispositivo que controla el voltaje dentro de una lámpara fluorescente se llama reactancia. Algunas reactancias funcionan a la misma frecuencia que nuestro detector IR, 38.5 kHz, haciendo que la lámpara emita señales a dicha frecuencia. Cuando se incluye la detección de objetos en la navegación del Home Boe-Bot, estas interferencias pueden causar comportamientos extraños.

Para esta experiencia necesitaremos:

(2) LED rojos
(2) Resistencias 220 Ω

- Desconecta las pilas de la placa y los servomotores
- Monta el circuito de la figura 8-7 usando el esquema de la figura 8-6 como referencia

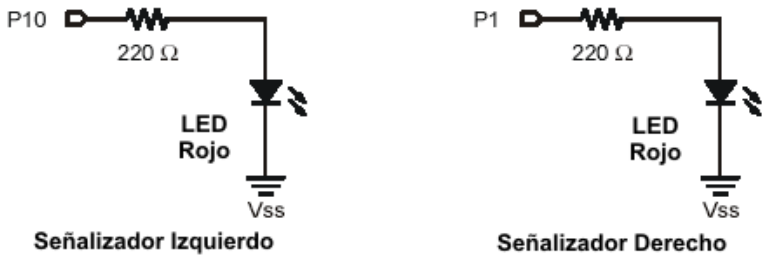


Figura 8-6.- *Esquema de conexionado de los dos LED rojos de monitorización.*

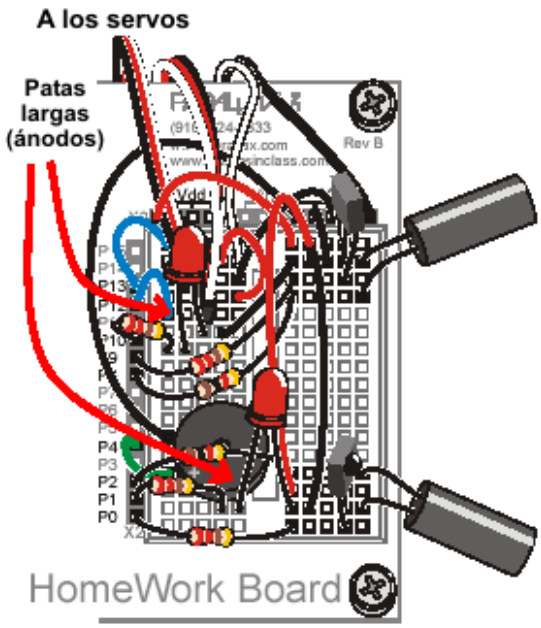


Figura 8-7.- *Apariencia del montaje una vez montados los dos LED.*

Hay bastantes componentes involucrados en este circuito por lo que la posibilidad de cometer un error en el cableado aumenta. Por eso es importante tener un programa de prueba para asegurarnos de que los detectores IR están "escuchando" a los emisores y lo hacen correctamente. Se puede utilizar este programa para comprobar que todo funciona correctamente antes de desenchufar el Home Boe-Bot del cable serie.

- Vuelve a conectar la pila a la placa
- Teclea, guarda y ejecuta el programa TestIrPairsAndIndicators.bs2
- Comprueba que el zumbador emite un tono audible mientras en el terminal de depuración se muestra el mensaje "Testing piezospeaker..."
- Usa el terminal de depuración para comprobar que la Home Work sigue recibiendo un 0 de cada detector IR cuando se coloca un objeto frente a ellos.
- Verifica que cada LED rojo al lado de los detectores emite luz cuando se detecta un objeto. Si alguno de los LED no funciona, comprueba de nuevo el cableado.

```
' El robot Home Boe-Bot - TestIrPairsAndIndicators.bs2
' Comprobar los circuitos IR de detección de objetos.

' {$STAMP BS2}                    ' Stamp directive.
' {$PBASIC 2.5}                   ' PBASIC directive.

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

irDetectLeft  VAR Bit
irDetectRight VAR Bit

' -----[ Inicialización ]---------------------------------------------------

DEBUG "Comprobando el piezo..."
FREQOUT 4, 2000, 3000

DEBUG CLS,
      "DETECTORES IR", CR,
      "Izdo.  Dcho.", CR,
      "-----  -----"

' -----[ Main Routine ]-----------------------------------------------------

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
```

Tu turno

Prueba a colocar objetos de diferentes colores en el camino del Boe-Bot. ¿Qué colores detecta y cuáles no?

Intentando captar interferencias IR

Si el Boe-Bot detecta algún obstáculo cuando realmente no hay ninguno, obviamente está haciendo una lectura incorrecta de su entorno. Normalmente esto sucede cuando la luz ambiente interfiere con los detectores IR. Lo peor que te puede pasar si estás haciendo una demostración de las habilidades de tu Home Boe-Bot es que no se comporte como has previsto, por eso es conveniente hacer esta sencilla comprobación en el escenario de la demostración.

La prueba es muy sencilla: simplemente dejaremos al Home Boe-Bot tomando lecturas a través de los detectores de infrarrojos sin que los LED estén emitiendo. De este modo, si se detecta algo, el zumbador comenzará a sonar y sabremos que se trata de la luz ambiental producida por lámparas fluorescentes y no tiene nada que ver con nuestro sistema de infrarrojos.

Programa InterferenceSniffer.bs2

Teclea, salva y ejecuta el programa InterferenceSniffer.bs2. Asegúrate de que el sistema está funcionando. Para ello utiliza un segundo Home Boe-Bot que esté emitiendo por los IR LED y comprueba que el zumbador suena cuando ambos están enfrentados. Si no dispones de un segundo Home Boe-Bot, puedes utilizar un mando a distancia de cualquier aparato de casa, por ejemplo el del televisor. Apunta hacia el Home Boe-Bot y presiona cualquier botón. Durante la pulsación el zumbador debería sonar.

```
' Robotics with the Boe-Bot – IrInterferenceSniffer.bs2
' Comprobar luces fluorescentes, infrarojos remotos y otras fuentes
' de interferencias IR a 38.5Khz

' {$STAMP BS2}                    ' Stamp directive.
' {$PBASIC 2.5}                   ' PBASIC directive.

counter VAR Nib

DEBUG "Interferencias IR no detectadas...", CR

DO
  IF (IN0 = 0) OR (IN9 = 0) THEN
    DEBUG "Interferencias IR detectadas!!!", CR
    FOR counter = 1 TO 5
      HIGH 1
      HIGH 10
      FREQOUT 4, 50, 4000
      LOW 1
      LOW 10
      PAUSE 20
    NEXT
  ENDIF

LOOP
```

8.4. EXPERIENCIA #3: AJUSTE DEL RANGO DE DETECCIÓN DE LOS INFRARROJOS

Cuanto más iluminan las luces de un coche, más lejos se puede ver en la oscuridad. Lo mismo le sucede al Boe-Bot: cuanta más intensidad de IR generen los LED, más aumentará el rango o distancia de detección de

obstáculos. Cuanta menos resistencia ofrezcamos a la corriente, más intensidad de rayos IR producirán los LED. Se trata de modificar los valores de las resistencias de polarización de los LED para variar la intensidad IR generada por ellos.

Se precisan los siguientes materiales:

(2) Resistencias 470Ω (Amarillo morado marrón)
(2) Resistencias 220Ω (Rojo rojo marrón)
(1) Resistencia 1 kΩ (Marrón negro rojo)

Programa P1LedHigh.bs2

- Teclea, guarda y ejecuta el programa P1LedHigh.bs2
- Cuando el programa se esté ejecutando, observa la intensidad luminosa del LED conectado a P1.

```
' El robot Home Boe-Bot - P1LedHigh.bs2
' Poner a nivel alto P1 para comprobar el brillo con diferentes valores de
' la resistencia: 220, 470 y 1K

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

HIGH 1

STOP
```

Se utiliza el comando STOP en lugar de END para finalizar el programa porque END pondría las patitas de salida de la Home Work a nivel bajo y se apagaría el LED.

Observa cómo brilla el LED conectado a P1 con la resistencia de 220Ω

- Sustituye la resistencia de 220 Ω del circuito del LED conectado a P1 por una de 470 Ω .
- Observa cómo brilla el LED.
- Reemplázala por una de 1 kΩ.
- Antes de terminar la actividad, vuelve a dejar la resistencia original. La de 220 Ω.
- Explica con tus palabras la relación entre el valor de la resistencia y el brillo generado por el LED.

Cuanto menor sea la resistencia de polarización, mayor será la intensidad con la que emitirá IR el LED. Una hipótesis razonable sería que cuanto mayor sea dicha intensidad, mayor será la distancia a la que se puede detectar.

- Abre y ejecuta el programa TestIrPairsAndIndicators.bs2
- Comprueba que ambos detectores funcionan correctamente
- Mide con una regla la máxima distancia a la que se puede detectar una hoja en blanco puesta frente al Home Boe-Bot
- Sustituye las resistencias de 1 kΩ que conectan P2 y P8 a los ánodos de los LED IR por resistencias de 470 Ω
- Determina la máxima distancia a la que se detecta la hoja esta vez
- Repite el proceso con resistencias de 220 Ω
- Anota los datos en una tabla que relacione el valor de la resistencia con la distancia de detección.
- Vuelve a colocar las resistencias originales
- Antes de terminar la práctica, vuelve a probar el Home Boe-Bot con el programa TestIrPairsAndIndicators.bs2 para ver si ha quedado funcionando perfectamente

8.5. EXPERIENCIA #4: DETECTANDO Y ESQUIVANDO OBJETOS

La salida de los detectores IR tiene nivel alto cuando no se detecta ningún objeto y bajo cuando algo se interpone en su camino. En esta experiencia, modificaremos el programa RoaminWithWhiskers.bs2 para que funcione con los detectores IR, en lugar de los bigotes.

El nuevo programa va a ser muy parecido al anterior, cambiando la descripción y añadiendo dos nuevas variables para almacenar los estados de los detectores IR.

```
IrDetectLeft    VAR    Bit
IrDetectRight   VAR    Bit
```

También se ha añadido una rutina para leer cada pareja de IR

```
FREQOUT 8,1,38500
IrDetectLeft = IN9
```

Las sentencias IF...THEN han sido modificadas para que ahora se tomen en cuenta las variables donde se almacenan las salidas de los detectores IR.

```
IF (irDetectLeft = 0) AND (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 RoamingWithIr.bs2

- Abre el programa RoaminWithWhiskers.bs2
- Modifícalo para que coincida con el programa siguiente
- Coloca las pilas de la placa y los servomotores
- Guarda y ejecuta el programa
- Comprueba que el Boe-Bot se comporta como lo hacía con el programa RoaminWithWhiskers.bs2, pero ahora sin contacto con el objeto.

```
' -----[ Título ]-------------------------------------------------------
' El robot Home Boe-Bot - RoamingWithIr.bs2
' Adaptando el programa RoamingWithWhiskers.bs2 para
' su empleo con los IR.

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

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

irDetectLeft      VAR Bit
irDetectRight     VAR Bit
pulseCount        VAR Byte
```

```
' -----[ Inicialización ]-------------------------------------------------------

FREQOUT 4, 2000, 3000                          ' Señal de Inicio/Reset

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

DO

  FREQOUT 8, 1, 38500                          ' Almacenar los valores de los IR
  irDetectLeft = IN9                           ' en las variables tipo bit.

  FREQOUT 2, 1, 38500
  irDetectRight = IN0

  IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    GOSUB Back_Up                              ' Ambos IR detectan obstáculo,
    GOSUB Turn_Left                            ' retroceso y giro a la izda.
    GOSUB Turn_Left
  ELSEIF (irDetectLeft = 0) THEN               ' Se activa el detector IR izdo.,
    GOSUB Back_Up                              ' retroceso y giro a la dcha.
    GOSUB Turn_Right
  ELSEIF (irDetectRight = 0) THEN              ' Se activa el detector IR dcho.,
    GOSUB Back_Up                              ' retroceso y giro a la izda.
    GOSUB Turn_Left
  ELSE                                         ' Ningún IR detecta obstáculo
    GOSUB Forward_Pulse                        ' Aplicar un pulso de avance
  ENDIF                                        ' y volver a comprobar

LOOP

' -----[ Subrutinas ]------------------------------------------------------------

Forward_Pulse:                                 ' Envía un pulso de avance.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                                     ' Giro a la izda 90º.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20                     ' Giro a la dcha. 90º.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN

Back_Up:                                       ' Retroceso.
```

```
FOR pulseCount = 0 TO 40
  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20
NEXT
RETURN
```

Modifica el programa RoamingWithIr.bs2 para que los pares emisor/receptor de IR se comprueben en una subrutina.

8.6. EXPERIENCIA #5: NAVEGACIÓN IR DE ALTO NIVEL

El estilo de maniobras preprogramadas usadas en la experiencia anterior es muy lento cuando se utilizan LED y detectores IR. Se puede mejorar la navegación del Home Boe-Bot comprobando la presencia de obstáculos antes de enviar pulsos a los servomotores. El programa puede utilizar las entradas de los sensores para elegir la mejor maniobra en cada momento, así el Home Boe-Bot nunca girará más de lo que debe y también puede esquivar el obstáculo de forma casi exacta.

Usando las parejas IR se detecta el obstáculo antes de tocarlo y el Home Boe-Bot dispone de cierto espacio para moverse alrededor de él. El Home Boe-Bot puede aplicar un pulso para girar en otra dirección, volver a comprobar si el obstáculo sigue ahí, aplicar otro pulso, etc, hasta que su camino esté libre. Entonces puede seguir enviando pulsos para continuar hacia delante. Después de experimentar con el siguiente programa, te darás cuenta que este procedimiento es mucho mejor que los anteriores de evitar colisiones.

Programa FastIrRoaming.bs2

Teclea, guarda y ejecuta el programa FastIrRoaming.bs2

```
' El robot HomeBoe-Bot - FastIrRoaming.bs2
' Navegación de alto nivel mediante el empleo de IR

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

irDetectLeft  VAR Bit                          ' Declaración de variables
irDetectRight VAR Bit
pulseLeft     VAR Word
pulseRight    VAR Word

FREQOUT 4, 2000, 3000                          ' Señal de Inicio/reset.

DO                                             ' Rutina principal
  FREQOUT 8, 1, 38500                          ' Comprobar los detectores IR
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                               ' Decidir cómo navegar.

  IF (irDetectLeft = 0) AND (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                    ' Generar el pulso de movimiento.
PULSOUT 12,pulseRight
PAUSE 15

LOOP
```

Este programa aplica los pulsos de una forma totalmente distinta. Aparte de que utiliza dos bits para guardar las salidas de los detectores IR, utiliza dos variables tipo Word para definir la duración de los pulsos utilizada por el comando PULSOUT.

```
irDetectLeft    VAR Bit
irDetectRight   VAR Bit
pulseLeft       VAR Word
pulseRight      VAR Word
```

Dentro del DO...LOOP, los comandos FREQOUT se usan para enviar una señal de 38.5 kHz a cada LED IR. Inmediatamente después, una variable almacena el estado de salida de cada detector IR. Es necesario porque si se espera algo más, el detector volverá al estado de no detectar nada (1), independientemente de si hay o no un objeto delante del Home Boe-Bot.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9
FREQOUT 2, 1, 38500
irDetectRight = IN0
```

En las sentencias IF..THEN, en vez de enviar pulsos o llamar a rutinas de navegación, este programa asigna valores a ciertas variables que luego serán utilizadas en los comandos PULSOUT como parámetro Duración.

```
IF (irDetectLeft = 0) AND (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 que el DO...LOOP se repita, lo último que se hace es enviar los pulsos a los servomotores. Observa que el comando PAUSE es 15, no 20 como antes, la razón es que se tarda aproximadamente 5 ms en comprobar los LED IR.

```
PULSOUT 13,pulseLeft
PULSOUT 12,pulseRight
PAUSE 15
```

Tu turno

- Guarda el programa FastIrRoaming.bs2 con el nombre de FastIrRoamingYourTurn.bs2
- Modifica el programa para que el zumbador emita diferentes tonos en cada pulso. Elige tonos para cada una de las diferentes combinaciones de pulsos (adelante, atrás, izquierda, derecha). Reduce el comando PAUSE a 7 ms y utiliza otros 7 ms para el comando FREQOUT.
- Utiliza los LED para mostrar que el Home Boe-Bot ha detectado un objeto.
- Intenta modificar los valores de pulseLeft y pulseRight para que el Home Boe-Bot haga todo a la mitad de velocidad.

8.7. EXPERIENCIA #6: EVITANDO LAS CAIDAS DE LA MESA

Hasta ahora el Home Boe-Bot ha sido programado para realizar maniobras evasivas cuando se detectaba un objeto, pero también hay aplicaciones en las que el Home Boe-Bot tiene que adoptar esas medidas cuando el objeto NO es detectado. Por ejemplo, si el Home Boe-Bot está moviéndose por la superficie de una mesa, sus detectores IR podrían estar mirando hacia abajo, hacia la superficie de la mesa, tal y como se muestra en la figura 8-8. El programa debería hacerle continuar moviéndose hacia delante mientras los detectores "vean" la mesa. Pero si deja de "verla" significa que se termina la mesa y si continua avanzando se caería, cosa que hay que evitar.

- Desconecta las pilas de la placa y de los servomotores
- Apunta las parejas detector/emisor IR hacia abajo y hacia fuera del Boe-Bot, como se indica en la figura 8-8

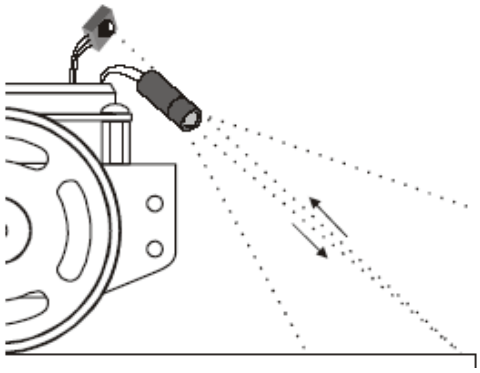


Figura 8-8.- *Los detectores IR se orientan hacia ja superficie de la mesa. Cuando dejen de "verla" hay que detener el movimiento para que no se caiga el robot.*

En esta práctica se necesita:

(1) rollo de cinta adhesiva negra de 19 mm de ancho.
(1) hoja blanca A2 o superior

Simulación del borde de la mesa con cinta adhesiva

Construye una superficie con la hoja blanca y la cinta adhesiva negra similar a la de la figura 8-9. Utiliza al menos tres bandas de cinta adhesiva de borde a borde sin dejar papel visible entre cada banda.

- Asegúrate de estar usando resistencias de 1 kΩ (marrón, negro, rojo) para conectar P2 a su LED y P8 al suyo.
- Conecta la pila de la placa.
- Utiliza el programa TestIrPairsAndIndicators.bs2 para estar seguro de que el Boe-Bot detecta la superficie blanca de la hoja pero no detecta las bandas negras.
- Ejecuta el programa IrInterferenceSniffer.bs2 para comprobar que las luces de iluminación no interfieren con los detectores IR del Boe-Bot.

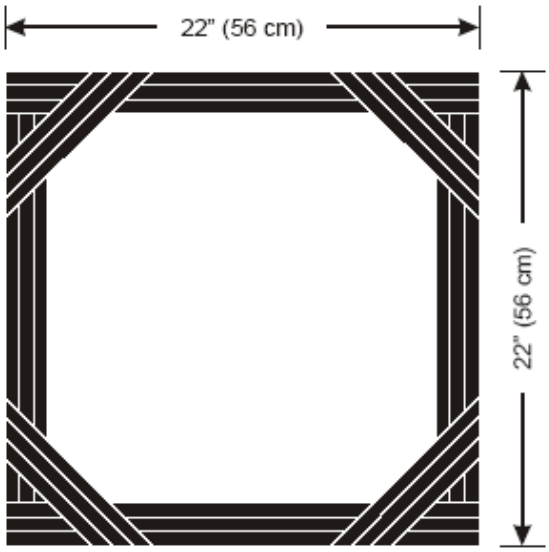


Figura 8-9.- *Simulación de una mesa con una hoja blanca y cinta adhesiva en los bordes.*

Detección del borde de la mesa

Se trata de programar el Home Boe-Bot para que se mueva por la mesa sin caer por sus bordes. Hay que ajustar las sentencias IF...THEN del programa FastIrNavigation.bs2. Los servomotores tienen que ser controlados para que el Home Boe-Bot avance cuando irDetectLeft e irDetectRight sean 0, indicando que un objeto (la superficie de la mesa) es detectado. También habrá que apartarsedel detector que no detecta objeto. Por ejemplo, si irDetectLeft es 1, el Home Boe-Bot tendrá que girar hacia la derecha.

Otra función de un buen programa que evite caídas es que se pueda ajustar la distancia de reacción Home. Puede que quieras que el Home Boe-Bot sólo envíe un pulso hacia delante entre cada comprobación de los sensores pero en cuanto detecta un "vacío" quieras que envíe varios pulsos para girar antes de volver a comprobar los sensores.

Sólo por estar enviando múltiples pulsos en la maniobra evasiva no significa que hayamos vuelto al comportamiento anterior del Home Boe-Bot cuando chocaba con los obstáculos y luego cambiaba de dirección. Se puede utilizar una variable para definir el número de pulsos a enviar en la maniobra. El comando PULSOUT se puede colocar dentro de un FOR...NEXT que se ejecute desde 1 hasta el valor de dicha variable. Para un pulso hacia delante, la variable valdría 1, para 10 pulsos, 10, etc.

Programa AvoidTableEdge.bs2

- Abre el programa FastIrNavigation.bs2 y guárdalo como AvoidTableEdge.bs2
- Modifica el programa para que coincida con el programa propuesto.
- Vuelve a conectar las pilas de la placa y los servomotores.
- Prueba el programa en la superficie delimitada por bandas de cinta adhesiva

```
' El robot Home Boe-Bot - AvoidTableEdge.bs2
' Detectando bordes.

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

irDetectLeft    VAR Bit                 ' Declaración de variables.
irDetectRight   VAR Bit
pulseLeft       VAR Word
```

```
pulseRight      VAR Word
loopCount       VAR Byte
pulseCount      VAR Byte

FREQOUT 4, 2000, 3000                   ' Señal de Inicio/Reset.

DO                                      ' Rutina principal.
  FREQOUT 8, 1, 38500                   ' Comp`rueba los detectores IR.
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                          ' Decidir el movimiento.

  IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    pulseCount = 1                        ' Ambos activados,
    pulseLeft = 850                       ' un pulso de avance.
    pulseRight = 650
  ELSEIF (irDetectRight = 1) THEN         ' Dcha. no activado,
    pulseCount = 10                       ' 10 pulsos a la izda.
    pulseLeft = 650
    pulseRight = 650
  ELSEIF (irDetectLeft = 1) THEN          ' Izda. no activado,
    pulseCount = 10                       ' 10 pulsos a la dcha.
    pulseLeft = 850
    pulseRight = 850
  ELSE                                    ' Ninguno activado,
    pulseCount = 15                       ' retroceso.
    pulseLeft = 650
    pulseRight = 850
  ENDIF

  FOR loopCount = 1 TO pulseCount         ' Enviar pulsos
    PULSOUT 13,pulseLeft
    PULSOUT 12,pulseRight
    PAUSE 20
  NEXT

LOOP
```

Se ha añadido un bucle FOR...NEXT al programa para controlar cuántos pulsos se envían cada vez a través de la rutina principal (DO...LOOP). Se han añadido dos variables, loopCount que funciona como índice del bucle FOR...NEXT y pulseCount que se usa como parámetro EndValue.

```
loopCount       VAR Byte
pulseCount      VAR Byte
```

Las sentencias IF...THEN ahora dan valores a pulseCount así como a pulseRight y pulseLeft. Si ambos sensores pueden ver la mesa, se envía un cauteloso pulso hacia delante.

```
IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
pulseCount = 1
pulseLeft = 850
pulseRight = 650
```

Si el sensor IR de la derecha no ve la mesa, gira 10 pulsos a la izquierda.

```
ELSEIF (irDetectRight = 1) THEN
pulseCount = 10
pulseLeft = 650
pulseRight = 650
```

Si el sensor IR de la izquierda no ve la mesa, rota 10 pulsos a la derecha.

```
ELSEIF (irDetectLeft = 1) THEN
pulseCount = 10
pulseLeft = 850
pulseRight = 850
```

Si ningún sensor ve la mesa, retrocede 15 pulsos y prueba de nuevo, esperando que uno de los sensores vea el borde antes que el otro.

```
ELSE
pulseCount = 15
pulseLeft = 650
pulseRight = 850
ENDIF
```

Ahora que los valores de pulseCount, pulseLeft y pulseRight han sido establecidos, el bucle FOR...NEXT envía el número de pulsos especificado por la variable pulseLeft y pulseRight.

```
FOR loopCount = 1 TO pulseCount
PULSOUT 13,pulseLeft
PULSOUT 12,pulseRight
PAUSE 20
NEXT
```

Puedes experimentar asignando diferentes valores a pulseLeft, pulseRight y pulseCount dentro de las sentencias IF...THEN. Por ejemplo, si el Home Boe-Bot no gira lo suficiente, puede que siga el borde de la mesa. Pivotando hacia atrás en vez de girar se puede conseguir comportamientos interesantes.

Modifica el programa AvoidTableEdge.bs2 para que siga el borde de cinta adhesiva ajustando los valores de pulseCount de tal modo que el Home Boe-Bot no se separe mucho del borde al girar.

Experimenta pivotando como forma de hacer moverse al Home Boe-Bot dentro del perímetro rodeado por la cinta.

8.8 PRUEBA DE AUTOEVALUACION

Cuestiones

¿Qué significa infrarrojos?
¿Cuáles son los dos tipos de filtros incorporados en los sensores IR usados en el Home Boe-Bot? ¿Qué hace cada uno?
¿Cuál es la frecuencia del armónico enviado por FREQOUT 2,1,38500?
¿Qué instrucción tiene que ir inmediatamente a continuación del comando FREQOUT para determinar si se ha detectado o no un objeto?
¿Qué significa que un sensor IR envíe una señal baja? ¿Y una alta?
¿Qué sucede cuando se cambia el valor de una resistencia en serie con un LED? ¿Qué sucede cuando se cambia el valor de una resistencia en serie con un LED IR?
¿En qué se parecen los programas RoamingWithWhiskers.bs2 y RoamingWithIr.bs2? ¿En qué se diferencian?

¿En qué se diferencian RoamingWithIr.bs2 y FastIrRoaming.bs2? ¿Cómo compararías AvoidTableEdge.bs2 con estos dos programas, qué es similar y qué es diferente?

Ejercicios

Si quisieras enviar una señal infrarroja armónica de 39 kHz al LED IR del Home Boe-Bot, ¿qué comando utilizarías?
Modifica una línea de código en IrInterferenceSniffer.bs2 de tal modo que sólo se monitorice uno de los pares LED IR/sensor.
Reescribe la rutina que aplica el pulso de FastIrRoaming.bs2 de tal forma que envíe tres pulsos en vez de uno. Explica cualquier sentencia PBASIC que se necesite previamente en el programa para conseguirlo.
Explica la función de pulseCount en AvoidTableEdge.bs2. ¿Qué tiene que ver con tu respuesta del ejercicio 3?

Proyectos

Diseña un programa para que el Home Boe-Bot permanezca parado hasta que pases tu mano por delante del mismo, momento en el que comenzará a andar.
Diseña un programa para que el Home Boe-Bot rote lentamente hasta que detecte un objeto. Cuando lo detecte, lo seguirá. Este es el clásico comportamiento de los microbots luchadores.
Diseña un programa para que el Home Boe-Bot que deambule, pero que si detecta interferencias infrarrojas haga sonar brevemente el zumbador y luego siga moviéndose. El sonido tienen que ser distinto que el de batería baja.
Añade bigotes al Home Boe-Bot con infrarrojos. Construye un laberinto con varios espacios delimitados por cinta adhesiva. En el centro coloca un objeto. El objetivo del Home Boe-Bot es recorrer el laberinto y encontrar el objeto (por contacto con los bigotes).
Quita los bigotes y añade circuitos de detección de luz. Modifica el laberinto de modo que en cierto sitio luzca una lámpara o algún tipo de luz. El nuevo objetivo del Home Boe-Bot es recorrer el laberinto hasta que encuentre la luz.

Tema 9

Medición de distancia con infrarojos

9.1. PRINCIPIOS PARA LA MEDIDA DE DISTANCIAS

En el capitulo anterior, vimos como conseguir que el Home Boe-Bot detectará obstáculos sin tener que tocarlos utilizando los sensores infrarrojos.¿No sería estupendo que también supiera la distancia que hay hasta el obstáculo? Esto podría solucionarse con un sónar, que envía un onda sonora y en función de la intensidad con la que retorna tras reflejarse y el tiempo que tarda calcula la distancia al obstáculo. Hay una forma de medir dicha distancia con un circuito muy similar al utilizado en el tema 8. El Home Boe-Bot será capaz de determinar la distancia a la que se halla un objeto sin tener que llegar hasta él. También podrás conseguir que el robot siga una línea blanca con lo que convertirás tu Home Boe-Bot en un rastreador.

Partimos del circuito que quedó montado en el tema 8 y necesitarás una regla y una hoja de papel blanco.

9.2. EXPERIENCIA #1 : PROBANDO EL BARRIDO DE FRECUENCIA

La figura 9-1 muestra las especificaciones de nuestro detector de infrarrojos (Panasonic PNA4602M) que es muy sensible a la frecuencia de 38,5 kHz. Si por ejemplo, le enviamos señales a 40 kHz sólo será un 50% sensible de lo que es a la frecuencia recomendada de 38,5 kHz. En las frecuencias que es menos sensible, los objetos deben estar más cerca para que estepuedan ser detectados.

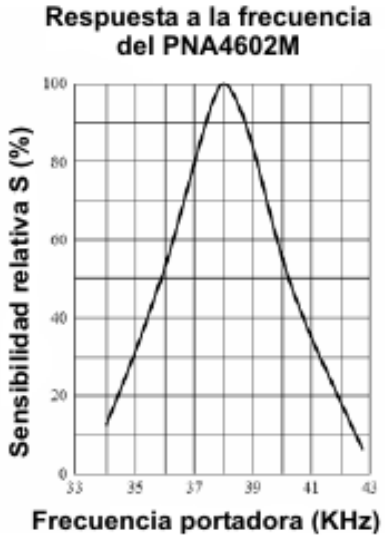


Figura 9-1.- *Nuestro sensor de infrarrojos es muy sensible a la frecuencia de 38,5 kHz y menos cuanto más se desvíe la frecuencia de dicho valor.*

A las frecuencias más sensibles en las que se detectan los rayos infrarrojos, el objeto puede ser detectado más lejos y en las menos sensibles, por el contrario, más cerca. La base sobre la que nos vamos a fundamentar para trabajar con las distancias, consiste en determinar a qué distancias son sensibles o funcionan los infrarrojos, de forma que a 38,5 kHz se detectará a un objeto a la máxima distancia y cuanto más se desvíe la frecuencia de ese valor más cercano deberá estar el objeto para poderlo detectar.

Programando la frecuencia de barrido para el calculo de la distancia

La figura 9-2 nos muestra un ejemplo de cómo el Home Boe-Bot puede calcular las distancias con el uso de distintas frecuencias. En nuestro ejemplo el objeto está en la zona 3. Esto quiere decir que el objeto puede ser detectado cuando el infrarrojo transmite frecuencias de 37500 y 38250Hz pero no así de 39500, 40500 y 41500Hz. Si movemos el objeto a la zona 2, ahora el objeto será detectado cuando se transmitan frecuencias de 37500, 38250 y 39500Hz, pero no cuando sean de 40500 y 41500Hz

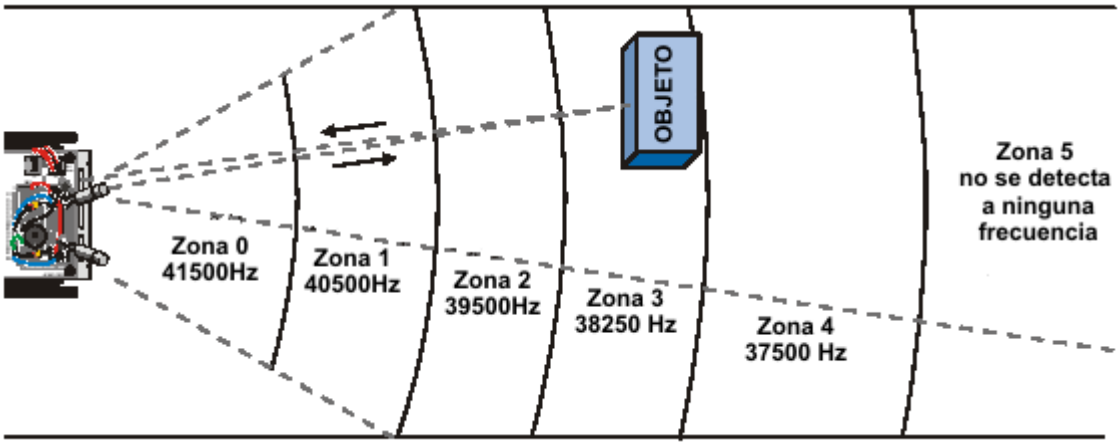


Figura 9-2.- *Según la distancia o zona a la que se encuentre el objeto la frecuencia a la que funcionan los infrarrojos para la detección es diferente.*

Para probar el detector del infrarrojo será necesario el uso de la instrucción FREQOUT, para enviar 5 frecuencias diferentes desde el Home Boe-Bot y comprobar a que distancia cada frecuencia puede detectar un objeto. No es recomendable usar el operador STEP del bucle FOR…NEXT ya que el cambio de frecuencias no sería suficiente. Lo mejor será utilizar las instrucciones DATA y READ. En vez de usar 5 veces la sentencia FREQOUT crearemos una lista de 5 frecuencias para lo cual se utiliza el comando LOOKUP que funciona de la siguiente manera:

LOOKUP Index, [Value0, Value1, …ValueN], Variable

Si el argumento Index es 0, el valor Value0 será el que tome la variable final de la sentencia, si el Index es 1, el valor que tomará en este caso será el de Value1 y así sucesivamente. Para entendernos mejor podemos decir que es un tipo de array en la programación normal. Se propone el siguiente ejemplo:

```
FOR freqSelect = 0 TO 4

        LOOKUP freqSelect, [37500, 38250, 39500, 40500, 41500], irFrequency
        FREQOUT 8,1, irFrequency
        irDetect = IN9
                ' Commands not shown...
NEXT
```

La primera vez que pasamos por el bucle la variable freqSelect es 0, por lo que la variable irFrequency tomará el primer valor que hemos metido en el LOOKUP, es decir, 37500. El comando FREQOUT lo que hace es enviarle este valor al emisor de infrarrojos que tenemos conectado a P8. Así sucesivamente con las 5 frecuencias que hemos metido. Los rayos infrarrojos son muy empleados en las agendas electrónicas, los teléfonos móviles, los mandos de los televisores y otros aparatos, etc..

Programa TestLeftFrequencySweep.bs2

Lo que hace este ejemplo son dos cosas básicamente. Primero comprueba el estado de los infrarrojos (conectados a P8 y P9) para asegurarse que realmente funcionan para la detección de distancia. Cuando ejecutes el programa TestLeftFrecuency.bs2 se mostrará algo similar a la figura 9-3.

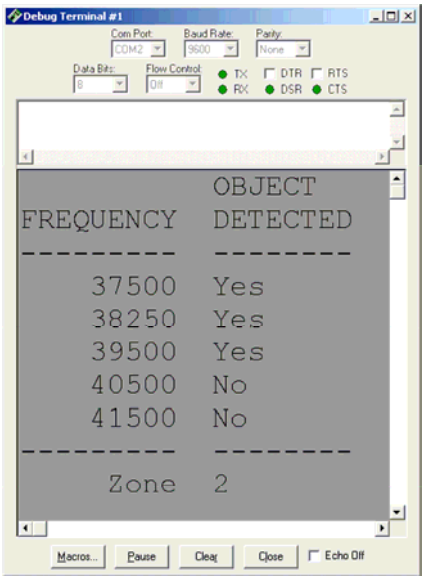


Figura 9-3.- *Ventana que aparece al ejecutar el programa TestLeftFrecuencySweep.bs2.*

- Teclea, guarda y ejecuta el programa TestLeftFrecuencySweep.bs2
- Utiliza una hoja de papel para probar la distancia en la que detecta el infrarrojo.
- Hay que empezar con la hoja muy cerca del infrarrojo (1 cm). La zona en la ventana de Debug deberá corresponder a 0 ó 1.
- Poco a poco aleja la hoja y toma nota de cuándo se hace el cambio de zona, para después poder trabajar con esos datos.

```
' -----[ Título ]-------------------------------------------------------------
' El robot Home Boe-Bot - TestLeftFrequencySweep.bs2
' Comprobar la respuesta a la distancia del detector IR a diferentes frecuencias.

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

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

freqSelect      VAR Nib
irFrequency     VAR Word
irDetect        VAR Bit
distance        VAR Nib

' -----[ Inicialización ]------------------------------------------------------

DEBUG CLS,
    "         OBJETO", CR,
    "FRECUENCIA DETECTAD0", CR,
    "---------- --------"

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

DO

  distance = 0
```

```
FOR freqSelect = 0 TO 4

  LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency
  FREQOUT 8,1, irFrequency
  irDetect = IN9
  distance = distance + 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,
    "Distancia ", DEC1 distance

LOOP
```

Prueba del par de infrarrojos de la derecha

Para probar este par (detector/emisor) de infrarrojos lo único que deberás hacer es cambiar estas dos líneas de código por las que se muestran más adelante.

```
FREQOUT 8,1, irFrequency
irDetect = IN9
```

Cambiar por :

```
FREQOUT 2,1, irFrequency
irDetect = IN0
```

Modifica el programa y ejecútalo.

Comprueba que la detección de los sensores en similar en ambos infrarrojos.

Mostrando ambas distancias.

A veces puede ser interesante crear un programa que nos muestre la distancia que detectan los sensores. Este programa está divido en dos subrutinas que son copias de otros programas que también necesitaban saber distancias.

Programa DisplayBothDistances.bs2

- Teclea, guarda y ejecuta este programa
- Repite la prueba de distancia que has realizado con una hoja de papel en cada infrarrojo y después con los dos a la vez.

```
' -----[ Título ]-------------------------------------------------------------
' El robot Home Boe-Bot - DisplayBothDistances.bs2
' Comprobar la respuesta a la distancia de ambos IR a diferentes frecuencias

' {$STAMP BS2}
```

```
' {$PBASIC 2.5}

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

freqSelect        VAR Nib
irFrequency       VAR Word
irDetectLeft      VAR Bit
irDetectRight     VAR Bit
distanceLeft      VAR Nib
distanceRight     VAR Nib

' -----[ Inicialización ]--------------------------------------------------

DEBUG CLS,
     "ZONA DEL OBJETO", CR,
     "Izda. Dcho.", CR,
     "----- -----"

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

DO

  GOSUB Get_Distances
  GOSUB Display_Distances

LOOP

' -----[ Subroutina - 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

' -----[ Subroutine - Display Distances ]----------------------------------

Display_Distances:
```

```
DEBUG CRSRXY,2,3, DEC1 distanceLeft,
    CRSRXY,9,3, DEC1 distanceRight

RETURN
```

Haz pruebas con hojas de distintos colores y con otros objetos que no sean hojas para que veas que las mediciones varían.

9.3 EXPERIENCIA #2: SIGUIENDO LA ESTELA DE OTRO BOE-BOT

Conseguir que un Home Boe-Bot siga a otro puede conseguirse en base al cálculo de la distancia que los separan. Es decir si el Home Boe-Bot que persigue al otro ve que el primero se va alejando hacia la derecha, seguirá su estela desviándose también a la derecha y manteniendo siempre a la misma distancia.

Normalmente la forma de conseguir en el control automático se mantenga algo de forma continua es realizando bucles con realimentación o cerrados. El caso del Home Boe-Bot no es menos y gracias a lenguaje PBASIC es posible hacerlo de forma sencilla. El diagrama de la figura 9-4 que describe el bucle del que hablamos, se puede realizar en una sentencia PBASIC. Este diagrama de control nos muestra como controlar la distancia en el infrarrojo derecho y que el motor de este mismo lado actúe en consecuencia para mantener esa distancia.

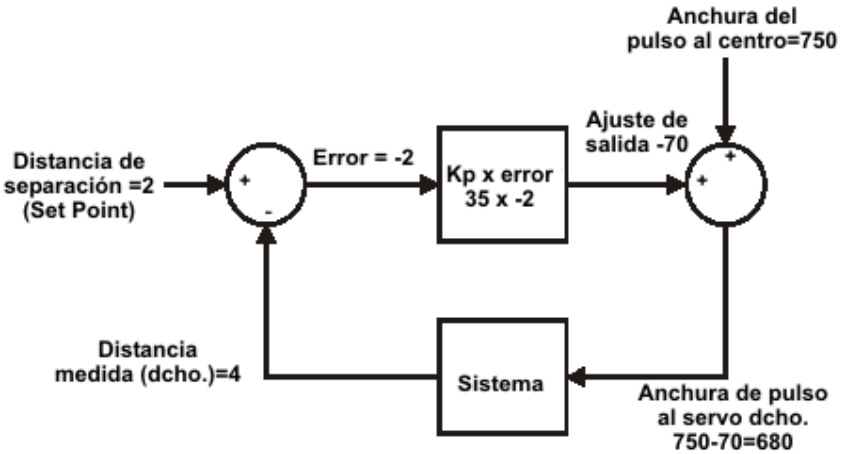


Figura 9-4.- *Diagrama de control para controlar la distancia en el infrarrojo derecho.*

Ahora vamos a explicar un poco más este diagrama de control. Como hemos dicho antes este ejemplo es el específico para el servo motor e infrarrojo derecho. El Set Point se denomina a la distancia que va a guardar el Home Boe-Bot con cualquier objeto que detecte. Hemos elegido el valor de 2. Si la distancia que se mide es 4, se trata de un objeto muy lejano. El error será la resta entre el Set Point y la distancia medida: 2 – 4= -2. Este valor es el que podemos ver en el circulo superior izquierdo de la figura 9-4. A continuación este error lo vamos a multiplicar por una constante proporcional que llamamos (Kp). El valor de esta constante es 35 por lo que obtendremos un resultado de –70, que será un valor que le llamaremos Output adjust (Ajuste de salida). Después realizamos la suma de este valor obtenido con la amplitud del pulso del servo motor que es 750. Por lo tanto, el resultado final es de 680 (amplitud de pulso) que hará que el servo gire en sentido horario a una velocidad ¼ menor de la normal. Con esto conseguiremos que el robot vaya hacia delante, de la misma forma que lo hace el objeto al que sigue. Esta corrección va motivada por el hecho de que el Home Boe-Bot y el Objeto al que persigue están a una distancia medida de 4.

La siguiente vez que pasa por el bucle la distancia de medición puede cambiar, y lo que se calculará será una corrección proporcional al error, el cual es la diferencia entre el Set Point y las distancias medidas.

Se presentan las ecuaciones que se utilizan en el bucle de control mostrado en la figura 9-4.

Error = Distancia establecida (Set Point) derecho - La distancia medida derecho = 2 – 4 =-2
Ajuste de Salida = error × Kp = -2 x 35 = -70
Salida del servo derecho = Ajuste de salida + Amplitud media del pulso = -70 + 750 = 680

Ahora aplicamos todo lo expuesto al infrarrojo y servo del lado izquierdo cuyo funcionamiento es muy similar al de la parte derecha. El diagrama de control es el que se muestra en la figura 9-5.

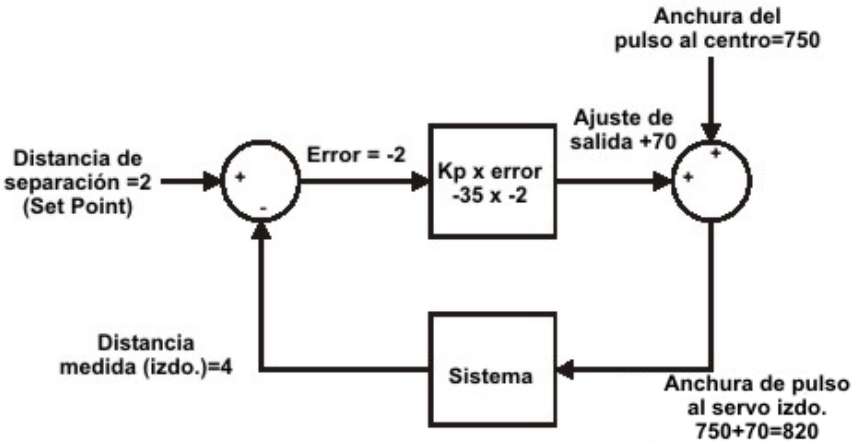


Figura 9-5.- *Bucle de control para el cálculo de la distancia del infrarrojo izquierdo*.

La diferencia radica en la constante Kp, que en este caso es de -35 en vez de 35. Si asumimos la misma distancia medida que en el infrarrojo derecho, el ajuste de salida del servo izquierdo sería de 820 ya que aplicando las ecuaciones, quedará:

Salida del servo izquierdo = (Set Point izquierdo – distancia izquierda medida) × Kp + Amplitud de pulso = ((2–4) × -35) + 750= 820

El resultado es una amplitud del pulso que hace que el motor gire en el sentido antihorario a ¼ de su velocidad normal, es decir vaya hacia delante como nos ha pasado con el otro servo motor. Al realizarse este bucle continuamente (unas 40 veces por segundo) está siempre tomando muestras de distancias por lo que siempre seguirá la estela del objeto al que sigue.

Programando el Home Boe-Bot para que siga a otro

Recuerda que la ecuación para la salida del servo derecho es:

Salida del servo derecho = (Set Point Derecho – Distancia medida derecho) x Kp + Amplitud de pulso

Aquí tienes un ejemplo en PBASIC para resolver esa misma ecuación. La distancia al Set Point derecho es 2, la distancia medida es una variable llamada distanceRight, que almacenará la medición de la distancia realizada por IR, Kp es 35 y la amplitud del pulso es 750:

pulseRight = 2 – distanceRight * (35) + 750

El servomotor izquierdo es diferente porque la constante Kp es –35

pulseLeft = 2 – distanceRight * (-35) + 750

Como los valores –35, 35, 2 y 750 tienen nombres, éste es lugar idóneo para declarar estas constantes:

```
Kpl         CON     -35
Kpr         CON     35
SetPoint    CON     2
```

```
CenterPulse    CON          750
```

Una vez declaradas estas constantes, puedes utilizar sus nombres en vez de los valores numéricos, es decir, en vez de 35 puedes escribir Kpr, por ejemplo. Después de estas declaraciones, los cálculos quedan de la siguiente forma:

PulseLeft = SetPoint – distanceLeft * Kpl + CenterPulse
PulseRight = SetPoint – distanceRight * Kpr + CenterPulse

La principal ventaja que nos aporta el uso de constantes es la posibilidad de cambiar cada valor una sola vez al principio del programa. Cada constante aparece un determinado número de veces a lo largo del programa. Sería muy costoso ir buscando estas ocurrencias e ir cambiando cada valor, sin embargo, una vez declarada la constante, sólo tenemos que ir a la línea de código donde se ha declarado y cambiar ahí su valor. Este cambio se verá reflejado automáticamente en el resto de lugares del programa donde aparezca la constante.

Programa FollowingBoeBot.bs2

FollowingBoeBot.bs2 repite el bucle de control que hemos explicado con cada pulso del servomotor. Es decir, antes de cada pulso se mide la distancia y se determina la señal de error. Después el error se multiplica por la constante Kp y el valor resultante es sumado/restado a la amplitud del pulso enviado al servomotor izquierdo/derecho.

- Teclea, salva y ejecuta el programa FollowingBoeBot.bs2
- Dirige el Home Boe-Bot hacia una hoja de papel situada enfrente de él para que la identifique como un obstáculo. El Boe-Bot debería mantener una distancia determinada con la hoja.
- Prueba a rotar la hoja de papel. El Home Boe-Bot debería rotar con ella.
- Intenta utilizar la hoja para dirigir los movimientos del Home Boe-Bot.
- Acerca la hoja al Home Boe-Bot. Éste debería retroceder.

```
' -----[ Title ]-----------------------------------------------------------
' El robot Home Boe-Bot - FollowingBoeBot.bs2
' El robot ajusta su posición para mantenerse dentro de la zona 2.

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

' -----[ 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

' -----[ Inicialización ]---------------------------------------------------
```

```
FREQOUT 4, 2000, 3000

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

DO

  GOSUB Get_Ir_Distances

  ' Calcula la salida proporcional.

  pulseLeft = SetPoint - distanceLeft * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse

  GOSUB Send_Pulse

LOOP

' -----[ Subroutine - 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

' -----[ Subroutine – Get Pulse ]-------------------------------------------

Send_Pulse:
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5
  RETURN
```

Cómo funciona FollowingBoe-Bot.bs2

FollowingBoeBot.bs2 declara cuatro constantes, Kpr, Kpl, SetPoint y CenterPulse usando la directiva CON. Cada vez que ves la palabra SetPoint a lo largo del programa, en realidad es el número 2 (una constante). Del mismo modo, cada vez que ves otros nombres, realmente lo que se usa es su valor.

```
Kpl          CON        -35
Kpr          CON        35
SetPoint     CON        2
CenterPulse  CON        750
```

Lo primero que hace la rutina principal es llamar a la subrutina Get_Ir_Distances. Cuando la subrutina ha terminado, distanceLeft y distanceRight contienen respectivamente el número correspondiente a la zona en la que se haya detectado un obstáculo.

```
DO
    GOSUB Get_Ir_Distances
```

Las siguientes dos líneas de código implementan los cálculos de control proporcional para cada servomotor.

```
PulseLeft = SetPoint – distanceLeft * Kpl + CenterPulse
PulseRight = SetPoint – distanceRight * Kpr+ CenterPulse
```

Una vez que se han realizado los cálculos se llama a la subrutina de SendPulse

```
GOSUB Send_Pulse
```

La figura 9-6 muestra al Home Boe-Bot principal seguido por el Home Boe-Bot sombra. El robot principal está ejecutando una versión modificada de FastIrRoaming.bs2 y el robot sombra está ejecutando FollowingBoeBot.bs2. Se pueden encadenar hasta 6 ó 7 Home Boe-Bots sombra añadiéndoles los paneles laterales y traseros.

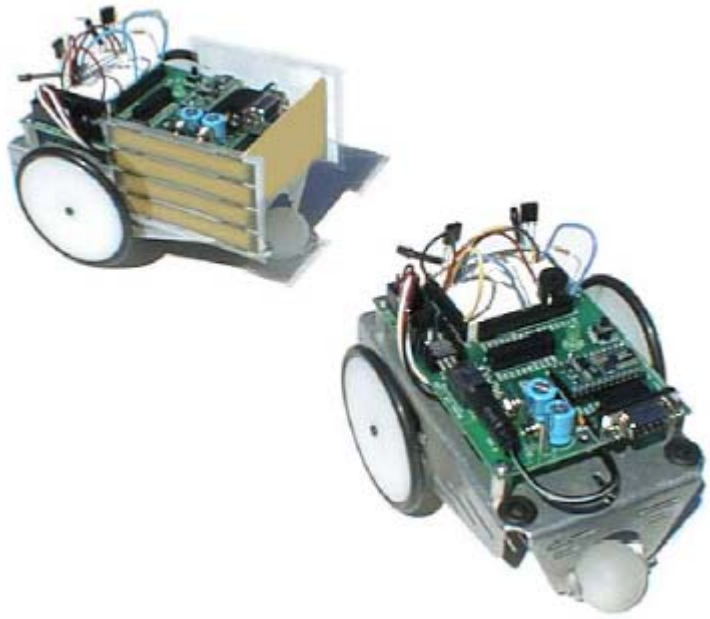


Figura 9-6.- *El Boe-Bot principal perseguido por el robot "sombra".*

- Monta los paneles de cola y laterales en el Home Boe-Bot de cabeza si tienes varios Home Boe-Bots.
- Si no tienes varios Home Boe-Bots, utiliza una hoja de papel a modo de robot principal.
- Sustituye las resistencias de 1 kΩ que conectan los LED IR con P2 y P8 del Home Boe-Bot principal por resistencias de 470Ω y 220Ω.
- Programa el Home Boe-Bot de cabeza para que esquive los obstáculos con una versión modificada de FastIrRoaming.bs2. Abre FastIrRoaming.bs2 y renómbralo como SlowerIrRoamingForLeadBoeBot.bs2
- Haz estas modificaciones a SlowerIrRoamingForLeadBoeBot.bs2: incrementa todos los parámetros **Duración** de 650 a 710 y reduce todos los parámetros **Duración** de 850 a 790.
- El Home Boe-Bot sombra debería ejecutar FollowingBoeBot.bs2 sin modificaciones

- Con cada robot ejecutando su respectivo programa, coloca el Home Boe-Bot sombra detrás del principal. El sombra debería seguir al principal a una distancia fija.

Puedes cambiar los valores de las constantes para conseguir un comportamiento distinto del Home Boe-Bot sombra. Utiliza la mano o una hoja de papel para dirigir al robot sombra mientras pruebas los siguientes ejercicios:

- Ejecuta FollowingBoeBots.bs2 usando valores para las constantes Kpr y Kpl en el rango comprendido entre 15 y 50. Observa la diferencia de comportamiento del Home Boe-Bot mientras sigue un objeto.
- Prueba a hacer ajustes al valor de la constante SetPoint. Utiliza valores entre 0 y 4.

9.4. EXPERIENCIA #3: SEGUIR UNA BANDA

La figura 9-7 muestra un circuito de prueba que puedes construir para que lo siga el Home Boe-Bot. Cada banda del circuito está compuesta por tres tiras de cinta adhesiva negra contiguas sobre fondo blanco. No debe haber separación entre las tres tiras.

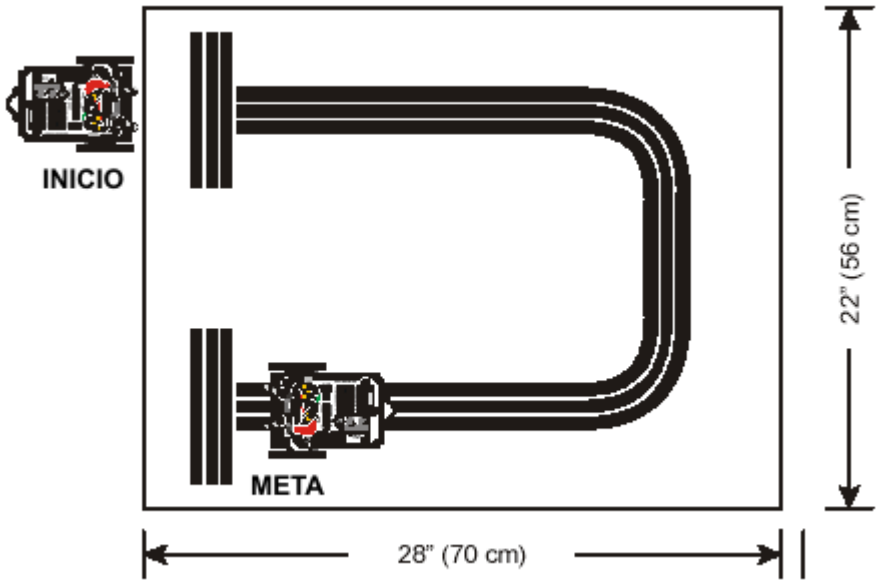


Figura 9-7.- *Circuito de prueba para ser recorrido por el Home Boe-Bot.*

Precisarás los siguientes materiales:

(1) Hoja blanca A3 o de mayor tamaño.
(2) Rollo de cinta adhesiva de color negro de 19 mm de ancho

- Construye el circuito mostrado en la figura 9-7
- Orienta cada par IR hacia fuera y hacia delante como se ve en la figura 9-8

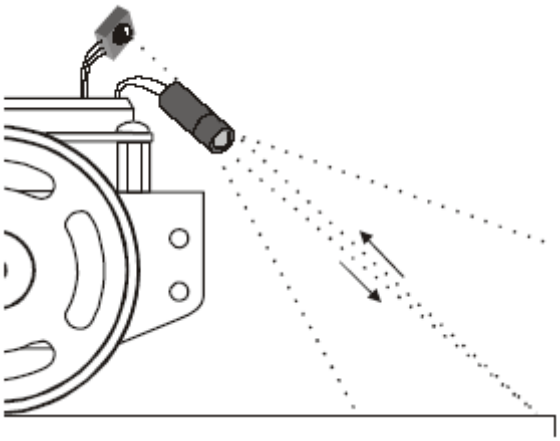


Figura 9-8.- *Cada pareja de emisor/receptor de infrarrojos debe ser orientado hacia abajo y hacia delante.*

- Asegúrate de que el circuito está libre de luz fluorescente que pueda causar interferencias.
- Ejecuta el programa DisplayBothDistances.bs2. Mantén el Home Boe-Bot conectado al cable serie para poder ver reflejadas las distancias.
- Comienza colocando el Home Boe-Bot mirando directamente al fondo blanco del papel como se muestra en la figura 9-9
- Comprueba que las lecturas indican que se está detectando un objeto en una zona muy cercana. Ambos sensores deberían generar un 1 ó un 0.

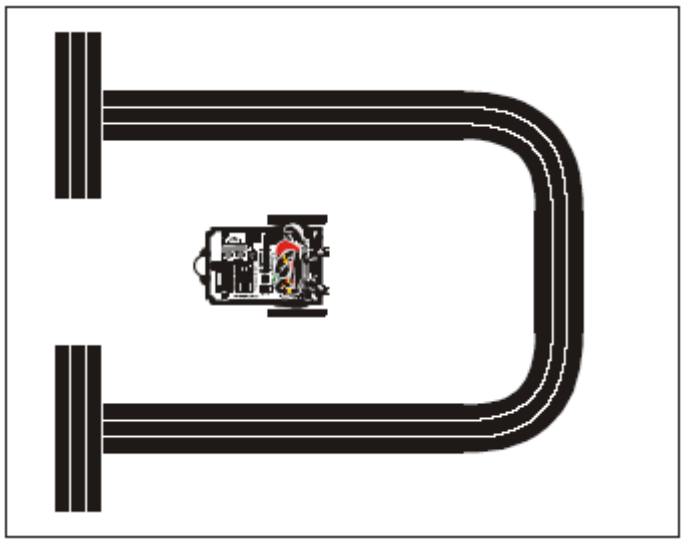


Figura 9-9.- *Se comienza colocando el Home Boe-Bot en el fondo blanco del papel.*

- Coloca el Home Boe-Bot de modo que ambos pares de emisor/receptor IR apunten directamente al centro de las cintas adhesivas (ver figuras 9-10 y 9-11)
- Luego cambia la posición del Home Boe-Bot hasta que ambos valores de zona alcancen el nivel 4 ó 5 indicando que el objeto se está detectando muy lejos o que no se está detectando.

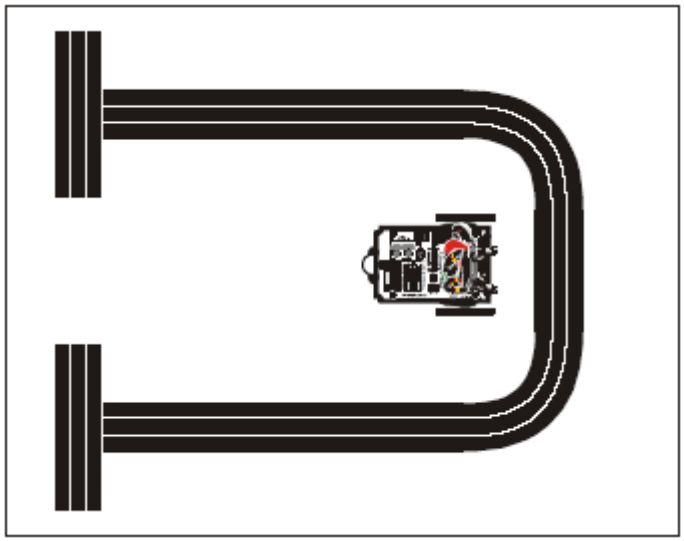


Figura 9-10.- *Los emisores/receptores de IR apuntan al centro de las cintas adhesivas*.

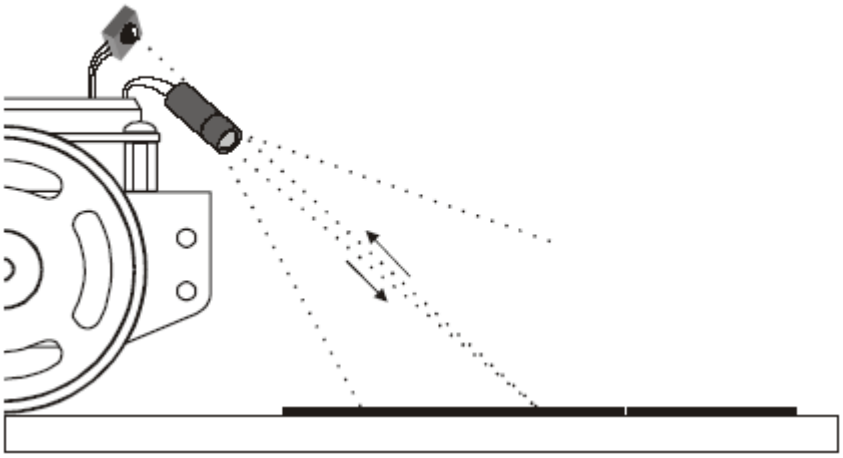


Figura 9-11.- *Orientación de los emisores/receptores de IR sobre las tiras negras adhesivas.*

- Ahora coloca el Home Boe-Bot sobre la línea formada por las tiras adhesivas de la forma que se muestra en la figura 9-12. La lectura de distancia en ambos pares IR debería ser 0 ó 1 de nuevo. Si los valores son mayores tienes que apuntarlos ligeramente más hacia fuera, separándolos más del borde de las tiras.

Cuando mueves el Home Boe-Bot en cualquiera de las direcciones indicadas por la flecha doble, uno de los pares IR apunta hacia la cinta negra haciendo que las lecturas de ese par que está sobre la cinta se incrementen hasta 4 ó 5. Ten en cuenta que si mueves el robot hacia su izquierda, los detectores de la derecha aumentarán el valor de sus lecturas y si lo mueves hacia su derecha, los sensores de la izquierda ofrecerán valores mayores.

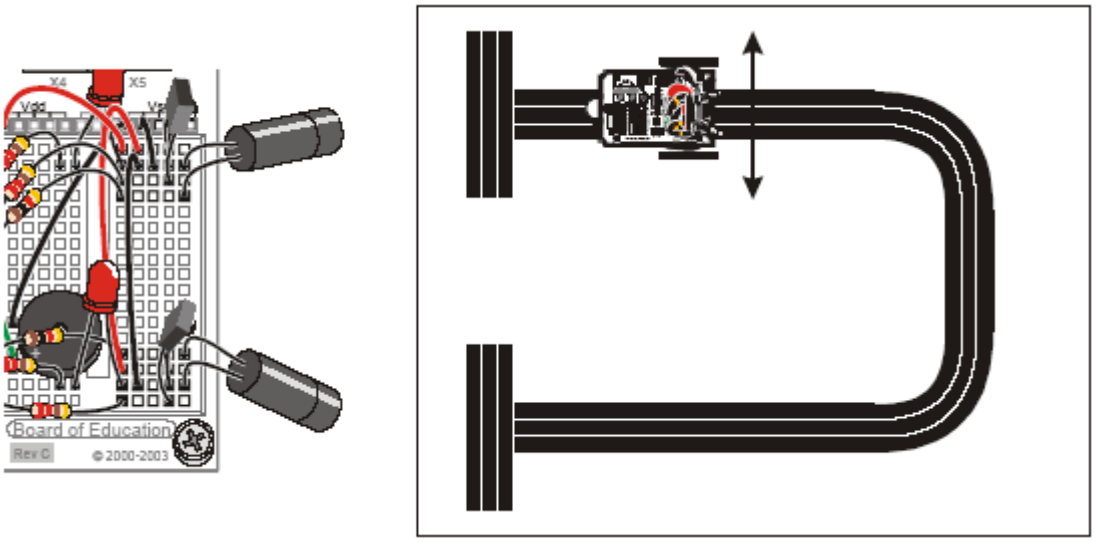


Figura 9-12.- *El Home Boe-Bot colocado sobre la línea formada por las tiras adhesivas.*

Programación para que siga la línea

Para que el Home Boe-Bot siga la línea construida con las tiras adhesivas, sólo tendrás que hacer unas pocas modificaciones en FollowingBoeBot.bs2. Debería moverse hacia los objetos más cercanos que SetPoint y huir de los más lejanos que SetPoint. Este comportamiento es el opuesto al conseguido con FollowingBoeBot.bs2. Para invertir la dirección en la que se mueve cuando detecta que un objeto no está a la distancia SetPoint, simplemente cambia los signos de las constantes Kpl y Kpr, es decir, cambia Kpl de –35 a 35 y Kpr de 35 a –35. Tendrás que experimentar con SetPoint. Los valores de 2 a 4 son los más recomendables. El siguiente programa de ejemplo utiliza un valor de SetPoint de 3.

Programa StripeFollowingBoeBot.bs2

- Abre FollowingBoeBot.bs2 y guárdalo como StripeFollowingBoeBot.bs2
- Cambia en la declaración de SetPoint el valor de 2 a 3
- Cambia Kpl de –35 a 35
- Cambia Kpr de 35 a –35
- Ejecuta el programa
- Coloca el Home Boe-Bot en la "salida" como se ve en la figura 9-13. El robot debería esperar ahí hasta que pongas la mano o la hoja frente a sus IR, momento en el que debería comenzar a avanzar. Cuando haya pasado la salida quita la mano o la hoja, debería seguir el recorrido. Al ver la tira final tendría que detenerse y esperar ahí.
- Suponiendo que logres obtener lecturas de distancias de 5 en la cinta adhesiva y de 0 en la parte blanca, los valores 2, 3 y 4 para SetPoint deberían funcionar. Prueba diferentes valores para SetPoint y observa el rendimiento del Home Boe-Bot en el circuito.

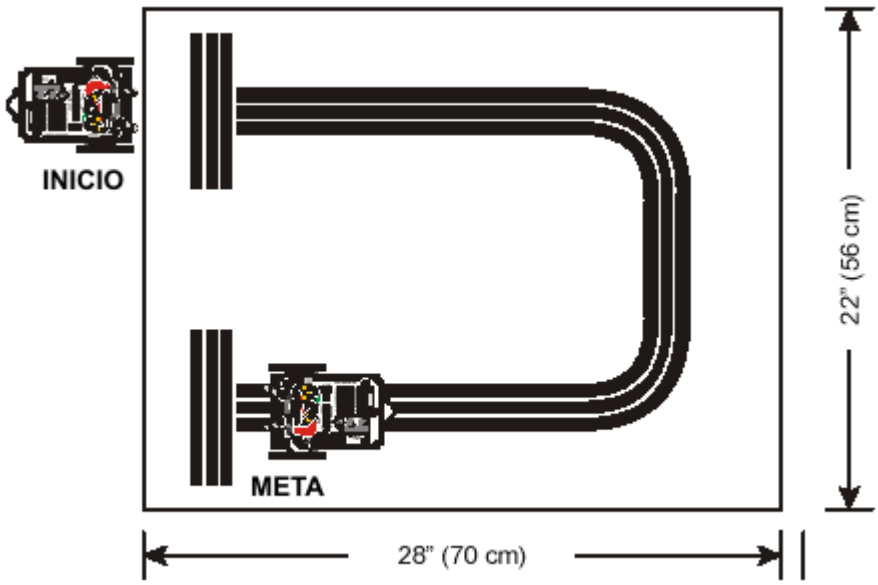


Figura 9-13.- *El Home Boe-Bot colocado a la salida del circuito a recorrer.*

```
' -----[ Título ]-------------------------------------------------------------
' El robot Home Boe-Bot - StripeFollowingBoeBot.bs2
' Siguiendo la línea

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

' -----[ Constants ]-----------------------------------------------------------

Kpl             CON 35                          ' Cambia de -35 a 35
Kpr             CON -35                         ' Cambia de 35 a -35
SetPoint        CON 3                           ' Cambia de 2 a 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

' -----[ Inicialización ]------------------------------------------------------

FREQOUT 4, 2000, 3000

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

DO
```

```
  GOSUB Get_Ir_Distances

  ' Calcula la salida proporcional.

  pulseLeft = SetPoint - distanceLeft * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse

  GOSUB Send_Pulse

LOOP

' -----[ Subroutine - 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

' -----[ Subroutine - Get Pulse ]-------------------------------------------

Send_Pulse:
  PULSOUT 13,pulseLeft
  PULSOUT 12,pulseRight
  PAUSE 5
  RETURN
```

Puedes convertir el circuito anterior en una carrera para obtener el menor tiempo en su recorrido. También puedes crear otros recorridos. Para obtener el mejor rendimiento, experimenta con diferentes valores de SetPoint, Kpl y Kpr.

9.5 PRUEBA DE AUTOEVALUACION

Preguntas

1. ¿Cuál sería la sensibilidad relativa del detector IR si enviases un armónico de 35 kHz? ¿Y de 36 kHz?
2. Si mueves un objeto a la zona 1, ¿qué frecuencias se pueden usar para detectar el objeto y cuáles no? ¿Y en la zona 4?
3. Observa el trozo de código que hay a continuación. Si la variable **index** es 4, ¿qué número pondríamos en la variable **prime** de este comando **LOOKUP**? ¿Qué valores tomará **prime** cuando **index** sea 0, 1, 2 y 7?

   LOOKUP index, [1, 3, 5, 7, 11, 17, 23], prime
4. ¿Qué hace el comando **irDetect = IN9** en los programas de ejemplo de este capítulo?

5. En la figura 9-4 ¿cuál sería el error si el punto de ajuste fuera 4 y la medida de la distancia derecha 1? ¿Cuál sería el ajuste de la salida? ¿Y la salida del servomotor de la derecha?
6. ¿En qué orden son evaluadas las expresiones PBASIC? ¿Cómo puedes forzar otro orden distinto?
7. ¿Qué directivas PBASIC utilizas para declarar una constante? ¿Cómo darías al número 100 el nombre "BoillingPoint"?
8. ¿En qué se diferencian StripeFollowingBoeBot.bs2 y FollowingBoeBot.bs2? ¿Por qué esa diferencia hace que el Home Boe-Bot siga el circuito? ¿Qué ocurre con la distancia de detección si el Home Boe-Bot se desplaza hacia fuera al seguir la línea? ¿Cómo influyen los cálculos proporcionales para que el Home Boe-Bot vuelva al circuito?

Ejercicios

1. Haz un listado con la sensibilidad del IR para cada frecuencia en kHz mostrada en la figura 9-1
2. Haz otro listado de los patrones sí/no que se obtienen con TestLeftFrequencySweep.bs2 cuando un objeto está ubicado en cada zona que se muestra en la figura 9-2
3. Escribe un trozo de código que haga un barrido de frecuencias de cuatro frecuencias en vez de cinco.
4. Realiza los cálculos proporcionales para el servomotor derecho (figura 9-4) para cada posible medida de distancia (0 a 5). Repítelo para la figura 9-5.
5. Haz un listado resumido de las comprobaciones que se deben hacer para asegurar un correcto comportamiento del Boe-Bot siguiendo el circuito.

Proyectos

1. Modifica TestLeftFrequencySweep.bs2 para que utilice el zumbador para indicar la distancia que detectan los sensores. Hay dos posibles formas: la primera es hacer que se oigan pitidos a intervalos regulares y conforme se vaya detectando un objeto más cerca, que el intervalo entre pitidos se vaya reduciendo. La segunda manera es hacer que el zumbador suene constantemente con una frecuencia determinada y cuando se vaya acercando a un objeto, que esa frecuencia se convierta progresivamente en más aguda. Esto es similar a los sistemas de ayuda para el aparcamiento de los automóviles.
2. Diseña un programa para que el Home Boe-Bot vaya mostrando la distancia al objeto por medio del parpadeo de los LED normales izquierdo y derecho.
3. *Avanzado* – Crea diferentes tipos de intersecciones con la cinta adhesiva y programa el Home Boe-Bot para que sepa de cuál se trata en cada caso. Las intersecciones pueden ser de 90º a la izquierda, de 90º a la derecha, de tres caminos, de cuatro caminos... Después de que el Home Boe-Bot detecta una intersección, tiene que usar un movimiento preprogramado para situarse sobre el centro de la misma y determinar si hay cinta o fondo blanco a ambos lados, después moverse un poco más y quizá rotar ligeramente para ver si el camino continúa o no.
4. *Avanzado* – Cuando hayas conseguido realizar con éxito el proyecto anterior, añade movimientos preprogramados para recorrer esas intersecciones. Crea un laberinto de cinta adhesiva. Observa el comando **RANDOM** en el manual de la BASIC Stamp y utilízalo para tomar las intersecciones de tres caminos y de cuatro caminos. La finalidad es asegurarse de que el Home Boe-Bot recorre el laberinto de forma distinta en cada intento.
5. *Avanzado* – Diseña un laberinto inventado por ti y programa el Home Boe-Bot para que salga de él.

Tema 10

GUI Bot, un entorno visual de programación

10.1 INTRODUCCION, ¿QUÉ ES EL GUI Bot ?

Consiste en un entorno visual de programación pensado expresamente para aquellos usuarios que no tienen conocimiento alguno de programación pero quieran introducirse en el apasionante mundo de la microbótica.

Diseñado para los robots "Home Boe-Bot" y "Boe-Bot" dotados del BASIC Stamp 2, dispone de un conjunto de iconos cada uno de los cuales representa una instrucción o acción básica. Seleccionando los iconos deseados en función de la tarea a realizar, se confecciona una lista de instrucciones o programa que se cargará en el robot para su ejecución. Todo ello en un entorno gráfico y muy intuitivo que trataremos de explicar en el presente tema.

El software GUI Bot es de libre distribución y se puede descargar desde las siguientes direcciones web: www.parallax.com/dl/sw/GUIBot.exe o www.microcontroladores.com.

Su instalación en el PC no presenta ninguna dificultad. Una vez descargado desde la red el programa GUIBot.exe, lo ejecutamos y seguimos las diferentes instrucciones que aparecen en pantalla. Una vez instalado, nos aparecerán dos accesos directos en nuestro escritorio como los mostrados en la figura 10-1. El de la izquierda llama al programa GUI Bot propiamente dicho y lo ejecuta. El de la derecha abre un fichero PDF que, a modo de manual, explica los pasos fundamentales para el manejo del software GUI Bot.

Figura 10-1. *Iconos del software GUI Bot una vez instalado.*

10.2 CONFIGURACION DEL ROBOT "HOME BOE-BOT"

El software GUI Bot es capaz de gestionar los dos tipos fundamentales de sensores que puede soportar el robot Home Boe-Bot, para la detección de objetos u obstáculos: los sensores IR y los bumpers o "bigotes". Por defecto el software asume que ambos tipos de sensores están instalados. A pesar de ello, con algunas sencillas modificaciones, el robot puede tener tres configuraciones posibles: Disponer de ambos tipos de sensores; Emplear únicamente los sensores IR; disponer únicamente de los bumpers o "bigotes".

10.2.1 Opción 1: El Robot dispone de ambos tipos de sensores

La figura 10-2 muestra el esquema eléctrico de conexión tanto de los bumpers (izdo y dcho) como de los sensores IR (izdo y dcho), y se corresponde con los esquemas que se explicaron en los temas 6 y 8 de este manual. Estos sensores se deben conectar forzosamente a las líneas de E/S indicadas en el esquema y no a ninguna otra ya que el software GUI Bot lo asume así por defecto.

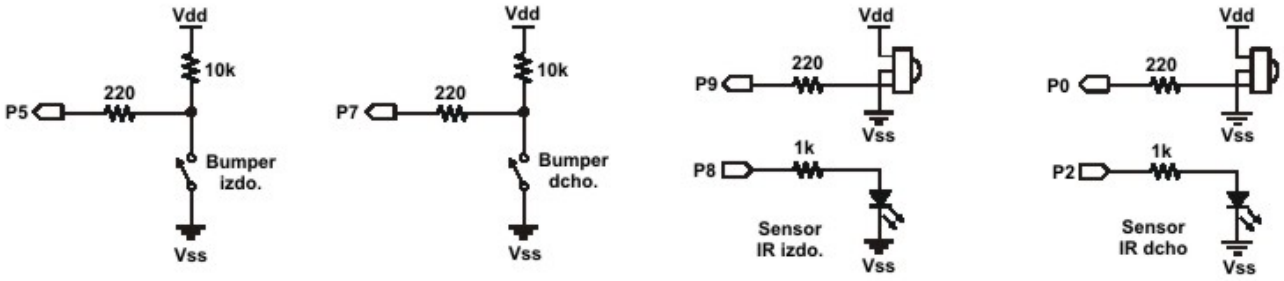


Figura 10-2. *Esquema eléctrico de conexión de ambos tipos de sensores*

En la figura 10-3 se puede apreciar la disposición de componentes sobre la board de la tarjeta de control del robot, según el anterior esquema eléctrico de conexiones.

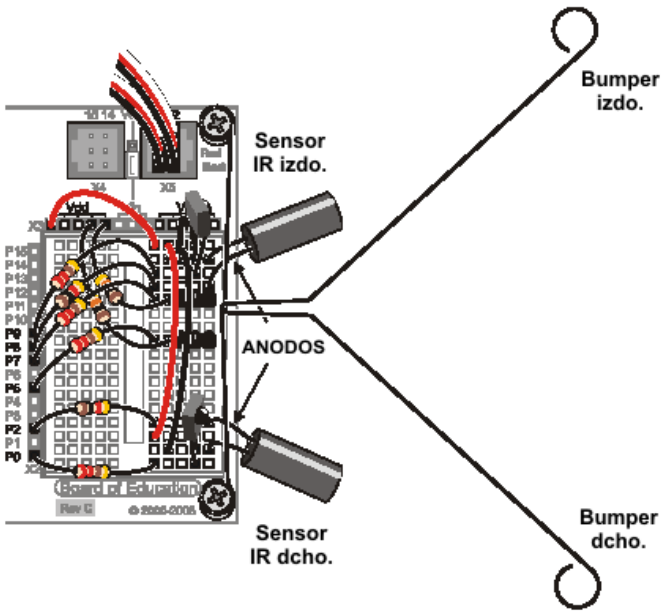


Figura 10-3. *Montaje práctico de ambos tipos de sensores*

10.2.2 Opción 2: Emplear sólo los sensores IR

El Home Boe-Bot puede emplear únicamente los sensores IR para detectar objetos u obstáculos, tal y como se explica en el tema 8. De todas formas, para poder hacer un uso correcto del software GUI Bot, es necesario añadir dos resistencias de 10KΩ entre las líneas de E/S P7 y P5 y Vdd. La figura 10-4 muestra el aspecto del montaje sobre la board de la tarjeta de control.

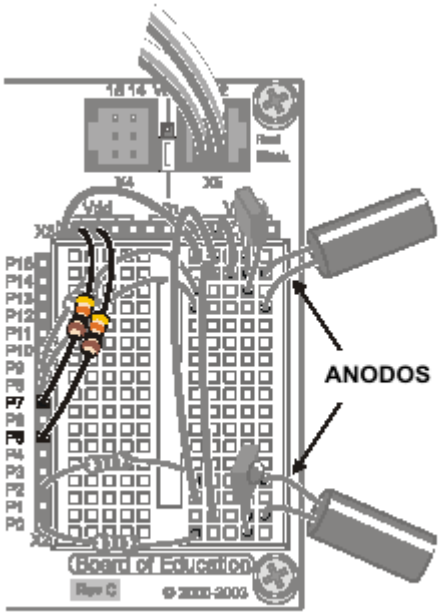


Figura 10-4. *Montaje del circuito para el empleo de los sensores IR*

10.2.3 Opción 3: Empleo únicamente de los bumpers

En esta ocasión el robot sólo emplea los bumpers o "bigotes" para la detección de objetos. El circuito es similar al que ya se explicó en el tema 6. Para que el software GUI Bot pueda gestionar correctamente el estado de los sensores es necesario conectar dos resistencias de 10KΩ entre las líneas P9 y P0 y Vdd, tal y como se muestra en la figura 10-5.

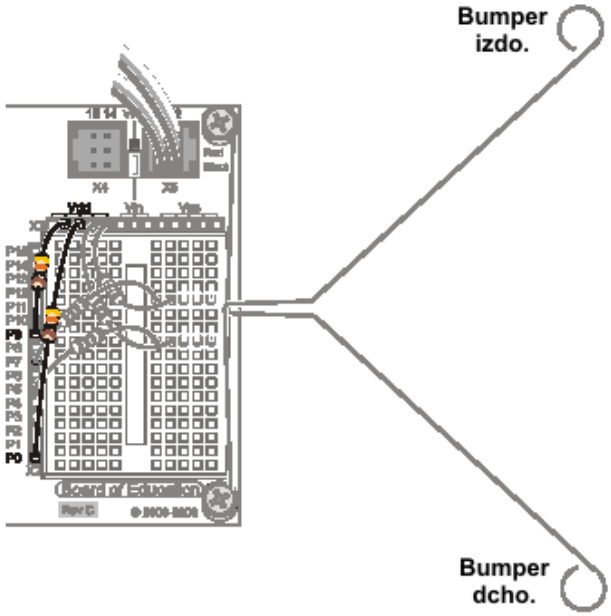


Figura 10-5. *Vista del montaje para el empleo de los bumpers*

Como veremos más adelante, el software GUI Bot dispone de un modo de trabajo, el modo Test, que permite comprobar el correcto funcionamiento de los sensores y motores del robot Home Boe-Bot.

10.3 EJECUTANDO EL SOFTWARE GUI Bot

Siempre que se ejecuta el software GUI Bot debemos elegir entre dos modos de trabajo: Básico o avanzado. Ver la figura 10-6.

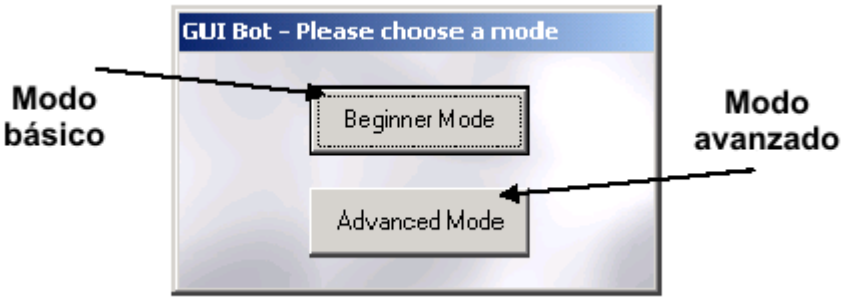


Figura 10-6. *Selección de los modos de trabajo*

El modo básico está pensado para aquellos usuarios que no tienen experiencia en la programación y/o es la primera vez que manejan este software. Dispone de una serie de simples acciones o instrucciones relativas al movimiento del robot. No se contempla el manejo de los sensores. Estos, si se desea, son controlados automáticamente por el propio software para evitar los posibles obstáculos.

En el modo avanzado se dispone de las mismas acciones que en el básico y, además, se puede controlar los sensores de entrada para llevar a cabo diferentes tareas en función de su estado.

10.4 EL MODO BASICO

Aparece una pantalla como la mostrada en la figura 10-6.

Figura 10-6. *La pantalla de GUI Bot en el modo básico*

La pantalla se divide en tres secciones para el modo básico. A la izquierda disponemos de un conjunto de iconos que representan las 8 acciones o instrucciones posibles relativas al movimiento del robot. Al centro disponemos de una sección donde se va confeccionando la lista de acciones o instrucciones que el robot debe ir ejecutando (programa). A la derecha aparece un panel de control con el que se selecciona el puerto serie que se va a emplear en la comunicación entre el PC y el robot Home Boe Bot. En esta sección también podemos activar o no a los sensores. Si, en el modo básico dichos sensores están activados, el robot tratará de esquivar obstáculos de forma automática (se recuerda que en el modo básico el usuario no tiene control sobre los sensores de entrada). En caso contrario el robot se limitará a ejecutar las acciones que el usuario haya incluido en la lista de acciones (programa).

10.4.1 Acciones disponibles

Se muestran en la figura 10-7.

Figura 10-7. *Iconos con las acciones posibles de movimiento.*

Todas estas instrucciones, excepto la de repetición, provocan el movimiento (o parada) del robot en la dirección indicada durante un cierto tiempo. La instrucción de Repetición permite retroceder en la lista de acciones (programa) de forma que se vuelvan a repetir y ejecutar todos los movimientos a partir del paso indicado.

Las instrucciones de giro a la izquierda o derecha hacen que el robot se mueva en ese sentido al tiempo que sigue avanzando. Las instrucciones de rotación hacen que el robot gire sobre su propio eje sin avance alguno.

10.4.2 La lista de acciones

Esta lista se va confeccionando, paso a paso, con las diferentes instrucciones que deseamos ejecute el robot, empezando por la primera y continuando secuencialmente hasta la última.

Para añadir una instrucción basta con arrastrar el icono correspondiente a la posición deseada dentro de la lista. También se puede hacer doble clic sobre el icono y su acción se almacena automáticamente al final de la lista.

Una vez que cualquier instrucción se deposita en la lista, aparece un valor numérico asociado a ella. Este valor representa el tiempo que debe durar la ejecución de la instrucción, y se puede modificar tecleando directamente un nuevo valor o bien "picando" en las flechas ↑ y ↓, que producen incrementos/decrementos de décimas de segundo. El rango válido va de 0,1" a 25,5".

En la figura 10-8 se muestra, a modo de ejemplo, una instrucción de avance durante 4 segundos insertada en la lista de acciones.

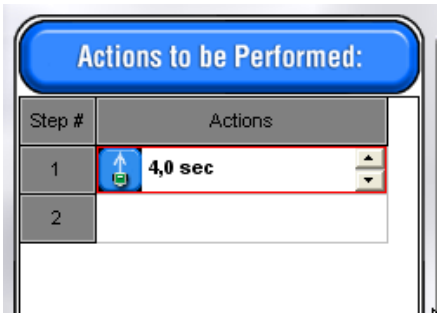


Figura 10-8. *Instrucción de avance durante 4 segundos*

10.4.3 Ejemplo1

El sencillo ejemplo que se propone consiste en hacer avanzar y retroceder al robot de forma continua. La figura 10-9 nos presenta la lista de acciones que vamos a realizar. **¡¡¡ Esto es un programa !!!**

El primer paso (Step 1) consiste en hacer avanzar al robot durante 2". El siguiente paso mantiene al robot parado (stop) durante 1". Seguidamente, en el paso 3, el robot retrocede durante otros 2", por lo que se queda donde estaba al principio. Se realiza una nueva parada de 1" en el paso 4. Finalmente, el paso 5 es una acción o instrucción de repetición que vuelve, en este ejemplo, al paso 1. Todo el proceso se vuelve a repetir de forma indefinida.

Figura 10-9. *Ejemplo 1, nuestro primer programa*

Ahora sólo queda descargar el programa anterior sobre la memoria del robot, para que pueda ser ejecutado. Esto se realiza mediante el botón **GO** que aparece en la pantalla principal del software GUI Bot. Se supone que el robot Home Boe Bot está debidamente alimentado y conectado al canal serie del PC.

10.4.4 Ejemplo 2

La figura 10-10 muestra otro ejemplo . En esta ocasión se trata de una serie de movimientos para realizar una determinada trayectoria.

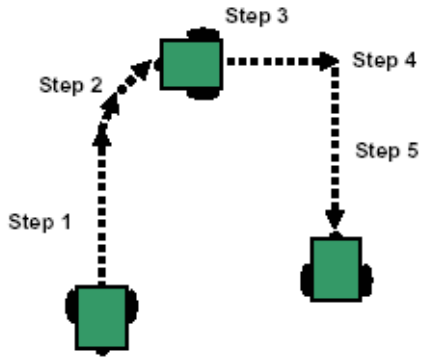


Figura 10-10. *Ejemplo 2, realizando una trayectoria*

En este ejemplo debemos notar la diferencia que hay entre el giro a la derecha del paso 2 y la rotación del paso 4. En el paso 2, además de girar se produce un avance. En el paso 4, el movimiento de rotación se realiza sobre el propio eje del robot, sin que se realice ningún tipo de avance.

10.4.5 Los sensores en el modo básico

Como ya se ha explicado anteriormente, en el modo básico el usuario no tiene control sobre los sensores de entrada. Sin embargo existe la posibilidad de que estos sean gestionados automáticamente por el propio software GUI Bot, según se van ejecutando las instrucciones de la lista realizada por el usuario. Efectivamente, si activamos la opción "Sensors Enabled" como se presenta en la figura 10-11, se consigue que cuando se detecte un obstáculo a la derecha del robot los esquive girando a la izquierda y viceversa.

Si se desactiva esta opción no se realiza gestión alguna de esos sensores. El robot se limita a ejecutar las instrucciones almacenadas por el usuario en la lista de acciones.

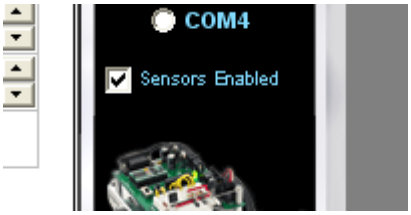


Figura 10-11. *Activando el control de los sensores*

10.4.6 Modificando la lista de acciones

La lista de acciones no es ni mas ni menos que un programa con un conjunto de instrucciones que se ejecutarán secuencialmente en el robot. Es posible que en ocasiones sea necesario editar dicho programa o lista, de forma que se puedan insertar, mover, borrar, etc las diferentes instrucciones de que consta.

La figura 10-12 muestra los diferentes botones que permiten la edición del programa almacenado en la lista de instrucciones.

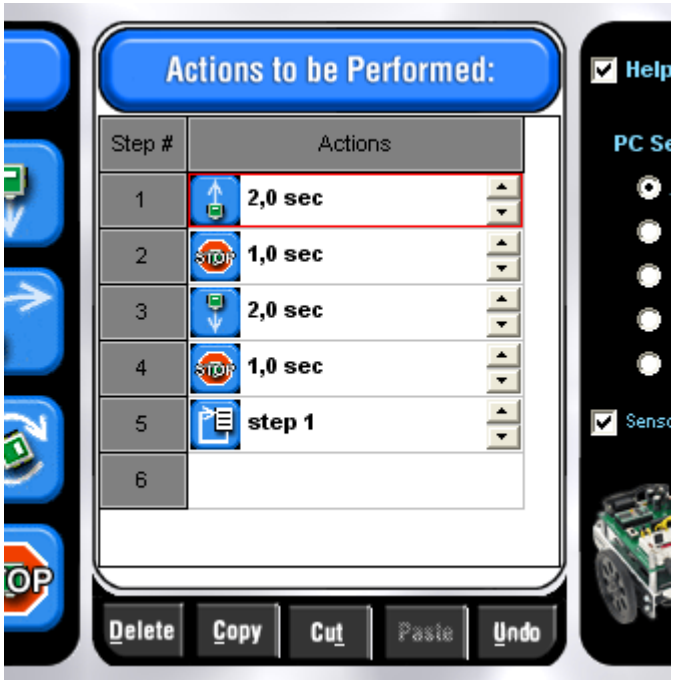


Figura 10-12. *Botones para la edición de un programa*

La tabla que se presenta a continuación resume la función de cada uno de los botones que permite editar o modificar una lista de acciones o programa.

BOTÓN	DESCRIPCIÓN
Delete	Borra de la lista la instrucción seleccionada. Para seleccionar una instrucción basta con hacer clic sobre la celda que contiene el número que la identifica (Step #). Una vez borrada la instrucción todas las que están por debajo se desplazan hacia arriba, quedando siempre agrupadas.
Copy	Copia la instrucción seleccionada en el porta papeles para poderla copiar posteriormente en otro lugar de la lista.
Cut	Borra la instrucción seleccionada y la almacena en el porta papeles para poderla copiar posteriormente otro lugar de la lista.
Paste	Pega la instrucción almacenada en el porta papeles sobre la lista, por encima de la instrucción seleccionada. Todas las instrucciones se desplazan una posición hacia abajo quedando siempre agrupadas.
Undo	Anula la última acción llevada a cabo. Es decir, si por ejemplo se borra accidentalmente una instrucción de la lista mediante el botón Delete, al pulsar Undo se recupera la instrucción borrada.

10.4.7 Botones de control

Son seis botones que se muestran la figura 10-13.

Figura 10-13. *Botones de control*

BOTÓN	DESCRIPCIÓN
Go	Cuando se acciona este botón la lista de instrucciones se transfiere directamente al robot y quedan almacenadas en el mismo. Comienza entonces la ejecución real del programa.
Clear	Borra toda la lista actual de instrucciones.
Test	Activa el modo Test en el que se permite la comprobación de los sensores y motores del robot Home Boe Bot
Load	Permite cargar desde disco un fichero que contiene una lista de instrucciones o programa. Los ficheros gestionados por el software GUI Bot tienen la extensión *.GBT
Save	Almacena la lista de instrucciones actual sobre un fichero con extensión *:GBT
Quit	Termina la ejecución del software GUI Bot.

10.4.8 El Modo TEST

Este modo es ideal para comprobar el hardware de nuestro robot y nos permite comprobar rápidamente el funcionamiento de los motores y sensores. Se activa al accionar el botón de control **Test** y muestra una nueva pantalla que representa al robot. Ver la figura 10-14.

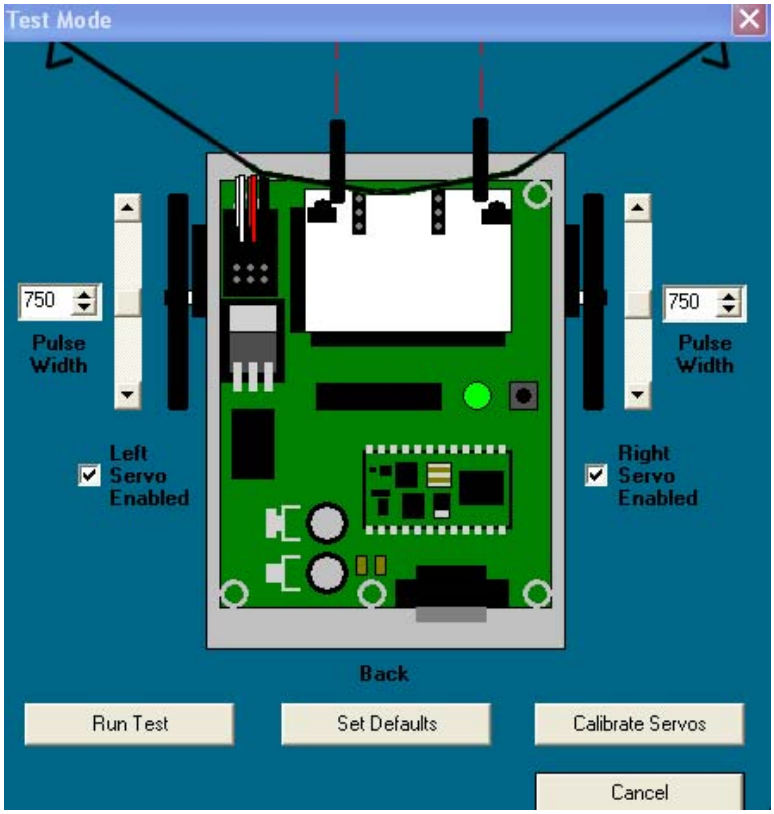


Figura 10-14. *Pantalla del modo Test*

Cuando se activa el botón "Run Test" se descarga sobre el robot un programa que, en tiempo real, lee el estado de los sensores y los transmite al PC, donde se visualizan de forma gráfica. Este a su vez transmite a ambos motores del robot sendos pulsos de anchura variable, lo que permite analizar el movimiento de los mismos.

El usuario puede observar cómo la activación o no de los sensores del robot, bien sean los bumpers o los IR, tienen un reflejo inmediato en la pantalla del modo test. Igualmente se puede variar la anchura de los pulsos aplicados a ambos motores y observar cómo estos giran en ambos sentidos.

Cuando se consigue que un motor esté totalmente detenido, se dice que dicho motor está calibrado. El valor de la anchura del pulso que se está aplicando en ese momento, recibe el nombre de "valor de calibración". Este valor puede registrarse internamente cuando se acciona el botón "Calibrate Servos" de la pantalla de Test.

El botón "Set Defaults" configura el modo Test con los valores por defecto. Por último, el botón "Cancel", finaliza la ejecución del modo Test devolviendo el control al software GUI Bot.

10.5 EL MODO AVANZADO

En el modo avanzado del software GUI Bot se dispone de las mismas opciones, instrucciones y posibilidades que en el modo básico y, además, el usuario puede gestionar el estado de los sensores de entrada con objeto de interactuar en base a ellos. Es decir, el robot podrá realiza diferentes movimientos o maniobras en función del estado de esos sensores. En la figura 10-15 se muestra la pantalla de trabajo en el modo avanzado del GUI Bot.

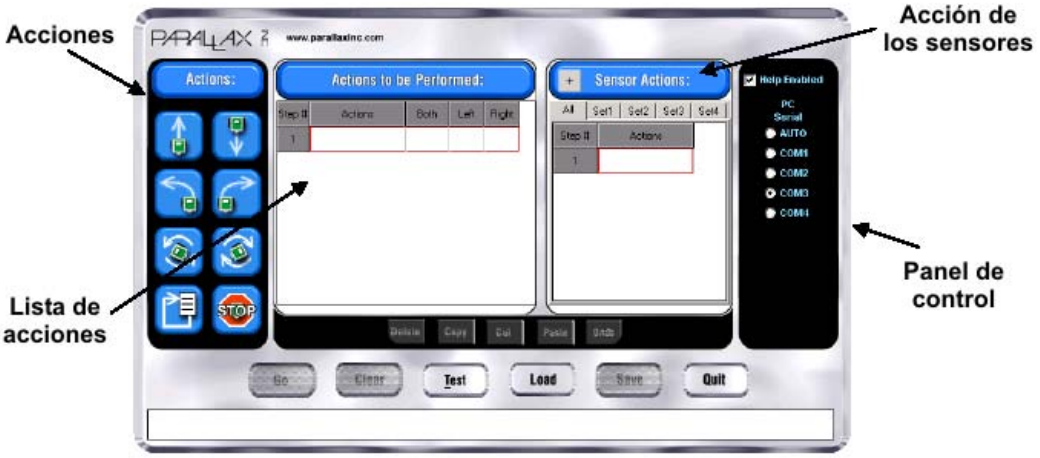


Figura 10-15. *Pantalla de trabajo en el modo avanzado*

Aparece una nueva columna denominada "Acción de los sensores" que permite confeccionar hasta 5 listas de instrucciones diferentes según qué pestaña se seleccione: All, Set1, Set2, Set3 y Set4.

En la lista de acciones (programa) aparecen tres nuevas columnas llamadas "Both", "Left" y "Right", en las que el usuario indicará qué hacer si se activan ambos sensores (Both), sólo el de la izquierda (Left) o sólo el de la derecha (Right), cuando se están ejecutando las instrucciones del programa. En estas columnas se debe indicar qué lista de instrucciones se debe ejecutar de las 5 posibles (All, Set1, Set2, Set3 y Set4), cuando se activa cualquiera de los sensores (Both, Left o Right). No se puede hacer distinción entre los sensores IR y los bumpers.

Supongamos el siguiente ejemplo:

- Si se activan ambos sensores se deben realizar las siguientes maniobras:

  1.- Parar el robot
  2.- Mover el robot hacia atrás
  3.- Rotación a la derecha 180º aproximadamente

- Si se activa el sensor de la izquierda se deben ejecutar las siguientes maniobras:

  1.- Parar el robot
  2.- Mover el robot hacia atrás
  3.- Rotación a la derecha

- Si se activa el sensor de la derecha se deben ejecutar las siguientes maniobras:

  1.- Parar el robot
  2.- Mover el robot hacia atrás
  3.- Rotación a la izquierda

Tenemos pues tres listas de instrucciones que se deben ejecutar en otros tantos casos. Dichas listas las confeccionamos sobre la columna "Acción de los sensores", en las pestañas All, Set1 y Set2, tal y como se muestra en la figura 10-16.

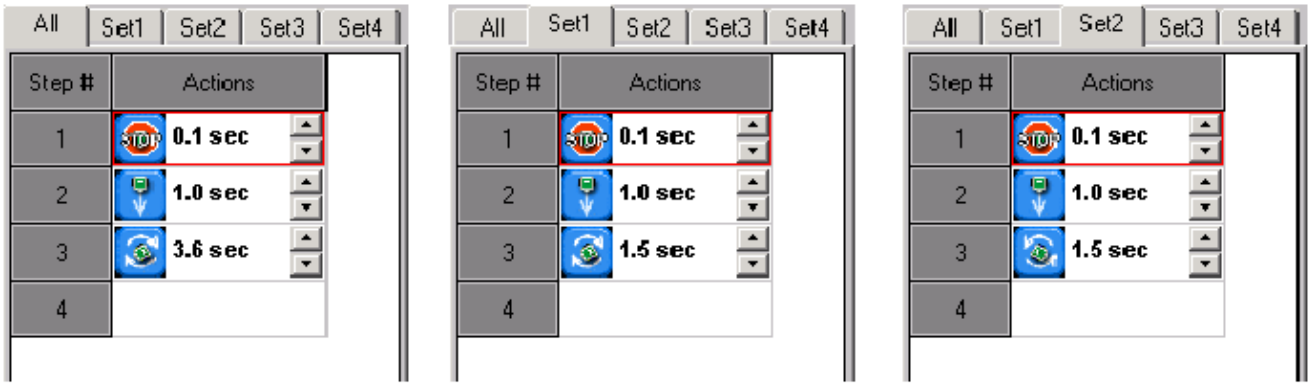


Figura 10-16. *Confección de las tres listas de "Acción de los sensores"*

Seguidamente, en la columna de acciones vamos a confeccionar el programa principal, que consiste en hacer avanzar el robot, girar a la derecha y seguir avanzando, todo ello teniendo en cuenta el estado de los sensores. Observemos el programa de ejemplo de la figura 10-17.

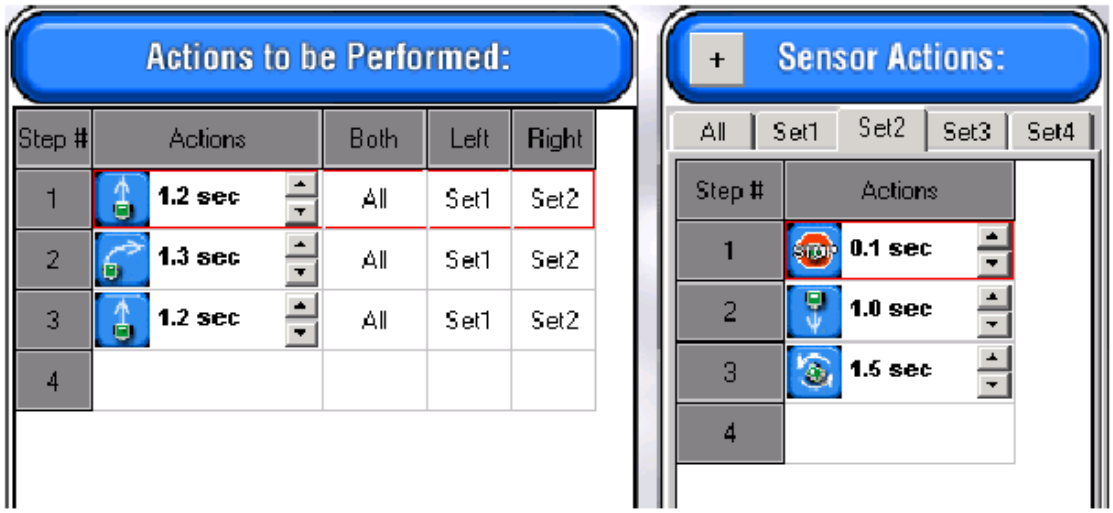


Figura 10-17. *El programa de ejemp*lo

En la primera instrucción (Step #1) se hace avanzar al robot durante 1,2". Si se activan ambos sensores (Both) se ejecutan las instrucciones contenidas en la lista All de "Acción de los sensores". Si se activa el sensor de la izquierda (Left) se ejecutan las instrucciones de la lista Set1. Finalmente, si se activa el sensor de la derecha (Right), se ejecutan las instrucciones de la lista Set2.

En las restantes instrucciones del programa principal (Step #2 y Step #3) se sigue el mismo esquema que en la primera instrucción.

Para relacionar las columnas Both, Left y Right del programa principal con cualquiera de las listas de "Acción de los sensores" basta seleccionar la pestaña de una de estas y arrastrar el icono con el signo + hasta cualquiera de las columnas Both, Left y Right del programa principal.

En otras palabras. Se pueden crear hasta 5 listas de instrucciones diferentes en la columna "Acción de los sensores". En la lista del programa principal se relaciona cada instrucción con la lista que se debe ejecutar en el supuesto de que se activen cualquiera de los sensores (Bth, Left o Right).

Anexo 1

Resumen del repertorio de instrucciones PBASIC

Desde la página principal de Parallax, www.parallax.com, se puede descargar el manual completo del PBASIC, donde podrás encontrar toda la información detallada y actualizada tanto de las instrucciones PBASIC como de los módulos BasicStamp. En este anexo le resumimos las características de las instrucciones PBASIC.

AN1.1 INSTRUCCIONES DE CONTROL y SALTO

Instrucción	***BRANCH***
Sintaxis	***BRANCH Offset, (dir0, dir1, ... dirn)***
Descripción	Realiza un salto a cualquiera de las direcciones (dir0, dir1, ... dirn) en función del valor del offset.
Ejemplo	***BRANCH valor,(Uno, Dos, Tres)*** Salta a la dirección Uno o Dos o Tres en función del contenido de la variable "Valor" (1,2 o 3)

Instrucción	***GOSUB***
Sintaxis	***GOSUB Dir***
Descripción	Almacena la dirección de la siguiente instrucción a la propia GOSUB, con objeto de poder retornar a ella. A continuación salta a la subrutina cuyo inicio se expresa en el parámetro Dir.
Ejemplo	***GOSUB Rutina1*** Salta ejecutar las instrucciones que comienzan en la dirección o etiqueta "Rutina1"

Instrucción	***GOTO***
Sintaxis	***GOTO Dir***
Descripción	Salta al punto del programa indicado o representado por el parámetro Dir. No se almacena la dirección de partida por lo que el retorno automático no es posible.
Ejemplo	***GOTO Tarea1*** Salta a ejecutar las instrucciones que comienzan en la dirección representada por "Tarea1"

Instrucción	***IF... THEN***
Sintaxis	***IF condición THEN dir***
Descripción	Evalúa la condición. Si esta es cierta, se salta al programa apuntado por "dir". En caso contrario se continua con la ejecución del programa. Las condiciones pueden ser de tipo aritmético o lógico
Ejemplo	***IF A=3*2 THEN Tarea1*** Ejecuta instrucciones desde "Tarea1" si el valor de la variable A es igual a 6

Instrucción	***IF... THEN ... ELSE***

Sintaxis	***IF condición THEN***
	Instrucción(es)
	{ ELSEIF condición (es) THEN
	Instrucción(es) }
	{ ELSE
	Instrucción(es) }
	ENDIF

Descripción	Evalúa la condición. Si es cierta se ejecutan las instrucciones que van a continuación de THEN. En caso contrario se evalúa la condición de ELSEIF. Si no hubiera condición ELSEIF se salta y ejecutan las instrucciones que van tras ELSE. Si no hubiera ELSE se ejecutan las instrucciones que haya tras ENDIF

Ejemplo	***IF (A > 40) THEN***
	DEBUG "A es mayor de 40"
	ELSEIF (V=40) THEN
	DEBUG "A es igual a 40"
	ELSE
	DEBUG "A es menor de 40"
	ENDIF

Instrucción	***ON ... GOSUB***
Sintaxis	***ON offset GOSUB dir1, dir2, … dirn***
Descripción	Salta a la subrutina indicada por el valor del offset. Previamente se guarda el lugar de partida para un posterior retorno
Ejemplo	***ON A GOSUB Sub_uno, Sub_dos, Sub_tres*** En función del valor de la variable A (1, 2 o 3) se salta a la subrutina que empieza en "Sub_uno" o en "Sub_dos" o en "Sub_tres".

Instrucción	***ON ... GOTO***
Sintaxis	***ON offset GOTO dir1, dir2, ... dirn***
Descripción	Salta a la dirección indicada por el valor del offset
Ejemplo	***ON A GOSUB Tarea1, Tarea2, Tarea3*** En función del valor de la variable A (1, 2 o3) se salta a la "Tarea1" o a "Tarea2" o a la "Tarea3".

Instrucción	***RETURN***
Sintaxis	***RETURN***
Descripción	Retorno desde una subrutina. Se vuelve al lugar del que se partió
Ejemplo	***RETURN***

Instrucción	***SELECT ... CASE***
Sintaxis	***SELECT Expresión*** ***CASE condición(es)*** ***Instrucción(es)*** ***{ CASE condición(es)*** ***Instrucción(es)*** ***CASE ELSE*** ***Instrucción(es) }*** ***ENDSELECT***
Descripción	Evalúa la Expresión y condicionalmente ejecuta un bloque de instrucciones basado en la comparación de la condición. Tras ejecutar el bloque el programa continua la ejecución tras ENDSELECT
Ejemplo	

Instrucción	***STOP***
Sintaxis	***STOP***
Descripción	Finaliza la ejecución de un programa hasta que se accione el RESET. El Basic Stamp no queda en el modo de bajo consumo
Ejemplo	

AN1.2 INSTRUCCIONES BUCLES

Instrucción	***DO ... LOOP***
Sintaxis	***DO { WHILE | UNTIL condición(es) }*** ***Instrucción(es)*** ***LOOP { WHILE | UNTIL condición(es) }***
Descripción	Realiza un bucle de repetición que ejecuta todas las instrucciones comprendidas entre DO y LOOP.
Ejemplo	***DO*** ***DEBUG "Hola",CR*** ***PAUSE 1000*** ***LOOP*** Envía al terminal el mensaje "Hola" de forma indefinida cada segundo

Instrucción	***FOR ... NEXT***
Sintaxis	***FOR valor_inicial TO valor_final {STEP valor} ... NEXT***
Descripción	Ejecuta todas las instrucciones comprendidas entre FOR y NEXT tantas veces como sea necesario para que el valor_inicial alcance el valor_final con incrementos/decrementos establecidos por STEP valor
Ejemplo	***FOR numero=1 To 100 STEP 2*** ***DEBUG numero,CR*** ***NEXT*** Visualiza todos los números impares comprendidos entre 1 y 100

Instrucción	***EXIT***
Sintaxis	***EXIT***
Descripción	Provoca el inmediato fin de cualquier bucle realizado mediante FOR .. NEXT, DO ... LOOP
Ejemplo	

AN1.3 INSTRUCCIONES DE ACCESO A EEPROM

Instrucción	***DATA***
Sintaxis	***DATA dato1, dato2, ...***
Descripción	Graba datos en la EEPROM durante el volcado de un programa desde el PC al BasicStamp
Ejemplo	***DATA 72,69,76,79*** ***DATA 101,104,108*** La primera instrucción DATA graba, a partir de la posición 0 de la EEPROM, cada uno de los datos que la acompañan en sucesivas posiciones. La siguiente(s) DATA siguen grabando datos en las posiciones siguientes

Instrucción	***READ***
Sintaxis	***READ posición, Var***
Descripción	Lee, de la EEPROM. el dato presente en la dirección dada y lo almacena en la variable "Var"
Ejemplo	***READ 2,VALOR*** Lee el dato de la posición 2 de la EEPROM y lo almacena en la variable "VALOR"

Instrucción	***WRITE***
Sintaxis	***WRITE posición, dato***
Descripción	Graba un dato sobre la EEPROM, en la dirección indicada
Ejemplo	***WRITE 2,VALOR*** Graba el contenido de la variable "VALOR" en la posición 2 de la EEPROM.

AN1.4 INSTRUCCIONES NUMERICAS

Instrucción	***LOOKDOWN***
Sintaxis	***LOOKDOWN Valor_Ref, [valor0,valor1, ... valorn],variable***
Descripción	Compara un valor de referencia con los valores de una lista. En la variable se almacena el índice de la lista que contiene el valor de referencia.
Ejemplo	***LOOKDOWN 13,[22,55,65,34,78,13,44,89,09,33],indice*** En la variable "indice" se almacena el valor 5 que se corresponde con la posición en la lista que ocupa el valor 13

Instrucción	***LOOKUP***
Sintaxis	***LOOKUP indice,[valor0,valor1,... valorn],variable***
Descripción	Extrae de una lista de valores el valor indicado por "índice" y lo deposita en la variable
Ejemplo	***LOOKUP 3,[23,45,67,13,55,67,64,77],valor*** En la variable "valor" se almacena el dato 13 que es el dato de la lista apuntado por el índice 3

Instrucción	***RANDOM***
Sintaxis	***RANDOM variable***
Descripción	Genera un número seudo aleatorio entre 0 y 65535 y lo almacena en la variable.
Ejemplo	

AN1.5 INSTRUCCIONES DE E/S DIGITALES

Instrucción	***BUTTON***
Sintaxis	***BUTTON pin,estado,delay,ciclos,var,val_ref,dir***
Descripción	Detecta el accionamiento de una señal de entrada (p.e. un pulsador). Realiza auto repetición y salta a una dirección si el pulsador se encuentra en un estado de referencia. Pin: Hace referencia a la línea que hay que detectar Estado: representa el nivel lógico que se genera cada vez que se acciona el pulsador Delay: Expresa el tiempo que debe permanecer la señal activa antes de la auto repetición Ciclos: Expresa el nº de cliclos entre cada auto repetición. Var: Es una variable empleada por la propia instrucción BUTTON. Val_ref: Expresa el estado en que debe encontrarse el pulsador para realizar el salto. Dir: Indica la dirección de salto
Ejemplo	

Instrucción	***COUNT***
Sintaxis	***COUNT pin, periodo, var***
Descripción	Cuenta el nº de transiciones (0-1-0 o 1-0-1) que se producen en el pin de entrada indicado, durante un periodo de tiempo determinado.
Ejemplo	***COUNT 0,1000,numero*** En la variable "numero" se almacena el número de pulsos que se aplican por la entrada 0 durante 1 seg. (1000 mS)

Instrucción	***HIGH***
Sintaxis	***HIGH pin***
Descripción	Pone a nivel lógico "1" el pin de salida indicado
Ejemplo	***HIGH 3*** Pone a "1" la salida 3

Instrucción	***INPUT***
Sintaxis	***INPIT pin***
Descripción	Configura como entrada al pin indicado
Ejemplo	***INPUT 7*** La línea 7 queda configurada como entrada

Instrucción	***LOW***
Sintaxis	***LOW pin***
Descripción	Pone a nivel lógico "0" el pin de salida indicado
Ejemplo	***LOW 3*** Pone a "0" la línea 3 de salida

Instrucción	***OUTPUT***
Sintaxis	***OUTPUT pin***
Descripción	Configura como salida la línea indicada
Ejemplo	***OUTPUT 7*** La línea 7 queda configurada como salida

Instrucción	***PULSIN***
Sintaxis	***PULSIN pin, estado, var***
Descripción	Mide la duración de un estado lógico presente en un determinado pin y la almacena en una variable.
Ejemplo	***PULSIN 7,1,duración*** En la variable duración se almacena el tiempo en que la señal presente en el pin 7 está a nivel "1".

Instrucción	***PULSOUT***
Sintaxis	***PULSOUT pin,periodo***
Descripción	Se genera sobre el pin indicado, un pulso con un periodo determinado
Ejemplo	***PULSOU 3,100*** Se genera un pulso de 100 x 2 μS de duración sobre el pin 3.

Instrucción	***REVERSE***
Sintaxis	***REVERSE pin***
Descripción	Invierte la configuración de un pin de E/S. Si es de entrada lo configura como salida y viceversa.
Ejemplo	***OUTPUT 7*** ***REVERSE 7*** Finalmente la línea 7 queda configurada como entrada

Instrucción	***TOGGLE***
Sintaxis	***TOGGLE pin***
Descripción	Invierte el estado lógico de un pin de salida
Ejemplo	***LOW 7*** ***TOGGLE 7*** El pin o línea de salida nº 7 pasa a nivel "1"

Instrucción	***XOUT***
Sintaxis	***XOUTMpin, Zpin, codigo, comando,ciclos***
Descripción	Envía un comando mediante el protocolo X10 a través de una línea eléctrica. Mpin: Representa la línea de salida por donde se envía la señal modulada. Zpin: Representa la línea de entrada para detectar el cruce por cero Codigo: Representa el código del dispositivo X10 seleccionado Comando: Representa el comando a transmitir Ciclos: es opcional y represente el nº de veces que se debe transmitir el comando
Ejemplo	

AN1.6 INSTRUCCIONES PARA E/S SERIE ASINCRONAS

Instrucción	***SERIN***
Sintaxis	***SERIN Rpin, Fpin, Baudios, Plabel, timeout, Tlabel, datos***
Descripción	Recibe datos en serie asíncronos Rpin: Representa el pin de entrada por donde se reciben los datos Fpin: Representa un pin de salida que indica estado de la recepción en ON Baudios: Representa la velocidad de recepción Plabel: Es opcional e indica la dirección de salto en caso de error de paridad Timeout: Representa el tiempo de espera en la recepción Tlabel: Es opcional y representa la dirección de salto en caso de que se exceda el Timeout Variable: Es una lista de variables donde se almacenan los datos recibidos
Ejemplo	

Instrucción	***SEROUT***
Sintaxis	***SEROUT Tpin, Fpin, baudios, lap, timeout, Tlabel, Datos***
Descripción	Transmite un dato serie asíncrono Tpin: Representa el pin de salida por donde se transmite Fpin: Representa un pin de entrada que indica permiso para la transmisión Baudios: Representa la velocidad de transmisión Lap: Representa el lapsus de tiempo entre la transmisión de un dato y el siguiente Timeout: Tiempo que debe esperarse hasta que Fpin indique permiso de transmisión Tlabel: Dirección de salto en caso de que se sobrepase el Timeout Datos: Lista de variables o constantes con los datos a transmitir
Ejemplo	

AN1.7 INSTRUCCIONES DE E/S SERIE SINCRONA

Instrucción	***SHIFTIN***
Sintaxis	***SHIFTIN Dpin, Cpin, Modo, Var \bits, var\bits ....***
Descripción	Lee datos serie síncronos desde un dispositivo serie Dpin: Determina la línea de entrada de datos Cpin: Determina la línea de salida de reloj Modo: Determina el modo de la recepción en serie Var: Representa la variable donde se almacena el dato recibido Bits: Representa el nº de bits de que debe constar el dato a recibir
Ejemplo	

Instrucción	***SHIFTOUT***
Sintaxis	***SHIFOUT Dpin, Cpin, Mode, Var\bits, Var\bits ....***
Descripción	Transmite datos en serie de forma síncrona hacia un dispositivo serie Dpin: Determina la línea de salida de datos Cpin: Determina la salida de reloj Modo: Determina el modo de transmisión serie Var: Contiene el dato a transmitir Bits: Representa el nº de bits de que consta el dato a transmitir
Ejemplo	

AN1.8 INSTRUCCIONES DE E/S ANALOGICAS

Instrucción	***PWM***
Sintaxis	***PWM pin, duty, ciclos***
Descripción	Convierte un valor digital en una salida analógica mediante modulación de anchura de pulsos. Pin: Define la línea de salida por donde se obtiene la señal PWM Duty: Expresa el valor analógico de salida (entre 0 y 5V) Ciclos: Representa la duración de la señal PWM de salida
Ejemplo	

Instrucción	***RCTIME***
Sintaxis	***RCTIME pin, estado, var***
Descripción	Mide el tiempo en que un determinado pin está en un determinado estado o nivel. Pin: Representa a la señal de entrada a medir Estado: Representa el nivel lógico "0" o "1" presente en el pin, que se quiere medir. Var: Variable donde se almacena el tiempo medido
Ejemplo	***RCTIME 1,1,tiempo*** Mide el tiempo en que la línea de entrada nº 1 se encuentra a nivel "1". El resultado se almacena en la variable "tiempo"

AN1.9 INSTRUCCIONES PARA GENERAR SONIDOS O TONOS

Instrucción	***DTMFOUT***
Sintaxis	***DTMFOUT pin, Ontime, OFFtime, Tono, Tono,....***
Descripción	Genera tonos multifrecuencia (teclado telefónico) Pin: Representa la línea de salida Ontime: Representa la duración del tono OFFtime: Representa el tiempo o pausa entre un tono y el siguiente Tono: Expresa el tono a generar
Ejemplo	***DTMFOUT 3, 50, 10, [9,4,4,2,3,0,6,5,1]*** Por la línea 3 se obtienen los tonos correspondientes al nº 944230551 (Tfno. de MSE). Cada tono tiene una duración de 50ms y el intervalo entre un tono y el siguiente es de 10mS.

Instrucción	***FREQOUT***
Sintaxis	***FREQOUT pin, periodo, frec1, frec2***
Descripción	Genera una o dos señales senoidales durante un periodo de tiempo determinado Pin: Representa la línea de salida de la señal Periodo: Representa la duración de la señal de salida Frec1: Representa la frecuencia de la 1ª señal Frec2: Representa la frecuencia de la 2ª señal
Ejemplo	

AN1.10 INSTRUCCIONES DE CONTROL DE TIEMPO

Instrucción	***PAUSE***
Sintaxis	***PAUSE periodo***
Descripción	Produce una pausa o temporización en la ejecución de un programa Periodo: Representa el tiempo a esperar
Ejemplo	***PAUSE 1000*** Temporiza un segundo (1000 ms)

AN1.11 INSTRUCCIONES PARA EL CONTROL ALIMENTACIÓN

Instrucción	***END***
Sintaxis	***END***
Descripción	Finaliza la ejecución de un programa dejando el Basic Stamp en el modo de bajo consumo indefinidamente.
Ejemplo	

Instrucción	***NAP***
Sintaxis	***NAP Periodo***
Descripción	Detiene la ejecución de un programa dejando el Basic Stamp en el modo de bajo consumo durante un intervalo de tiempo. Dicho intervalo se calcula: (2^Periodo) + 18 mS. Donde periodo es un valor comprendido entre 0 y 7, por lo que el intervalo mínimo es de 0mS y el máximo 2304 mS (2,3s)
Ejemplo	***NAP 4*** El Basic Stamp queda en el modo de bajo consumo durante (2^4)*18 = 288 mS

Instrucción	***SLEEP***
Sintaxis	***SLEEP Periodo***
Descripción	Detiene la ejecución de un programa dejando al Basic Stamp en el modo de bajo consumo durante un intervalo de tiempo. Dicho intervalo se calcula: (periodo * 2.3 seg)/2. El periodo puede estar comprendido entre 1 y 65535, por lo que el intervalo mínimo es de 2.3 seg y el máximo de aproximadamente unas 20h.
Ejemplo	

AN1.12 INSTRUCCIONES DE DEPURACIÓN DE PROGRAMA

Instrucción	***DEBUG***
Sintaxis	***DEBUG datos,datos***
Descripción	Envía y visualiza información al PC, sobre la ventana del Basic Stamp Terminal. Este comando se puede emplear para visualizar texto o números con diferentes formatos, de modo que sirve para analizar el desarrollo de la ejecución de un programa.
Ejemplo	

Instrucción	***DEBUGIN***
Sintaxis	***DEBUGIN dato***
Descripción	Recibe información del usuario a través del PC, mediante la ventana del Basic Stamp Terminal correspondiente.
Ejemplo	

Anexo 2

Otros Microbots: El PICBOT-2 y PICBOT-3

AN2.1. EL PICBOT-2

El microbot PICBOT-2 diseñado y comercializado por Ingeniería de Microsistemas Programados S.L. (en lo sucesivo MSE) consiste en un robot móvil de propósito general compacto y de reducidas dimensiones. Supone un paso adelante en cuanto a los conceptos de programación. Su tarjeta de control emplea un PIC16F84 que se programa directamente desde su propio lenguaje natural: El Ensamblador. La principal diferencia con el Home Boe-Bot es el lenguaje de programación, que en el caso del PICBOT-2 al ser ensamblador, exige conocer en profundidad la arquitectura del microcontrolador.

Como cualquier otro microbot, consta de cuatro partes fundamentales, de las cuales MSE proporciona una versión funcional, didáctica y económica de las mismas:

- La estructura
- Los elementos motrices
- Los sensores que informan del entorno
- La unidad de control inteligente

Posteriormente será el propio usuario quién determine y diseñe el tipo de estructura óptima para su propia aplicación. El usuario será también quién elija el tipo de mecanismos o motores a emplear, en base a la velocidad, consumo, potencia requerida, etc. Igualmente y, dada la enorme cantidad de tipos de sensores existentes, será también el usuario quién determine el tipo de los mismos en función del entorno en el que su propio microbot deberá desenvolverse.

Finalmente, la unidad de control será la encargada de ejecutar el programa de trabajo determinado también por ese usuario. Precisamente esta unidad es la que proporcionará un cierto nivel de inteligencia al microbot. Ella será la encargada de evaluar el entorno, comunicarse, controlar los movimientos, en base a una serie de algoritmos programados previamente.

La microbótica es una ciencia en pleno desarrollo. No hay nada determinado, no hay límites. Todo está por hacer. Es la imaginación de cada cual quién determina ese desarrollo y las múltiples aplicaciones posibles.

La figura AN2-1 muestra el aspecto que, una vez montado, tendrá el microbot **PICBOT-2.**

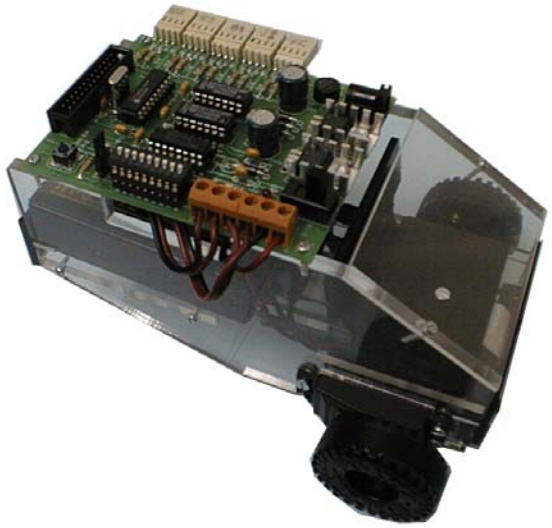


Figura AN2-1. *El PICBOT-2*

En la foto de la figura AN2-2 se muestra el conjunto de elementos y componentes necesarios para el completo montaje y puesta en marcha del PICBOT-2.

Figura AN2-2. *Composición del PICBOT-2*

Las características más relevantes del microbot PIC-BOT que propone Ingeniería de Microsistemas Programados S.L. (MSE), son las siguientes:

- Estructura realizada en metacrilato. El kit se compone de diferentes piezas y accesorios que se fijan entre sí mediante tornillos. Se consigue una estructura rígida y de fácil montaje, que puede servir como modelo para otras nuevas que desarrolle el propio usuario.

- Los elementos motrices están formados por dos motores de corriente continua. Se trata de los servomotores de la firma FUTABA S3003. Este tipo de motores disponen de un grupo reductor que garantiza un buen par de fuerza así como una muy reducida inercia en los arranques y paradas. Igualmente disponen de un buen sistema de anclaje que permite acomodarlos fácilmente a cualquier tipo de estructura.

- Se proporcionan dos tipos de sensores básicos: infrarrojos y mecánicos. Gracias a ellos se puede determinar entornos sencillos en los que se mueve el microbot. Posteriormente el usuario podrá adaptar otro tipo de sensores que permitan trabajar en entornos más complejos.

- La unidad de control está formada por la tarjeta **MSx84** de **MSE**. Su unidad de proceso está formada por el popular microcontrolador PIC16F84. Este permite ser reutilizado pudiéndose re grabar en numerosas ocasiones. La tarjeta MSx84 permite conectar directamente hasta 5 sensores de entrada y gobernar dos motores DC o uno paso a paso PAP.

Lista de materiales

El lote completo para la construcción del **PICBOT-2** de **MSE** está compuesto de los siguientes materiales y componentes:

CANTIDAD	DESCRIPCION
1	Tarjeta de control MSx84 gobernada por el microcontrolador PIC 16x84
1	Juego de piezas de metacrilato con las que se confecciona la estructura
2	Motores FUTABA S3003 como elementos motrices
3	Sensores infrarrojos tipo CNY70
2	Sensores infrarrojos tipo H21A1
2	Sensores mecánicos de final de carrera tipo Bumper
5	Cables para la conexión de los sensores con la tarjeta MSx84
1	Portapilas para batería de 9V
1	Portapilas para 4 baterías de 1.5V

Relación de herramientas

Para la construcción de todo el conjunto **PICBOT-2** es necesario un conjunto de herramientas y accesorios adicionales que no están incluidas en el kit, pero que son de fácil localización y están presentes en el taller o laboratorio de cualquier profesional y/o aficionado.

- Soldador/Desoldador de punta fina de 30 W de potencia máxima.
- Estaño al 60% de 1mm de grosor máximo.
- Alicates de corte.
- Destornillador fino de punta de estrella.
- Destornillador fino de punta plana.
- Tijeras o herramienta pela cables.
- Adhesivo universal instantáneo tipo LOCTITE o similar.
- Cinta adhesiva o cinta aislante.

También es necesario disponer de algún circuito de grabación con su correspondiente software para poder grabar el PIC con el programa de aplicación diseñado por el usuario. En Ingeniería de Microsistemas Programados S.L. disponemos del grabador Micro'PIC Programmer (ref. MPICPRO) y/o del entrenador/grabador Micro'PIC Trainer (ref.MPICTRAM). Ambos equipos vienen acompañados de su correspondiente software e instrucciones de grabación. En las figuras AN2-3 y AN2-4 se muestran las fotografías de los mismos.

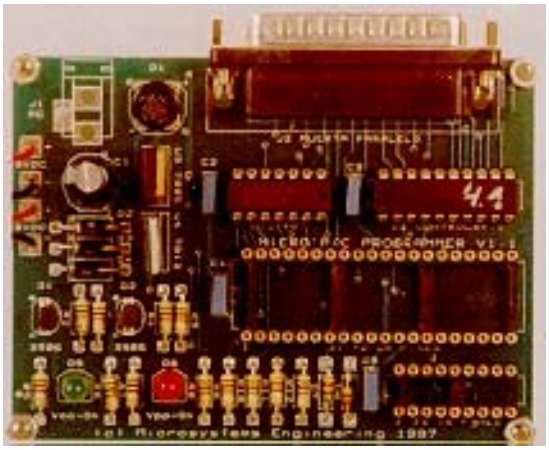


Figura AN2-3. *El grabador Micro'PIC Programmer (MPICPRO)*

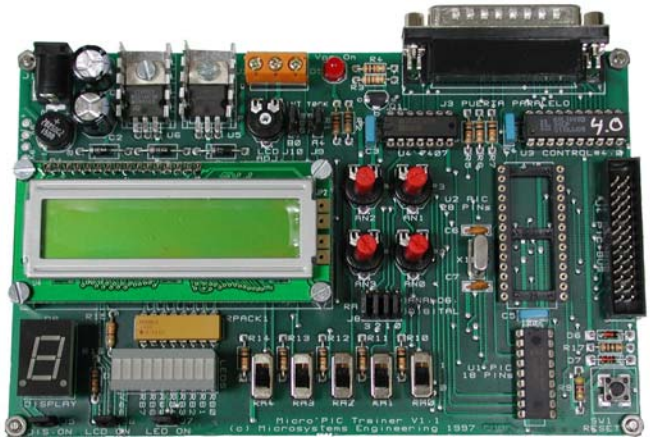


Figura AN2-4. *El entrenador/grabador Micro'PIC Trainer (MPICTRAM)*

El PICBOT-2 viene acompañado de un completo manual en castellano que se puede bajar desde nuestra página www.microcontroladores.com, en la sección de Downloads. En el Tema 1 del mismo se irá explicando paso a paso, mediante dibujos y fotografías, la forma de construir la estructura del **PICBOT-2** así como la fijación de los motores y sensores.

El tema 2 del manual está dedicado a explicar el funcionamiento y aplicaciones de la tarjeta de control **MSx84**, que se aprecias en la figura AN2-5.. En dicho tema se darán ideas de cómo controlar y aplicar diferentes tipos de sensores así como el gobierno de motores tanto DC como paso a paso (PAP).

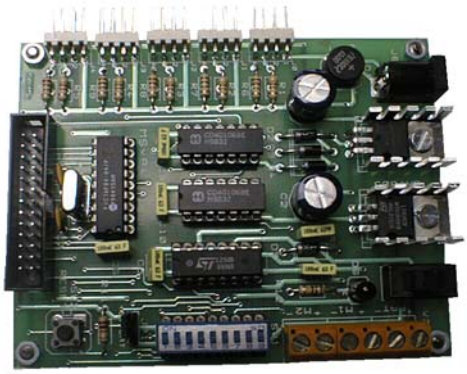


Figura AN2-5. *La tarjeta de control MSX-84*

Finalmente, el tema 3 está dedicado a la presentación de diversas aplicaciones desarrolladas para el **PICBOT-2**. Este debiera ser un tema en continua expansión. Ingeniería de Microsistemas Programados S.L. pretende incluir y publicar periódicamente, en sus notas de aplicación, todas las sugerencias, modificaciones, aplicaciones e ideas originales que los usuarios nos hagan llegar. De esta forma será posible dar a la microbótica un carácter abierto y una continuidad en la que los límites sea la imaginación de los propios usuarios.

AN2.2. EL PICBOT-3

Se trata de la última generación de Microbots desarrollado y comercializado por *Ingeniería de Microsistemas Programados S.L.* El PICBOT-3 es un robot basado en los potentes microcontroladores de la familia PIC16F87X y de diseño totalmente modular.

La fotografía de la figura AN2-6 muestra el aspecto del robot en su versión básica. A partir de ella, el usuario puede ir añadiendo diferentes sensores, actuadores y complementos, para adaptar el robot a diferentes entornos de trabajo.

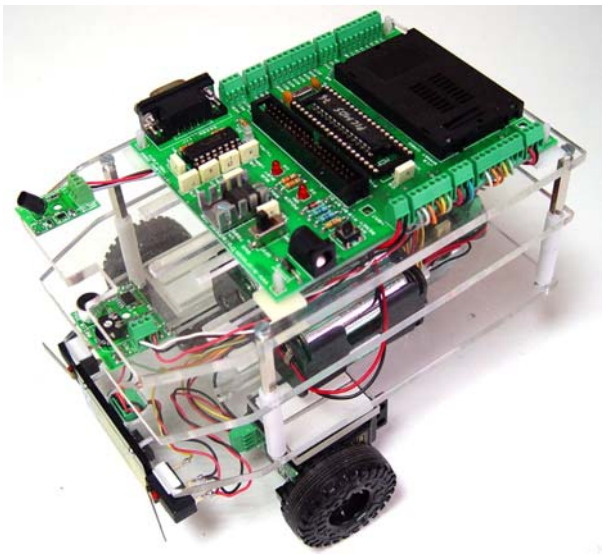


Figura AN2-6. *El PICBOT-3*

En el manual que lo acompaña y, que se puede bajar desde www.microcontroladores.com , se pretende explicar los diferentes elementos de que consta el robot: estructura, tracción, electrónica, etc. así como el montaje paso a paso del mismo. Somos conscientes que, finalmente, será el usuario quién terminará adaptando el PICBOT-3 a sus gustos y necesidades. Sabemos de sobra que en el mundo de la microbótica nada está escrito. Aunque nosotros propongamos un determinado modelo estamos seguros de que el usuario terminará modificándolo. Desde aquí animamos a que así sea.

En dicho manual también se irán presentando las diferentes opciones de ampliación junto con una variada selección de ejemplos de carácter didáctico, que permiten el empleo de los numerosos recursos y posibilidades del PICBOT-3.

La estructura del PICBOT-3

Se basa fundamentalmente en 3 bases de metacrilato que permiten realizar una estructura tipo torre en la que se alojarán los diferentes sensores y actuadores que forman el robot. En *Ingeniería de Microsistemas Programados S.L.* pensamos que, si bien la estructura tiene una cierta importancia, no es menos cierto que el tipo de sensores, motores y actuadores empleados, son fundamentales, al igual que el tipo de tarjeta de control y el correspondiente procesador.

Es por ello que la estructura que proponemos es una estructura sencilla y económica aunque no por ello deja de ser versátil y flexible.

En la figura AN2-7 se muestran los materiales empleados para la construcción de la estructura del PICBOT-3.

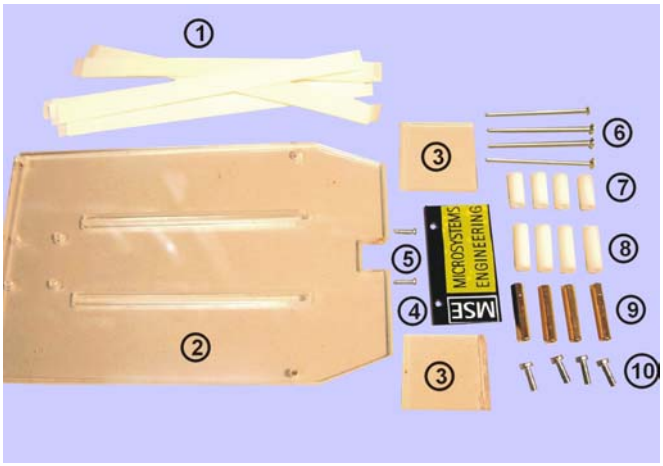


Figura AN2-7. *Componentes que conforman la estructura del PICBOT-3*

1.- **Cintas adhesivas.** Lo son por ambas caras y se emplearán para la sujeción de prácticamente todos los elementos de que consta PICBOT-3. Proporcionan una adherencia muy consistente y, al poderse despegar, ofrecen la ventaja de poder realizar las modificaciones que sean necesarias.

2.- **Bases de metacrilato.** Hay tres bases iguales que formarán una estructura tipo torre de tres alturas, planta baja, planta intermedia y planta superior. El metacrilato es una material bastante ligero, consistente y fácil de mecanizar. Al no ser poroso admite muy bien todo tipo de adhesivos.

3.- **Soportes de metacrilato.** Hay dos y se emplearán para la sujeción de los motores, de forma que estos queden a cierta altura respecto al plano.

4.- **Panel frontal.** Se trata de una placa de metacrilato serigrafiada. Se colocará en la parte delantera del robot.

5.- **Tornillos.** Son dos tornillos rosca chapa para la sujeción del panel frontal sobre la base de metacrilato de la de la planta baja.

6.- **Tornillos.** Cuatro tornillos de 50 mm y métrica 3 que servirán para unir la planta baja con la planta intermedia.

7.- **Separadores.** Cuatro separadores de plástico de 15mm empleados para separar la planta baja de la intermedia.

8.- **Separadores.** Cuatro separadores de plástico de 20mm empleados para separar la planta baja de la intermedia.

9.- **Separadores.** Cuatro separadores metálicos de 25mm que se emplean para separar la planta intermedia de la planta superior.

10.- **Tornillos.** Cuatro tornillos de 10mm métrica 3. Se emplean para unir la planta intermedia con la planta superior de la estructura.

El sistema de tracción

Se encarga de producir los movimientos necesarios que permitan maniobrar al microbot PICBOT-3. Consta de los elementos mostrados en la fotografía de la figura AN2-8.

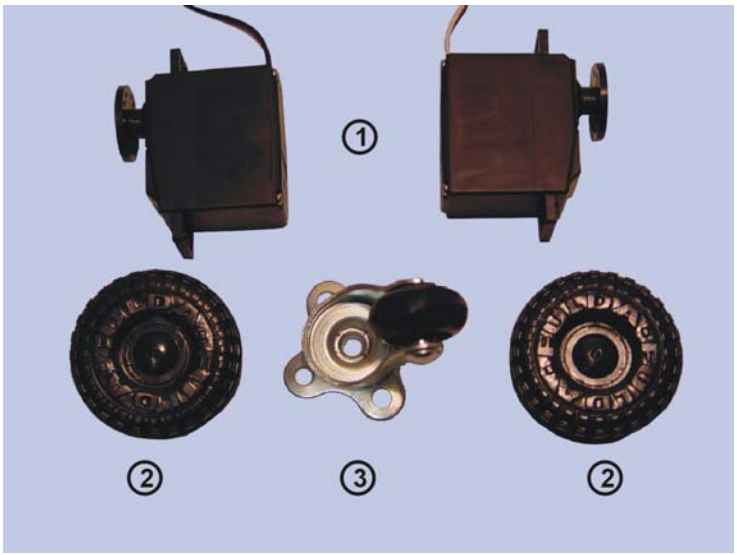


Figura AN2-8. *Sistema de tracción del PICBOT-3*

1.- Motores. Se emplean dos servomotores FUTABA S3003 previamente modificados. Estos motores están diseñados, fundamentalmente, para tareas de servo control en modelos teledirigidos (barcos, aviones, etc.). Sin embargo, los vamos a modificar retirando la electrónica de control que poseen para conseguir así que tengan un giro libre. Estos motores no son precisamente económicos y además vamos a anular algunas de sus prestaciones, sin embargo en *Ingeniería de Microsistemas Programados S.L.* hemos optado por su empleo ya que ofrecen las siguientes ventajas:

- Fácil localización en el mercado, principalmente en el sector de radio control y modelismo.
- En la misma carcasa se incluye un grupo reductor que proporciona un muy buen par de fuerza y una muy buena estabilidad en la velocidad de giro.
- Baja inercia en los instantes de arranque y parada. La respuesta es casi inmediata a las señales de ON/OFF generadas por el sistema de control. De esta forma se evitan desplazamientos no deseados, se consigue una rápida respuesta del robot así como una buena precisión en el movimiento.
- Tensión de alimentación flexible y de bajo consumo. Las pruebas realizadas en nuestros laboratorios han dado unos resultados satisfactorios con tensiones de entre 5Vdc y 12Vdc.
- El reducido tamaño de la carcasa (40.4 x 19.8 x 36mm) y lo compacta que es, permite que el motor se pueda ubicar de forma fácil en cualquier tipo de estructura.
- Incluye un eje de aplicación dentado y un conjunto de accesorios que permiten múltiples posibilidades para acoplarse con diferentes tipos de ruedas o dispositivos para la transmisión del movimiento.

- El reducido peso de todo el conjunto (37 g) contrasta con el apreciable par de fuerza (3.2 Kg/cm).

2.- **Ruedas.** Consisten en dos ruedas de PVC de 44 mm de diámetro. El usuario puede optar por otro tipo de ruedas que tengan una mejor tracción en función del terreno donde deba desenvolverse el robot.

3.- **Rueda trasera.** Es una rueda con giro libre que se colocará como rueda trasera del robot PICBOT-3.

Dispositivos electrónicos

Es un conjunto de circuitos, sensores, actuadores, etc. que posibilitan que el PICBOT-3 tenga conocimiento del entorno que le rodea, así como de la tarjeta de control que realiza todas las tareas de gobierno. En la figura AN2-9 se muestran los elementos de que consta el PICBOT-3 en su versión básica. *Ingeniería de Microsistemas Programados S.L.* pone a disposición de todos sus clientes y usuarios un variado conjunto de dispositivos opcionales que permitirán un aumento de las posibilidades y prestaciones del robot.

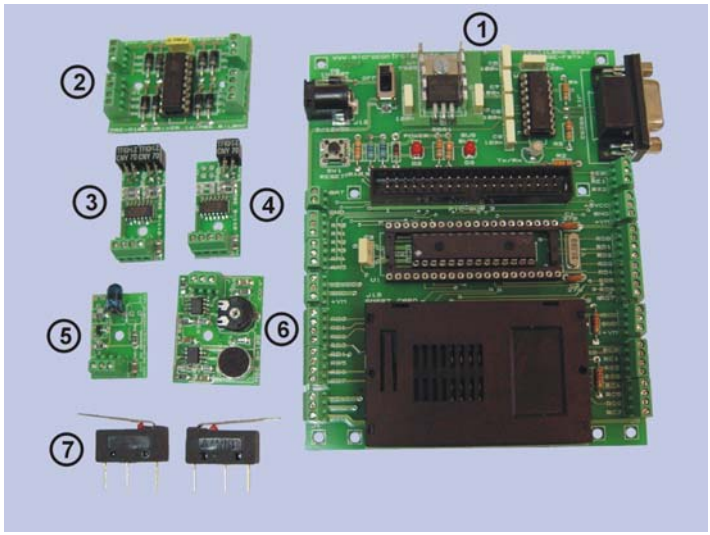


Figura AN2-9. *Dispositivos electrónicos*

1.- **Tarjeta de control.** Se trata de la tarjeta de control MSEF87X diseñada y comercializada por *Ingeniería de Microsistemas Programados S.L.* De serie incorpora un microcontrolador PIC16F876 dotado del programa monitor PICMOS'76. Opcionalmente se puede incorporar el PIC16F877 con el programa monitor PICMOS'77. El programa monitor PICMOS'7X permite la comunicación con el software Real-Pic para PC, que facilita la edición, ensamblado y grabación de los programas de aplicación del usuario. La velocidad de trabajo en ambos casos es de 20MHz lo que significa ejecutar una instrucción cada 200nS. Todas las líneas de E/S están accesibles mediante un conjunto de bornas que facilitan la conexión con los múltiples sensores y periféricos disponibles. La tarjeta incorpora el sistema de estabilización de tensión, un canal serie para comunicación con el PC y un conector Smart-Card que permite el empleo de tarjetas de memoria Memory Card para salvar y recuperar los programas de aplicación de forma autónoma.

2.- **Driver MSE-A100.** Se trata de un driver amplificador de 4 canales con los que se obtiene una señal de salida de hasta 35Vdc con una corriente de 1 A por cada canal. Se emplea para el gobierno de los dos motores de que consta PICBOT-3 permitiendo el control ON/OFF de cada uno de ellos así como el sentido de giro.

3.- **Sensor MSE-S110.2.** Contiene dos transductores IR de reflexión que permitirá controlar la trayectoria del robot siguiendo una línea negra/blanca trazada en el suelo.

4.- **Sensor MSE-S110.1** Contiene un transductor IR de reflexión que permitirá controlar el desplazamiento que produce una de las ruedas del robot a modo de encoder.

5.- **Sensor MSE-S130.** Se trata de un sensor capaz de proporcionar información sobre la luz ambiente que incide sobre él. De esta forma el PICBOT-3 puede reaccionar ante los cambios de luz que se produzcan en su entorno o área de trabajo.

6.- **Sensor MSE-S100.** Es un sensor que detecta cualquier señal sonora que se produzca dentro de su radio de acción. El umbral de sonido se ajusta mediante un potenciómetro y permitirá que el PICBOT-3 pueda reaccionar ante la presencia de cualquier tipo de ruido.

7.- **Bumpers.** Se suministran dos de estos dispositivos electromecánicos. Permiten detectar la colisión del PICBOT-3 con cualquier objeto que se halle en su trayectoria.

Varios

Finalmente, en la figura AN2-10 se muestran una serie de accesorios adicionales que componen la versión básica del PICBOT-3, a los que habría que añadir el manual de usuario así como un disco con el software y ejemplos de demostración (estos ejemplos también se pueden bajar desde www.microcontroladores.com.

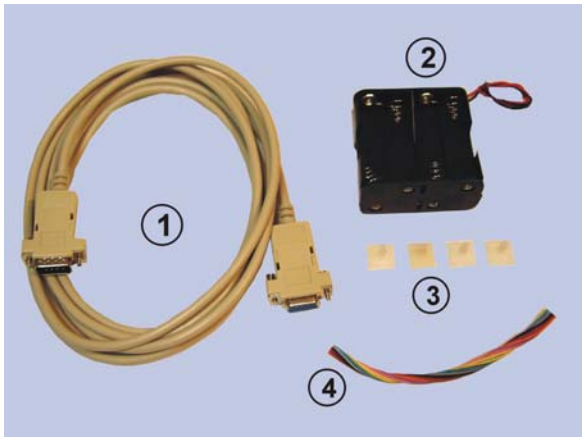


Figura AN2-10. *Accesorios*

1.- **Cable serie.** Es el cable para la comunicación serie entre la tarjeta de control MSEF87X y el PC.

2.- **Portapilas.** Admite 8 pilas (no incluidas) del tipo RA6/AAA para la alimentación general del sistema.

3.- **Separadores.** Se trata de 4 separadores adhesivos que se emplearán para fijar la tarjeta de control MSEF87X sobre la planta superior de la estructura del PICBOT-3.

4.- **Manguera.** Consiste en una manguera de cables de 3m y 12 hilos de diferentes colores. Se empleará para conectar entre sí todos los sensores, motores y tarjeta de control de que consta el PICBOT-3

Sistema de alimentación

La alimentación del PICBOT-3 es flexible y fácil de realizar. En *Ingeniería de Microsistemas Programados S.L.* hemos probado diferentes alternativas con las que se han obtenido resultados satisfactorios. Las explicamos a continuación.

Fuentes de Alimentación (F.A.)

Se trata de un forma sencilla de alimentar al PICBOT-3 mediante el empleo de las clásicas fuentes de alimentación, como la mostrada en la figura AN2-11, fáciles de conseguir y económicas.

Figura AN2-11. *Fuente de alimentación*

Se recomienda el empleo de una fuente de alimentación con una tensión de salida de 12 a 15Vdc y una corriente de 500 a 1000mA. Si es estabilizada mejor. La conexión con la tarjeta de control MSEF87X se realiza mediante la correspondiente clavija con el positivo al centro de la misma.

El empleo de este tipo de alimentación resulta muy cómodo y económico cuando el usuario está en fase de desarrollo y prueba de sus programas de aplicación. No gasta baterías ni pilas. Sin embargo tiene el inconveniente de que la F.A debe estar conectada a la red eléctrica con lo que el robot pierde toda autonomía.

Baterías alcalinas recargables

El empleo de baterías recargables del tipo RA6/AAA como las mostradas en la figura AN2-12 es una buena solución, que permite que el robot tenga total autonomía a un precio asequible y durante un tiempo razonable. En los laboratorios de *Ingeniería de Microsistemas Programados S.L.* hemos probado y obtenido excelentes resultados mediante el empleo de las baterías FREAK de la firma METROTECNIA S.L. Entre las características a destacar cabe citar que, además de ser recargables, cada batería ofrece una tensión de salida de 1,5Vdc en lugar de los típicos 1,2 Vdc, y una corriente de 700mA/Hora.

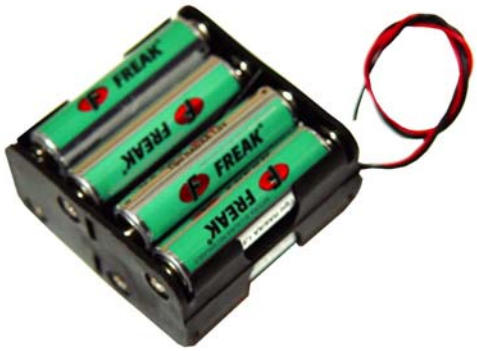
Figura AN2-12. *Baterías alcalinas recargables del tipo RA6/AAA*

El mismo fabricante dispone de un cargador como el mostrado en la figura AN2-13 que permite la carga de hasta 4 baterías del tipo RA6/AAA o del tipo RA3/AA (nunca mezcladas).

Figura AN2-13. *Cargador de baterías alcalinas*

Tal y como se muestra en la figura AN2-14, con un conjunto de 8 baterías de este tipo, alojadas en el correspondiente porta pilas (incluido), se obtiene una tensión total de 12Vdc. Con esta tensión se alimenta a todos los elementos de que consta el PICBOT-3, dotándole de una autonomía de unos 45 minutos aproximadamente.

Figura AN2-14. *Alimentación del PICBOT-3 mediante un pack de 8 baterías*



Baterías recargables de plomo ácido

Es otra posible solución a la alimentación del PICBOT-3, con la que hemos conseguido también excelentes resultados. Se trata del empleo de una única batería recargable de plomo ácido como la mostrada en la figura AN2-15, para la alimentación de todos los elementos del PICBOT-3.

Figura AN2-15. *Las baterías de plomo ácido*

Aunque tienen un peso un tanto elevado proporcionan una tensión de salida de 12Vdc y 800mA/hota, dotando al PICBOT-3 de una excelente autonomía. También está disponible en el mercado, el correspondiente cargador como el mostrado en la figura AN2-16.

Figura AN2-16. *Cargador para las baterías de plomo ácido*

Como solución final cabe citar que la opción más óptima (pero la más cara) consiste en emplear dos alimentaciones diferentes. Mediante un pack de 8 baterías alcalinas como el de la figura AN2-14 se alimenta a toda la electrónica (tarjetas de control y sensores). Mediante otra batería de plomo ácido se alimentan única y exclusivamente los motores. La estructura del PICBOT-3 tiene el espacio suficiente para alojar a ambas. Lógicamente se consigue así una mayor autonomía y además, como la alimentación de los motores está separada de la alimentación de la electrónica, se evitan problemas de ruidos e interferencias. Efectivamente, cada vez que los motores arrancan o cambian de sentido de giro, se producen unos picos de consumo elevados que, dependiendo del estado de carga de las baterías pueden provocar bajadas en la tensión de alimentación. Esto a su vez puede producir re inicios no deseados tanto en la tarjeta de control como en algunos de los sensores.

Anexo 3

Sensores y actuadores para microbots: Módulos Conectar & Funcionar

AN3-1 INTRODUCCION

Ingeniería de Microsistemas Programados S.L. viene, desde hace tiempo, desarrollando y comercializando una serie de sensores y actuadores de propósito general. Estos dispositivos se caracterizan por ser totalmente autónomos y funcionales por sí mismos. Se pueden emplear en cualquier aplicación de carácter comercial como pueden ser sistemas de control de procesos, alarmas, control de acceso, automatismos y, por supuesto, en microbótica.

Efectivamente, se tratan de dispositivos de bajo coste y fácil instalación que puede emplearse tanto en el Home Boe-Bot como en cualquier otro tipo de robot, haciendo que dicho robot tenga una mayor percepción del entorno que le rodea.

AN3-2 EL SENSOR DE SONIDO MSE-S100

Se trata de un sensor activado por sonido, ver la figura AN3-1. Un micrófono recoge la señal de sonido o ruido ambiente. Esta señal es amplificada y, si se alcanza un determinado nivel o umbral, se produce un pulso lógico de disparo de unos 100 mS de duración y activo por flanco ascendente.

Mediante un potenciómetro de ajuste es posible regular el nivel sonoro al que se desea se produzca la señal de disparo en la salida. De esta forma se puede ajustar la sensibilidad del circuito.

El circuito en reposo (ausencia de ruido/sonido) mantiene la señal de salida a nivel lógico "0" permanente.

El circuito dispone de un orificio que permite una flexible instalación y sujeción del mismo sobre cualquier tipo de estructura.

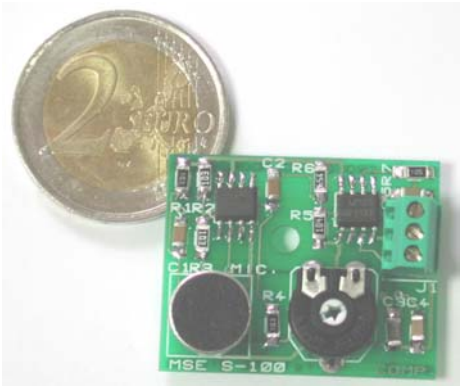


Figura AN3-1. *El sensor de sonido MSE-S100*

Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones del circuito	32 x 26	mm
Tensión de alimentación	5	Vcc
Consumo aprox. en reposo (sin sonido)	2.6	mA
Consumo aprox. en activación (con sonido)	1.7	mA
Tensión de salida en reposo ("0")	0	Vcc
Tensión de salida en activación ("1")	> 3.5	Vcc
Duración aprox. del pulso en activación	100	mS
Flanco de salida activo	ascendente	

Conexionado

Se realiza mediante una borna de 3 contactos con paso 2.54, tal y como se muestra en la figura AN3-2.

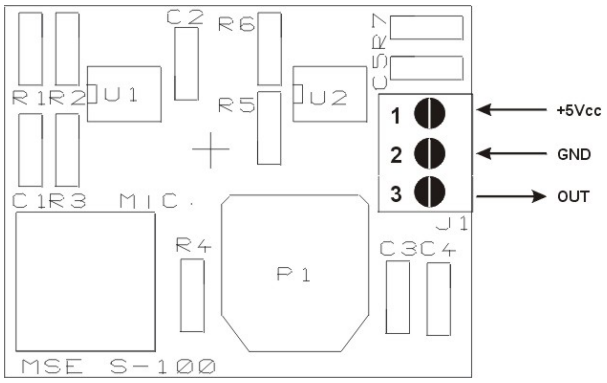


Figura AN3-2. *Conexionado del MSE-S100*

Ajustes

El ajuste del sensor de sonido MSE-S100 permite regular el nivel sonoro necesario para disparar la señal de salida. Se realiza mediante el potenciómetro ajustable P1.

La alimentación de +5Vcc se aplica a través de las conexiones 1 y 2 de la borna. Con un voltímetro se mide la señal de salida entre las conexiones 2 y 3. Se genera un sonido o ruido (p.e. con un zumbador, dando palmadas, etc..) y se mide el momento en que la señal de salida sube a nivel lógico "1". Mediante el potenciómetro P1 se ajusta el nivel de ruido al umbral de disparo deseado.

En las posteriores aplicaciones que se realicen con el sensor, es importante que, tras conectar la alimentación, se espere un mínimo de 100mS antes de procesar la señal de salida. Ello es debido a que el circuito necesita de ese tiempo para su propia estabilización y durante el cual se pueden generar falsas señales de disparo.

Aplicaciones

El sensor de sonido MSE-S100 es capaz de generar una señal lógica en función del sonido o ruido ambiente. Las aplicaciones son numerosas:

- Automatismos digitales en los que se procesan señales que provienen de diferentes sensores o transductores.
- Alarmas activadas por ruidos o sonidos provocados por intrusos.
- Microbótica en donde es necesario que se actúe en función del entorno que rodea al robot

AN3-3 EL SENSOR DE REFLEXION MSE-S110

El sensor de reflexión MSE-S110 está basado en los populares dispositivos CNY70 y están disponibles en versiones con 1 o 2 de estos dispositivos (MSE-S110.1 o MSE-S110.2). Ver la figura AN3-3. Cada dispositivo dispone de un emisor/receptor de luz IR. Cuando la luz se dispersa o es absorbida por una superficie oscura, la salida del correspondiente dispositivo, tras ser acondicionada, es de nivel "1". Sin embargo, cuando la luz es reflejada por una superficie clara, se genera una señal lógica de nivel "0".

Estas señales lógicas de salida se obtienen a través de las conexiones 3 y 4 de la borna. OUT1 se corresponde con la señal captada por el dispositivo 1 y OUT2 con la del dispositivo 2 (sólo en la versión MSE-S110.2). El circuito dispone de un orificio que permite una flexible instalación y sujeción del mismo sobre cualquier tipo de estructura.

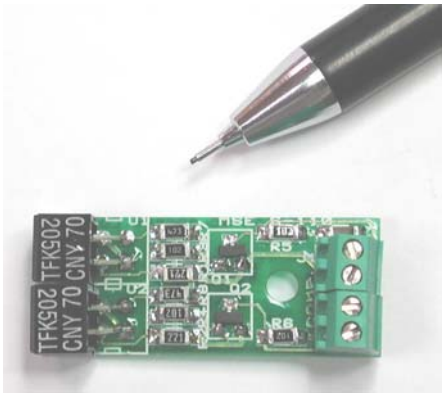


Figura AN3-3. *El sensor de reflexión MSE-S110.2*

Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones aproximadas del circuito	31x15	mm
Tensión de alimentación	5	Vcc
Consumo aprox. en reposo (sin reflexión)	35	mA
Consumo aprox. en activación por cada dispositivo	+ 5	mA
Tensión de salida en reposo ("1")	5	Vcc
Tensión de salida en activación ("0")	< 0.1	Vcc
Distancia aproximada de reflexión	10	mm

Conexionado

Se realiza mediante una borna de 4 contactos con paso 2.54, tal y como se muestra en la figura AN3-4.

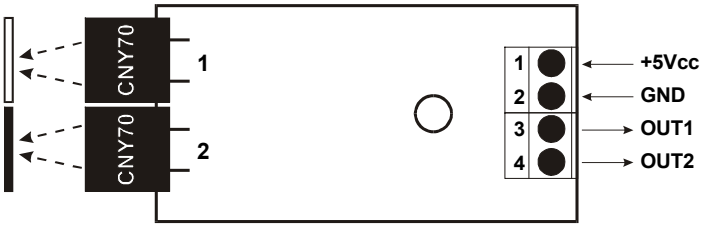


Figura AN3-4. *Conexionado del sensor de reflexión MSE-S110.2*

Ajustes

El sensor MSE-S110.X no necesita de ajuste alguno. La alimentación se aplica por las conexiones 1 y 2 de la borna. Las señales de salida OUT1 y OUT2 se corresponden con el estado de los dispositivos CNY70 1 y 2 respectivamente. Con un voltímetro se pueden medir y analizar la tensión de las señales de salida entre GND y OUT1/OUT2 cuando, frente a los dispositivos se coloca, por ejemplo, un objeto blanco/negro.

Aplicaciones

Este tipo de sensores puede tener gran cantidad de aplicaciones tanto a nivel industrial, como a nivel didáctico, microbótica, etc. A continuación y, a modo de ejemplo, se presentan algunas ideas.

En la figura AN3-5 se muestra una aplicación consistente en contar objetos que pasan frente a un sensor. Estos objetos deben estar suficientemente contrastados con el fondo. Se obtiene un tren de pulsos que se generan en cada transición de blanco a negro. Se puede emplear la versión MSE-S110.1 con un único dispositivo.

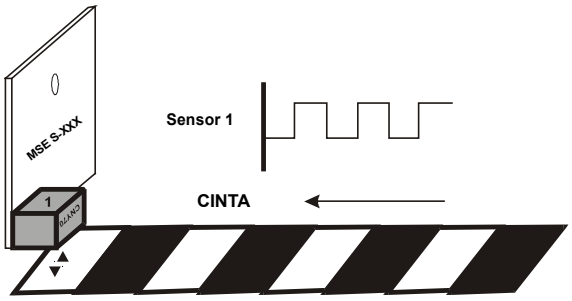
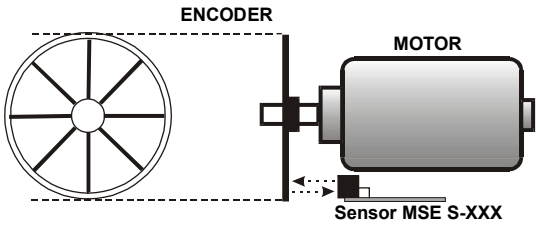


Figura AN3-5. *Contando objetos con el sensor de reflexión MSE-S110.1*

Empleando también la versión MSE-S110.1 con un dispositivo, podemos controlar el desplazamiento de giro de un motor y su velocidad. Ver la figura AN3-6. Efectivamente, al eje del motor se le acopla un disco con radios negros y blancos a modo de encoder. El sensor se coloca frente al disco. Durante el giro del motor se generan una serie de pulsos. Si conocemos el arco que hay entre un radio del enconder y el siguiente, podemos conocer la rotación producida en el eje. Igualmente, midiendo la frecuencia de los pulsos podemos controlar la velocidad de movimiento o rpm.

Figura AN3-6. *Controlando el movimiento de un motor*



La figura AN3-7 muestra un ejemplo que permite conocer la velocidad y sentido de desplazamiento de una cinta transportadora o cualquier sistema de transmisión similar. Se emplea la versión MSE-S110.2 con dos dispositivos CNY70. Durante el desplazamiento de la cinta, el sensor genera dos señales desfasadas 180º y que se obtienen por las salidas OUT1 y OUT2. Analizando si la señal de OUT1 está adelantada respecto a OUT2 o viceversa, se determina el sentido del desplazamiento. Igualmente si se mide la duración de los pulsos o su frecuencia, también se puede determinar la velocidad. La idea puede ser aplicada, entre otras, al encoder anterior.

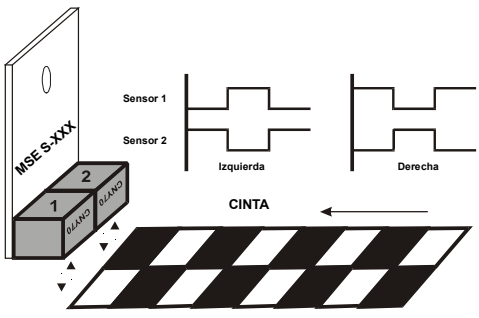


Figura AN3-7. *Conocer la velocidad y sentido de movimiento mediante el sensor de reflexión MSE-S110.2*

Una de las aplicaciones mas extendidas en el campo de la microbótica consiste en el control de la trayectoria de una estructura móvil (p.e. el robot). Este se desplaza sobre una superficie clara y la trayectoria se determina por una línea negra, ver la figura AN3-8. También puede ser a la inversa. Es necesario conocer cuándo

el robot se sale de la línea, bien por la izquierda o bien por la derecha, con objeto de hacer las correcciones necesarias en la dirección del mismo. Se emplea el sensor MSE-S110.2 con dos dispositivos CNY70.

Analizando las señales de salida OUT1 y OUT2 se determina el desvío que se ha producido. Si ambas señales son de nivel "1", el robot está en la trayectoria correcta, sobre la línea negra del ejemplo. Si OUT1 está a "1" y OUT2 a "0", significa que el robot se ha desviado por la izquierda, en caso contrario el desvío se ha producido por la derecha.

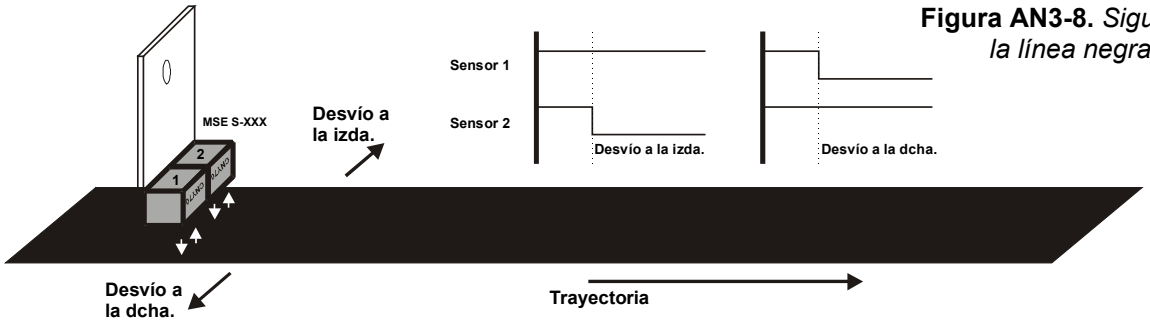


Figura AN3-8. *Siguiendo la línea negra*

AN3-4 EL SENSOR ULTRASONICO MSE-S120

Basado en los ultrasonidos, el sensor es capaz de detectar cualquier movimiento o obstáculo dentro de su radio de acción. Ver la figura AN3-9. Una cápsula ultrasónica emite una señal con una frecuencia en torno a los 40KHz. Cuando la señal rebota sobre un objeto, es captada por otra cápsula receptora. Tras amplificar y acondicionar la señal recibida, se genera un impulso lógico de salida por la conexión OUT (3) de la borna J1. Dicho pulso es activo por flanco ascendente y tiene una duración aproximada de 0.5 ".

Mediante el jumper JP1 el sensor puede actuar de forma autónoma o de forma controlada. Cuando JP1 está cerrado, el circuito queda activado permanentemente y se emite señal de forma constante. En este caso la entrada CONTROL (4) debe estar desconectada. Si se abre el jumper JP1, el funcionamiento del circuito se controla mediante la entrada CONTROL (4) de la borna J1. Efectivamente, cuando esta entrada se pone a nivel "1" el circuito se activa en modo normal de funcionamiento. Si la entrada CONTROL se pone a nivel "0" el circuito deja de emitir señal ultrasónica con lo que únicamente se captan o reciben ruidos, interferencias, armónicos, etc.., u otras fuentes ultrasónicas. El radio de acción queda por tanto prácticamente inexistente y reducido al mínimo. En cualquiera de los casos la sensibilidad del sensor se ajusta mediante el potenciómetro P1.

El circuito dispone de orificios que permiten una flexible instalación y sujeción del mismo sobre cualquier tipo de estructura.

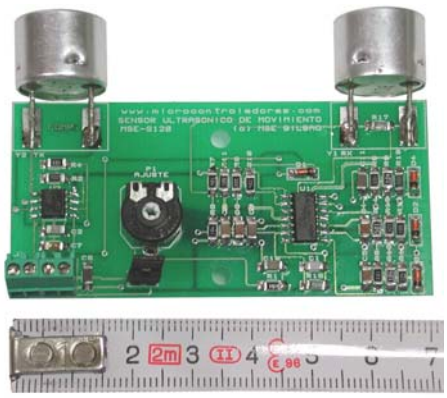


Figura AN3-9. *El sensor ultrasónico MSE-S120*

Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones aproximadas del circuito	70 x 35	mm
Tensión de alimentación	5	Vcc
Consumo aprox. en reposo	8.5	mA
Consumo aprox. en activación	8.6	mA
Tensión de salida en reposo ("0")	0	Vcc
Tensión de salida en activación ("1")	> 3.5	Vcc
Duración aproximada del pulso en activación	1000	mS
Flanco de salida activo	ascendente	
Radio de acción aproximado	10-250	cm

Conexionado

Se realiza mediante una borna de 4 contactos con paso 2.54, tal y como se muestra en la figura AN3-10.

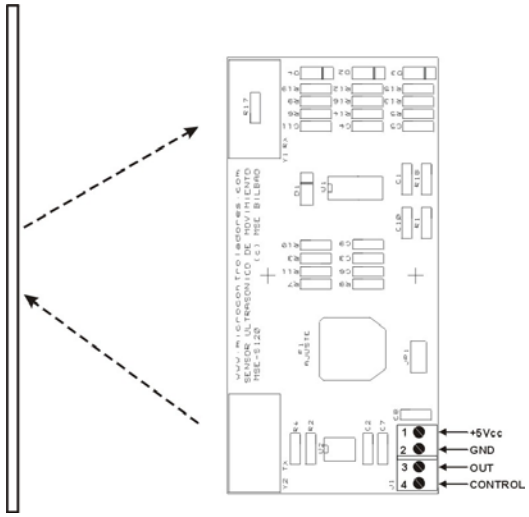


Figura AN3-10. *Conexionado del sensor ultrasónico MSE-S120*

Ajustes

El ajuste del sensor ultrasónico de movimiento MSE-S120 permite regular el radio de acción necesario para disparar la señal de salida. Se realiza mediante el potenciómetro ajustable P1.

La alimentación de +5Vcc se aplica a través de las conexiones 1 y 2 de la borna. Con un voltímetro se mide la señal de salida entre las conexiones 2 y 3. Se recomienda mantener cerrado el jumper JP1. Se realiza la aproximación de cualquier objeto hacia las cápsulas y se mide el momento en que la señal de salida sube a nivel lógico "1". Mediante el potenciómetro P1 se ajusta el momento en que la tensión de salida OUT pasa a "1" en función de la distancia deseada entre el objeto y las cápsulas.

Puede ser necesario mover ligeramente las cápsulas con objeto de concentrar o dispersar el haz ultrasónico de la señal emitida/recibida.

Se recomienda que, tras conectar la alimentación, se espere un mínimo de 1000mS antes de procesar la señal OUT de salida. Ello es debido a que el circuito necesita de ese tiempo para su propia estabilización y durante el cual se pueden generar falsas señales de disparo.

Aplicaciones

El sensor ultrasónico de movimiento MSE-S120 es capaz de generar una señal lógica cada vez que se detecte un objeto o movimiento dentro de su radio de acción. Las aplicaciones son numerosas:

- Automatismos digitales en los que se procesan señales que provienen de diferentes sensores o transductores, proximidad de objetos, movimiento de los mismos, etc.
- Alarmas activadas por presencia o movimientos provocados por intrusos.
- Microbótica, en donde es necesario que se actúe en función del entorno que rodea al robot.

AN3-5 EL SENSOR DE LUZ MSE-S130

Se trata de un dispositivo sensor de luz visible basado en el foto transistor BPW40. El circuito, mostrado en la figura AN3-11, se alimenta con una tensión de +5Vcc. La variación de luz ambiente detectada por el foto transistor es acondicionada y amplificada para proporcionar a la salida una tensión variable entre 0,1 Vcc y 5Vcc en función de dicha variación.

La tensión de salida se obtiene por la salida OUT (conexión 3 de la borna) y puede ser tratada de forma analógica o digital en los posteriores procesos de automatización y control.

El dispositivo dispone de un orificio que permite una flexible instalación y sujeción del mismo sobre cualquier tipo de estructura.

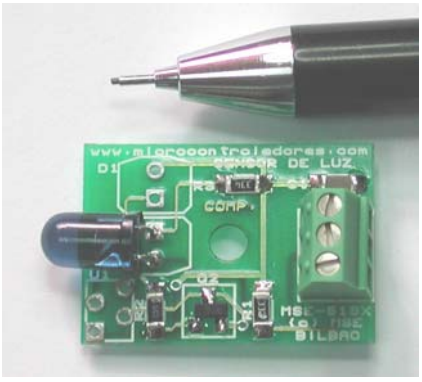


Figura AN3-11. *El sensor de luz visible MSE-S130*

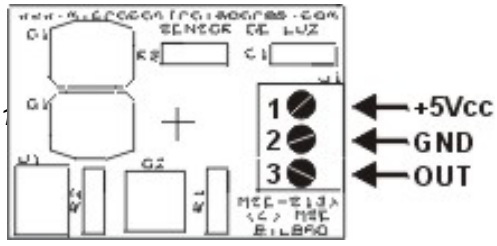
Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones del circuito	25 x 18	mm
Tensión de alimentación	5	Vcc
Consumo en corte (máxima luz incidente)	< 20	μA
Consumo en saturación (mínima luz incidente)	< 2	mA
Tensión de salida en corte (máxima luz)	5	Vcc
Tensión de salida en saturación (mínima luz)	< 0.1	Vcc
Nivel lógico "1" a partir de	> 2.5	Vcc
Nivel lógico "0" por debajo de	< 1	Vcc

Conexionado

Se realiza mediante una borna de 3 contactos con paso 2.54, tal y como se muestra en la figura AN3-12.

Figura AN3-12. *Conexiones del sensor de luz MSE-S1*



Ajustes

El dispositivo MSE-S130 no necesita de ningún ajuste especial. En determinadas aplicaciones quizá se deba mover ligeramente la posición del sensor según la orientación de la fuente de luz a medir.

En algunas aplicaciones también puede resultar interesante rodear al sensor mediante algún tipo de cilindro opaco dejando libre sólo el extremo. De esta forma se evita captar luz ambiente y sólo se capta la luz que incida directamente sobre dicho sensor.

El circuito se alimenta con +5Vcc. Con ayuda de un voltímetro se mide la tensión de salida entre GND y OUT (conexiones 2 y 3 de la borna). El usuario puede realizar unas medidas de referencia anotando diferentes tensiones de salida en función de distintos umbrales de luz.

Teniendo en cuenta que, los diferentes colores absorben la luz en mayor o menor cantidad, es posible analizar cómo varían los umbrales de luz en función del color de un objeto. Así, se pueden tomar diferentes referencias o muestras de la tensión de salida, para objetos de distintos colores.

Aplicaciones

El sensor de luz MSE-S130 es capaz de proporcionar una tensión variable en función de la luz que incide sobre él. Se puede emplear en todas aquellas aplicaciones en las que sea necesario hacer un tratamiento de la luz visible, como pueden ser aplicaciones de carácter didáctico, entretenimiento, control de procesos, robótica, etc.

El procesamiento de la señal de salida se puede hacer en forma digital o analógico. En este último caso será necesario el empleo del convertidor ADC correspondiente, pero a cambio es posible realizar un análisis de la cantidad de luz, color, etc.

AN3-6 EL DETECTOR IR DE OBSTÁCULOS MSE-S135

El circuito MSE-S135 es un detector IR de obstáculos que detecta la presencia de un objeto sin contacto físico con el mismo. Se muestra en la figura AN3-13. Consiste en un emisor/detector de luz infrarroja modulada. Esta característica lo hace prácticamente inmune a interferencias provocadas por otras fuentes de luz.

Un diodo emisor emite una haz infrarrojo modulado a una frecuencia de 7.7 KHz. El rebote de dicho haz sobre un objeto, es captado por un foto transistor detector que acondiciona la señal recibida, compara si corresponde con la señal emitida y, en caso afirmativo, genera una señal de salida activa por flanco descendente. Si no se detecta ningún rebote de la señal emitida, la salida se mantiene en reposo, a nivel lógico "1".

El dispositivo dispone de un orificio que permite una flexible instalación y sujeción del mismo sobre cualquier tipo de estructura.

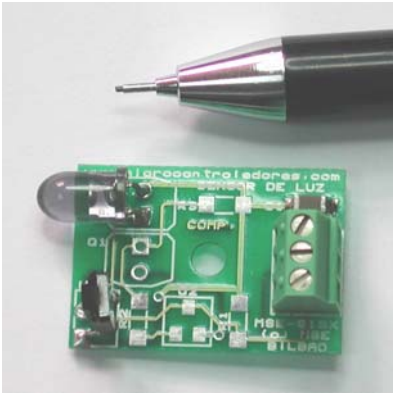


Figura AN3-13. *El sensor IR de obstáculos MSE-S135*

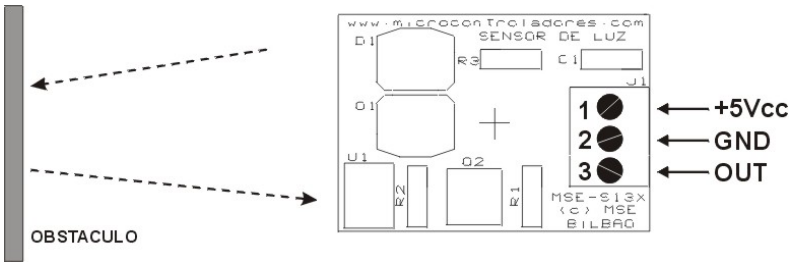
Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones del circuito	25 x 18	mm
Tensión de alimentación	5	Vcc
Consumo en reposo (aprox)	6.5	mA
Consumo activado (detección de señal rebotada)	7.8	mA
Tensión de salida en reposo ("1")	> 4.5	Vcc
Tensión de salida activado ("0")	< 0.2	Vcc
Longitud de onda de emisión/recepción	940-950	nm
Distancia máxima de detección de obstáculo (aprox). Esta distancia varía en función del color del objeto. Los colores oscuros absorben más luz y por tanto la distancia de detección es menor.	70	mm

Conexionado

Se realiza mediante una borna de 3 contactos con paso 2.54, tal y como se muestra en la figura AN3-14.

Figura AN3-14. *Conexionado del sensor IR de obstáculos MSE-S135*



Ajustes

El dispositivo MSE-S135 no necesita de ningún ajuste especial. En determinadas aplicaciones quizá se deba mover ligeramente la orientación tanto del emisor IR como del receptor, al objeto de orientar el haz de luz infrarroja.

En primer lugar se alimenta el circuito con +5Vcc. Con ayuda de un voltímetro se mide la señal de salida entre GND y OUT (conexiones 2 y 3 de la borna). Cuando no se detecta ningún obstáculo (reposo), la tensión medida en OUT debe ser de unos 5Vcc (nivel lógico "1"). Al colocar un objeto frente al circuito, a una distancia de unos 70mm o menos (depende del color del objeto), se debe medir una tensión de 0Vcc (nivel lógico "0").

Aplicaciones

El dispositivo MSE-S135 es capaz de detectar objetos a distancia, sin necesidad de que haya contacto físico con los mismos. Entre las numerosas aplicaciones posibles cabe citar las siguientes:

- Automatismos y control industrial. Detección de presencia, detección de posicionamiento, detección de paso de piezas, encoders, finales de carrera, etc..
- Robótica y microbótica. Detección de obstáculos, protección, posicionamiento, etc..

AN3-7 DRIVER AMPLIFICADOR MSE-A100

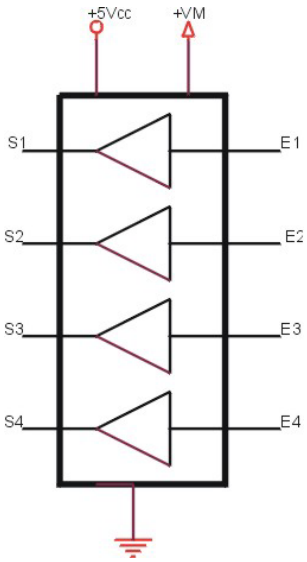
Se trata de un driver de propósito general basado en el dispositivo L293B de la firma SGS-THOMSON. Se muestra en la figura AN3-15.

Figura AN3-15. *El Driver amplificador MSE-A100*

Consiste en 4 canales amplificadores totalmente independientes entre si. Cada canal es capaz de soportar corrientes de salida de 1 A con picos de hasta 2 A. Poseen una alta inmunidad al ruido, protección de sobre temperaturas y tensión de alimentación de las cargas separada de la tensión de alimentación de la lógica. La señal de entrada de cada canal es compatible con señales TTL. Las señales de salida disponen de los correspondientes diodos de absorción para las corrientes inversas que generan las cargas inductivas. La figura AN3-16 muestra el esquema simplificado del driver MSE-A100, junto con una descripción de sus señales.

SEÑAL	DESCRIPCION
E1-E4	Señales de entrada, una por cada canal. Estas señales son compatibles con niveles lógicos TTL.
S1-S4	Señales amplificadas de salida, una por cada canal. Estas se conectan a las cargas que se desean controlar. Cada salida puede soportar cargas de hasta 1 A.
+5Vcc	Entrada de +5V para alimentación de la lógica interna.
+VM	Entrada de tensión para la alimentación de las cargas cuyo valor máximo es de 35V
GND	Tierra de alimentación.

Figura AN3-16. *Descripción de señales del driver MSE-A100*



Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones del circuito	45 x 33	mm
Tensión de alimentación para la lógica interna (+5Vcc)	5	V
Tensión máxima de alimentación de las cargas (+VM)	35	V
Tensión de entrada máxima en E1-E3 a nivel bajo	1.5	V
Tensión de entrada mínima en E1-E3 a nivel alto	2.3	V
Corriente máxima de entrada en E1-E3 a nivel bajo	-10	µA
Corriente típica de entrada en E1-E3 a nivel alto	30	µA
Intensidad máxima de salida en S1-S3	1000	mA
Intensidad de pico máxima en S1-S3	2000	mA
Disipación total de potencia	5	W

Conexionado

Se realiza mediante una serie bornas que permiten una fácil conexión. Se presenta en la figura AN3-17.

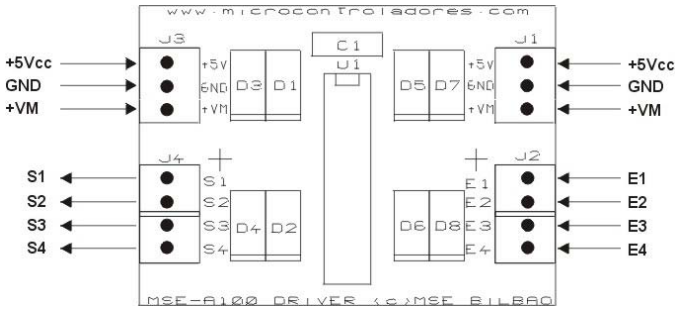


Figura AN3-17. *Conexiones del driver amplificador MSE-A100*

El driver MSE-A100 puede controlar diferentes tipos de cargas. A continuación se muestra, a modo de ejemplos, la conexión de MSE-A100 con diferentes tipos de periféricos. Así, la figura AN3-18, muestra la conexión del driver con cargas luminosas de tipo led.

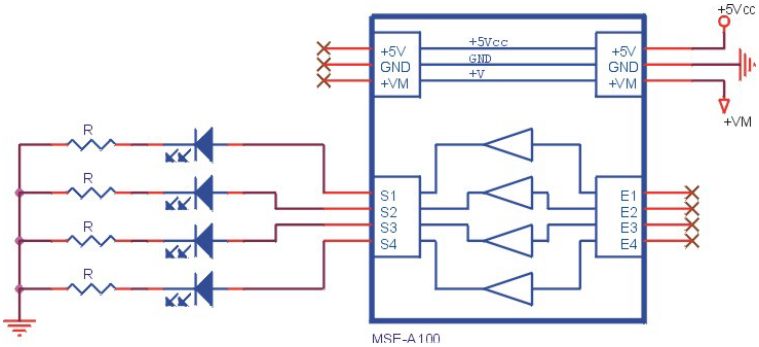


Figura AN3-18. *Conexión del MSE-A100 con cargas tipo led*

La resistencia R de absorción asociada a cada led se debe calcular en función de la tensión +VM empleada según la siguiente fórmula: R = (+VM – VLED) / ILED

La figura AN3-19 muestra la conexión del driver MSE-A100 con dos motores CC. Cada motor se conecta con dos de las salidas y se gobierna desde las correspondientes dos entradas. Tal y como se muestra en la tabla, es posible controlar la conexión/desconexión del motor así como el sentido de giro del mismo. También es posible regular la velocidad de cualquiera de los dos motores. Basta con aplicar por la entrada apropiada una señal PWM.

E1	E2	MOTOR 1
0	0	OFF
0	1	Giro horario
1	0	Giro antihorario
1	1	OFF

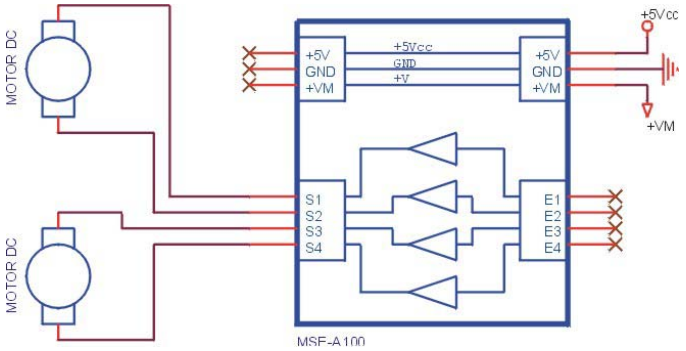


Figura AN3-19. *Controlando 2 motores con el MSE-A100*

La figura AN3-20 muestra la forma de conectar, a modo de ejemplo, dos relés y dos motores CC. En este caso tanto los relés como los motores se conectan a cada una de las 4 salidas disponibles, por lo que únicamente pueden tener el estado ON/OFF. En el caso de los motores sólo pueden tener un único sentido de giro, que será horario o antihorario en función de cómo se realicen las conexiones de los mismos.

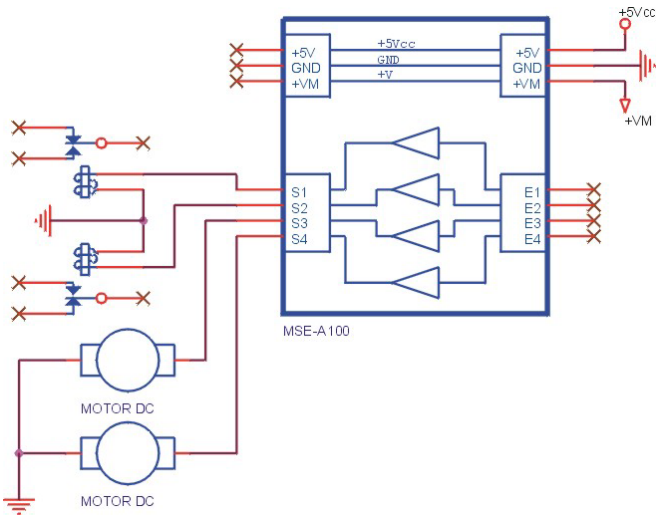


Figura AN3-20. *Controlando motores y relés con el MSE-A100*

Finalmente, la figura AN3-21 presenta la conexión del driver MSE-A100 con un motor paso a paso (P-P) de dos bobinas.

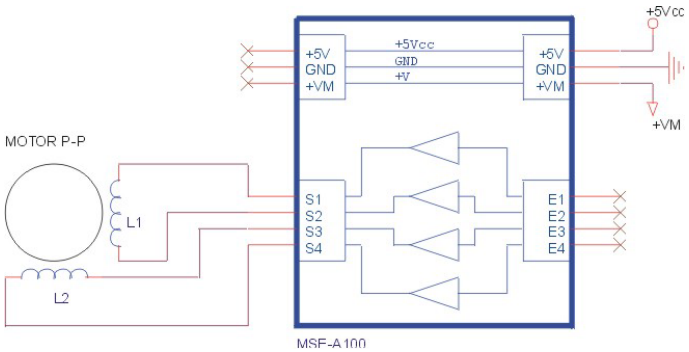


Figura AN3-21. *Gobernando un motor PAP con el MSE-A100*

Según las combinaciones binarios que se apliquen por las entradas E1-E3, las bobinas se excitan con una determinada polaridad y produciendo un desplazamiento de rotación en el eje del motor. El número de grados de

esta rotación o "paso" dependerá del motor empleado. Las siguientes tablas muestran las secuencias binarias que han de aplicarse para producir un giro en uno u otro sentido.

SENTIDO HORARIO				
PASO	**E4**	**E3**	**E2**	**E1**
1	1	0	0	1
2	0	1	0	1
3	0	1	1	0
4	1	0	1	0

SENTIDO ANTI HORARIO				
PASO	**E4**	**E3**	**E2**	**E1**
1	1	0	1	0
2	0	1	1	0
3	0	1	0	1
4	1	0	0	1

Ajustes

El driver MSE-A100 no necesita de ningún tipo de ajuste ni calibración

Aplicaciones

MSE-A100 es un driver de 4 canales de propósito general capaz de actuar sobre diferentes tipos de cargas. Su empleo está dirigido a cualquier aplicación en la que sea necesario amplificar señales lógicas de control para ser aplicadas a diferentes tipos de actuadores: diferentes tipos motores, relés, indicadores luminosos, sonoros, etc.

AN3-8 CAMARAS DE VIDEO MSE-V1XX

Se describen dos cámaras de vídeo con salida por R.F. y que se muestran en la figura AN3-22:

MSE-V10X Cámara en b/n con salida por radio frecuencia. Está disponible en dos versiones. **MSE-V100** tiene su salida sintonizada en el canal 12 de VHF de TV, a 224 MHz. **MSE-V102** tiene la salida sintonizada en el canal 22 de UHF de TV, a 479 MHz.

MSE-V11X Cámara en color con salida por radio frecuencia. Está disponible en dos versiones. **MSE-V110** tiene su salida sintonizada en el canal 12 de VHF de TV, a 224 MHz. **MSE-V112** tiene la salida sintonizada en el canal 22 de UHF de TV, a 479 MHz.

Cualquiera de los dos modelos se presentan sobre una placa impresa de reducidas dimensiones (62 x 30 mm) y disponen de dos modos de funcionamiento que se establece mediante el jumper JP1. Cuando está cerrado el circuito se mantiene en constante funcionamiento. La entrada CONTROL (conexión 3 de la borna) debe estar sin conexión. Si el jumper se abre, el funcionamiento ON/OFF de la cámara se controla mediante la entrada de CONTROL. Un nivel "1" por esta entrada la activa, un "0" la desactiva, reduciendo así el consumo de la aplicación final.

Los moduladores de R.F. están basados en los módulos AUREL MAV-VHF224 (canal 12) y MAV-UHF479 (canal 22). Estos vienen insertados en la placa impresa a través del correspondiente zócalo, y son fácilmente intercambiables. Estos módulos se pueden adquirir por separado y se seleccionan según el canal libre disponible en la zona.

IMPORTANTE: Las emisiones realizadas se hacen en las bandas comerciales de VHF y UHF. Ingeniería de Microsistemas Programados S.L. no se hace responsable de la utilización que de estos dispositivos haga el usuario ni de las interferencias que pudieran provocarse en otros receptores de TV.

Un trozo de cable de unos 15 cm soldado en el terminal de antena (ANT.) de la placa impresa, mejora notablemente el alcance y calidad de la imagen emitida.

.

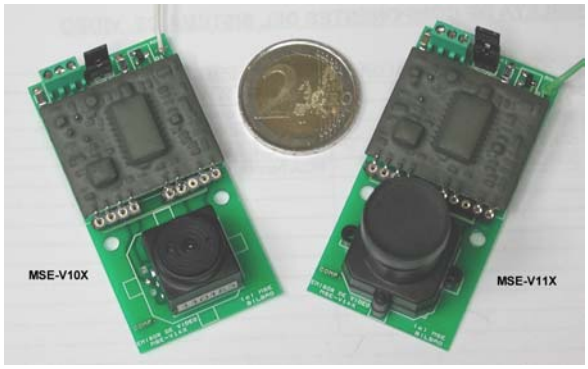


Figura AN3-22. *Cámaras de vídeo MSE-V1XX*

Características técnicas

MSE-V100/MSE-V102 b/n

PARAMETRO	VALOR	UNIDAD
Tensión de alimentación	5	VCC
Sensor Omnivision CMOS 1/3 "		
Resolución	240	Líneas TV
Pixels	288x352	
Sensibilidad (con F=1,4)	2	Lux
Obturador electrónico	1/50 a 1/6000	Seg.
Apertura angular del objetivo	50	º
Consumo	100	mA
Potencia de salida en RF	2	mW/75Ω
Portadora de vídeo con modulación PAL CH12/CH22	224.5/479.55	MHz

MSE-V110/MSE-V112 color

PARAMETRO	VALOR	UNIDAD
Tensión de alimentación	5	VCC
Sensor Omnivision CMOS 1/3 "		
Area efectiva de imagen	5,78x4,19	mm
Pixels	628x582 PAL	
Obturador electrónico	1/50 a 1/5000	Seg.
Consumo	100	mA
Potencia de salida en RF	2	mW/75Ω
Portadora de vídeo con modulación PAL CH12/CH22	224.5/479.55	MHz

Conexionado

Se realiza mediante una borna de 3 contactos con paso 2.54, tal y como se muestra en la figura AN3-23.

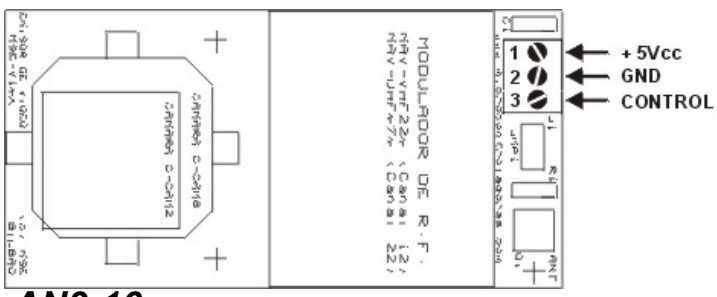


Figura AN3-23. *Conexionado de las cámaras MSE-V1XX*

Ajustes

Ninguno de los modelos de cámaras propuestos necesita de ajuste especial alguno. Como mucho se podrá ajustar el enfoque de la cámara girando de izda. a dcha. o viceversa el objetivo de la misma. Por las bornas 1 y 2 se conecta la tensión de alimentación de +5Vcc. Si el jumper JP1 está cerrado el circuito queda permanentemente activado. En estos casos se debe desconectar la entrada CONTROL (conexión 3 de la borna).

Aplicaciones

Este tipo de cámaras suponen una solución sencilla y de bajo coste para aplicaciones tales como vigilancia, alarmas, visión artificial, microbótica, etc.

AN3-9 MEDIDOR ULTRASONICO SRF04

El módulo SRF04 consiste en un medidor ultrasónico de distancias de bajo costo desarrollado por la firma **DEVANTECH Ltd**. Emplea un microcontrolador PIC12C508 que realiza las funciones de control y dos cápsulas ultrasónicas de 40KHz. Se muestra en la figura AN3-24.

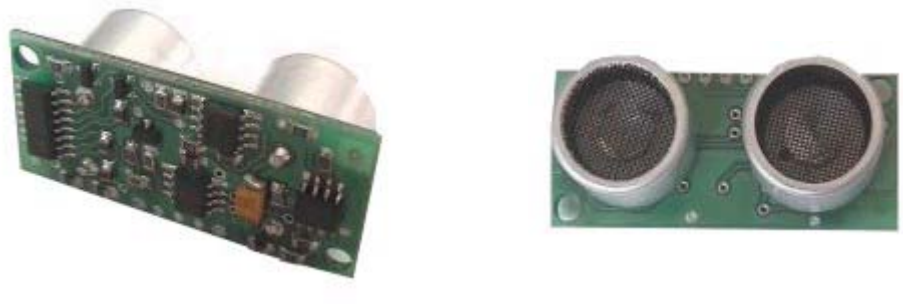


Figura AN3-24. *El medidor ultrasónico SRF04*

El rango de medidas es desde unos 3 cm hasta unos 3m aproximadamente. Medidas por debajo de los 3 cm provocan una serie de errores derivados del acoplamiento entre las propias cápsulas emisor-receptor del módulo. En este caso es muy difícil distinguir si la señal recibida es consecuencia de dicho acoplamiento o del eco recibido. Por otra parte es posible medir distancias superiores a los 3 m, pero nos podemos encontrar con problemas derivados de la dispersión del haz ultrasónico o de múltiples rebotes que pudieran generarse.

Tal y como se muestra en el diagrama de tiempos de la figura AN3-25, el modo de empleo es muy sencillo. Externamente se aplica, por parte del usuario, un pulso de disparo o trigger. Se inicia la secuencia. El módulo transmite un tren de pulsos o "burst" de 8 ciclos a 40KHz. En ese momento la señal de salida ECO pasa a nivel "1". Cuando la cápsula receptora recibe la señal transmitida como consecuencia de haber rebotado en un objeto (eco), esta salida pasa de nuevo a nivel "0". El usuario debe medir la duración del pulso de esta señal, es decir, el tiempo en que la señal eco se mantiene a "1".

Con objeto de que el módulo se estabilice, se debe dejar un lapsus de tiempo de unos 10mS mínimo entre el momento en que la señal de eco pasa a "0" y un nuevo pulso de disparo que inicie el siguiente ciclo o medida.

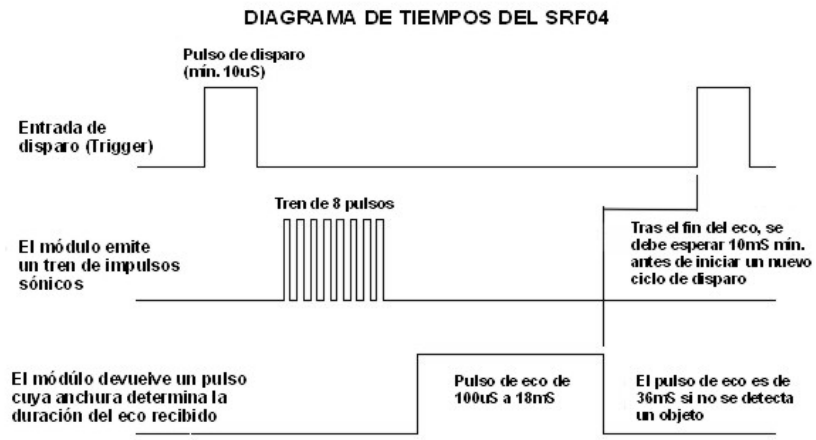


Figura AN3-25. *Diagrama de tiempos del medidor SRF04*

La duración del pulso eco de salida varía entre 100μS y 18mS, en función de la distancia entre las cápsulas del módulo y el objeto. La velocidad del sonido es de 29.15 μS/cm que, como realiza un recorrido de ida y vuelta, queda establecida en 58.30μS/cm. Así pues el rango mínimo que se puede medir es de 1.7 cm (100μS/58) y el máximo de 310 cm (18mS/58).

Características técnicas

PARÁMETRO	VALOR	UNIDAD
Dimensiones del circuito	43 x 20 x 17	mm
Tensión de alimentación	5	Vcc
Frecuencia de trabajo	40	KHz
Rango máximo	3	m
Rango mínimo	3	cm
Duración mínima del pulso de disparo (nivel TTL)	10	μS
Duración del pulso eco de salida (nivel TTL)	100-18000	μS
Tiempo mínimo de espera entre una medida y el inicio de otra	10	mS

Conexionado

El módulo emplea tan sólo 4 conexiones que se pueden realizar soldando directamente 4 cables o bien mediante un conector de 5 vías. Estas se muestran en la figura AN3-26.

+5Vcc	Tensión positiva de alimentación
ECO	Salida del pulso cuya anchura determina el tiempo del recorrido de la señal ultrasónica
Disparo	Entrada de inicio de una nueva medida. Se aplica un pulso con una duración mínima de 10Ms
N.C.	Línea sin conexión. Se emplea en la fase de fabricación y comprobación del propio módulo. No conectar nada.
GND	Tierra de alimentación.

Figura AN3-26. *Conexionado del medidor ultrasónico SRF04*

Ajustes

El módulo SRF04 viene perfectamente ajustado y calibrado de fábrica, por lo que no necesita de ningún tipo de ajuste. Su funcionamiento se puede verificar aplicando una serie de pulsos por la entrada de disparo. Con ayuda de un osciloscopio se puede medir la anchura de la señal eco de salida. Esta anchura, representada en μS y dividida entre 58.30 nos permite calcular la distancia del objeto.

Aplicaciones

El módulo SRF04 es capaz de generar una señal lógica de salida cuya duración determina la distancia de un objeto. Las aplicaciones son numerosas, citamos unas cuantas a modos de ejemplo:

- Aplicaciones de control conde se deba actuar en función de la distancia o tamaño de objetos diversos.
- Alarmas activadas cuando el intruso se aproxima a una determinada distancia
- Microbótica en donde es necesario que se actúe en función de la distancia que separa al robot de cualquier otro objeto.

AN3-10 MEDIDOR ULTRASONICO SRF08

El módulo SRF08 consiste en un medidor ultrasónico de distancias de bajo costo desarrollado por la firma **DEVANTECH Ltd**. y es una versión mejorada del módulo SRF04. Emplea un microcontrolador PIC16F872 que realiza todas las funciones de control e interface, dos capsulas ultrasónicas de 40KHz y una célula LDR capaz de proporcionar una medida de luz ambiente. Se muestra en la figura AN3-27.

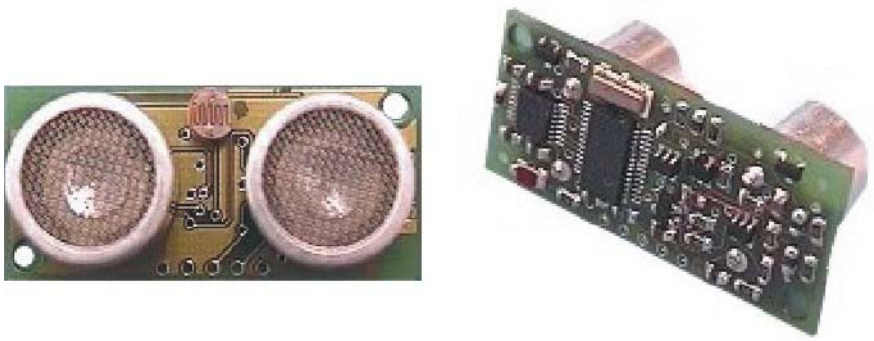


Figura AN3-27. *El medidor ultrasónico SRF08*

Las principales diferencias del SRF08 frente al SRF04 son las siguientes:

- Rango máximo de distancia hasta 6 m
- El SRF04 necesita dos conexiones por cada módulo a controlar. El SRF08 se controla desde un bus I2C estándar, por lo que se pueden gobernar varios módulos empleando 2 únicas conexiones.
- El consumo es reduce a 3 mA en standby y 15mA en funcionamiento
- Es capaz de medir diferentes ecos recibidos por la señal ultrasónica que puede rebotar contra uno o varios objetos a diferentes distancias.
- Tanto la ganancia de los amplificadores internos como el rango de mediadas es ajustable por el usuario
- Dispone de una célula LDR que permite realizar medidas de luz ambiente.
- Ofrece una lectura directa que se puede representar en centímetros, pulgadas o micro segundos.
- Un diodo led en la parte posterior del módulo genera un código de intermitencias que expresa la dirección I2C actual del módulo así como el inicio de una nueva medida.

La comunicación con el módulo SRF08 se realiza según el protocolo I2C, disponible en la mayor parte de los microcontroladores actuales aunque también puede ser implementado por software. La comunicación se realiza de la misma manera que con cualquier otro dispositivos I2C. La dirección del módulo es, por defecto, la 0xE0, aunque existe la posibilidad de que el usuario cambie esta dirección por cualquiera de las 16 siguientes: 0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF2, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC o 0xFE. Esto permite controlar hasta 16 módulos SRF08 con un mismo bus (2 líneas).

Además de estas 16 direcciones, todos los módulos responden a la dirección 0x00 de llamada general. Esto permite escribir un determinado comando en esa dirección que inicie una nueva medida en todos los módulos SRF08 disponibles en el bus, al mismo tiempo. Posteriormente, la medida de cada módulo se lee de forma individual, indicando su dirección particular.

Características técnicas

PARAMETRO	VALOR	UNIDAD
Dimensiones del circuito	43 x 20 x 17	mm
Tensión de alimentación	5	Vcc
Frecuencia de trabajo	40	KHz
Rango máximo	6	m
Rango mínimo	3	cm
Ganancia variable en 32 pasos	94 – 1025	
Sensor de luz en la cara anterior del módulo		
Capaz de medir hasta 17 señales de eco		
Devuelve la lectura en pulgadas, centímetros o micro segundos		
Conexión con bus I2C estándar		

Conexionado

El módulo emplea tan sólo 4 conexiones que se pueden realizar soldando directamente 4 cables o bien mediante un conector de 5 vías. Estas se muestran en la figura AN3-28.

+5Vcc	Tensión positiva de alimentación
SDA	Línea de E/S de datos correspondiente al bus I2C
SCL	Línea de entrada de la señal de reloj del bus I2C
N.C.	Línea sin conexión. Se emplea en la fase de fabricación y comprobación del propio módulo. No conectar nada.
GND	Tierra de alimentación.

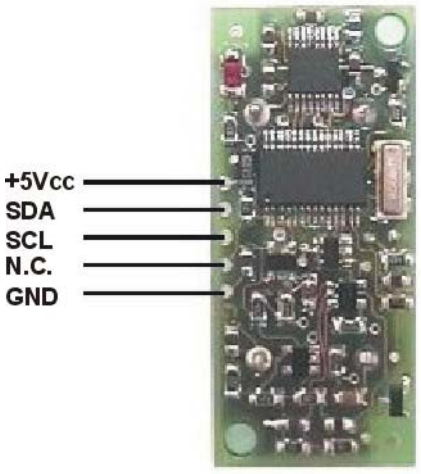


Figura AN3-28. *Conexionado del medidor SRF08*

Ajustes

El módulo SRF08 viene perfectamente ajustado y calibrado de fábrica, por lo que no necesita de ningún tipo de ajuste externo. Los únicos ajustes son, como ya se ha explicado, el ajuste del rango, de calibración y el cambio de la dirección I2C.

Aplicaciones

El módulo SRF08 es capaz de proporcionar información acerca de la distancia que hay entre el propio módulo y un objeto. También es capaz de medir la luz ambiente. Las aplicaciones son numerosas, citamos unas cuantas a modos de ejemplo:

- Aplicaciones de control conde se deba actuar en función de la distancia o tamaño de objetos diversos.
- Alarmas activadas cuando el intruso se aproxima a una determinada distancia
- Medición de luz ambiente
- Microbótica en donde es necesario que se actúe en función de la distancia que separa al robot de cualquier otro objeto

Desde la sección de "Downloads" de la página www.microcontroladores.com, se puede bajar información detallada sobre el manejo de este medidor ultrasónico-

AN3-11 COMPAS ELECTRÓNICO CMPS03

Se trata de un módulo diseñado por **DEVANTECH Ltd**. capaz de medir su posición en grados respecto al norte magnético. Es decir, consiste en un compás o brújula digital. Emplea los sensores KMZ51 de Philips sensibles al campo magnético de la tierra. Ambos sensores están montados en ángulo recto entre sí sobre la placa impresa y su salida se emplea para calcular la dirección en que se encuentran. Ver la figura AN3-29.

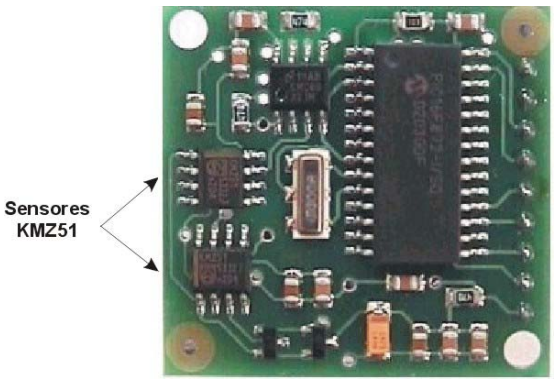


Figura AN3-29. *El compás CMPS03*

El dispositivo se alimenta con una única tensión de +5Vcc y proporciona dos salidas diferentes que indican su posición. Por una parte dispone de una salida PWM cuya anchura determina dicha posición y, por otra, un interface I2C para comunicación de datos con un microcontrolador master.

La salida PWM

La salida PWM es la mas sencilla de emplear. Proporciona un pulso modulado en anchura que representa el ángulo de la posición del módulo respecto al norte magnético. La anchura de este pulso varía desde 1mS (0º) hasta 36.99 mS (359.9º). Tiene por tanto una resolución de 100μS/º con un offset de +1mS. Cada vez que finaliza una medida, esta señal pasa a nivel "0" durante unos 65mS antes de iniciarse una nueva. El ciclo total tiene una duración mínima de 66mS (para una posición de 0º) y máxima de 102 mS (para una posición de 359.9º). El pulso lo genera un Timer interno de 16 bits que evoluciona cada 1μS, pero no es recomendable realizar medidas con intervalos inferiores a los 10μS (0.1º). La figura AN3-30 muestra un diagrama de tiempos en el que se han representado 4 medidas o ciclos diferentes.

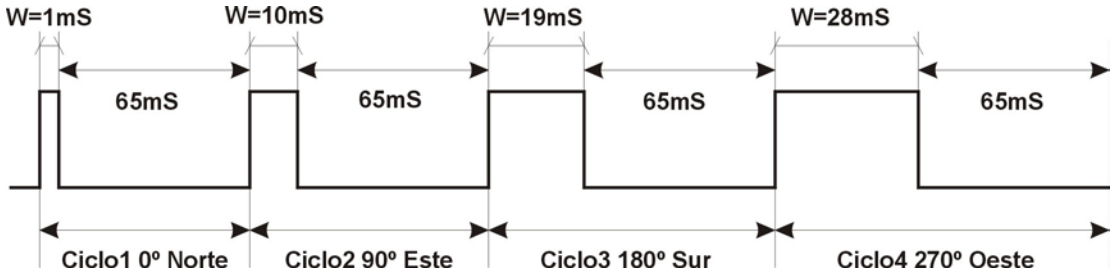


Figura AN3-30. *La salida PWM*

El ciclo 1 tiene una duración total de 66mS con una anchura W de 1mS. Este es el offset y representa una orientación de 0º, el Norte. El ciclo 2 con una duración total de 75ms, tiene una anchura de 10mS. Si se le quita 1mS de offset queda una anchura de 9mS que representa una posición de 90º, Este. En el ciclo 3 la duración es de 84mS y una anchura de 19mS, lo que indica posición Sur a 180º. Finalmente, el ciclo 4 con una duración de 93 mS y una anchura de 28mS, representa una posición Oeste a 270º.

El Interface I2C

El interface I2C permite conectar al módulo en un bus I2C junto con otros dispositivos y ser así gobernado desde un microcontrolador master, encargado de realizar las tareas oportunas. El protocolo empleado es el mismo que el que se emplea con cualquier otro dispositivo I2C.

Características técnicas

PARÁMETRO	VALOR	UNIDAD
Dimensiones del circuito	32 x 35	mm
Tensión de alimentación	5	Vcc
Consumo	20	mA
Resolución	0.1	º
Salida PWM en incrementos de 0.1mS	1-37	mS
Salida por el interface I2C (la velocidad de reloj puede ser desde 100KHz hasta 1MHz)	0-255 y 0-3599	º
Precisión después de la calibración	3-4	º

Conexionado

Todas las señales eléctricas, así como las conexiones del módulo, se representan en la figura AN3-31. A continuación se explican cada una de esas señales.

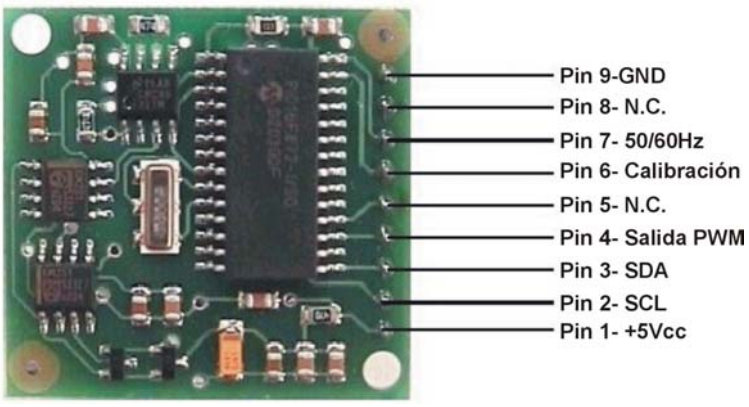


Figura AN3-31. *Conexiones del compás CMPS03*

Nº Pin	Señal	Descripción
1	+5Vcc	Entrada de alimentación de +5Vcc
2	SCL	Entrada de reloj procedente del bus I2C puede ser desde 100KHz (estándar) hasta 1 MHz
3	SDA	Entrada/salida de datos del bus I2C
4	Salida PWM	Salida modulada en anchura que representa la orientación del módulo.
5	N.C.	No conectada
6	CALIBRACION	Entrada para la calibración por hardware del módulo. Ver el apartado 4
7	50/60Hz	Mediante esta entrada se activa un filtro interno que permite anular en cierta medida los campos magnéticos generados por la red eléctrica. Cuando se pone a nivel alto "1" se activa un filtro de 60Hz (por defecto). Poniendo esta entrada a nivel bajo "0" se activa un filtro a 50Hz.
8	N.C.	No conectada
9	GND	Tierra de alimentación

NOTA: Las señales SCL y SDA necesitan de sendas resistencias pull-up conectadas a +5Vcc. En caso de emplearse la salida PWM, se recomienda colocar unas resistencias de 47K. En caso de emplearse el protocolo I2C para realizar las medidas, se recomiendan que estas resistencias sean de un valor entre 1.2K y 1.8K. En este caso es muy probable que las resistencias estén ya colocadas en el propio bus I2C, por lo que no habrá que poner otras.

Ajustes

Antes de proceder al calibrado o ajuste del móduloCMPS03 debemos asegurarnos de colocarlo horizontal y paralelo a la superficie de la tierra, con los componentes hacia arriba y alejado de cualquier objeto metálico especialmente si es magnético (destornilladores, tijeras, etc.). Se debe orientar hacia el norte tal y como se muestra en la figura AN3-32.

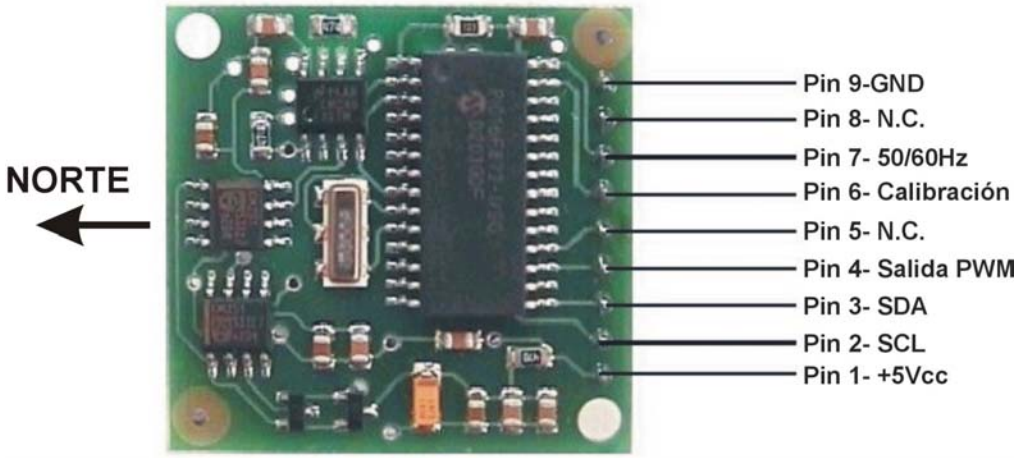


Figura AN3-32. *Orientación del CPS03 para el ajuste*

La calibración se puede hacer de dos formas: por el método hardware, empleando el pin de calibración o bien el método software, empleando el bus I2C. A continuación se explica la forma de proceder para ambos métodos.

Método hardware

Es probablemente el método mas sencillo. Se emplea la entrada de Calibración del pin 6. Este entrada dispone de su propia resistencia pull-up que, en estado de reposo, la mantiene a nivel "1". Para realizar el calibrado sólo es necesario aplicar por esta entrada un impulso negativo (transición alto-bajo-alto) por cada uno de los 4 puntos cardinales. Para ello basta con un simple pulsador. Un extremo de este se conecta con la entrada del pin 6 y el otro con GND. Cuando el pulsador está sin accionar, la entrada se mantiene a "1" (por el pull-up). Al pulsarse se aplica "0" y al soltarlo, la entrada vuelve a "1".

Se recomienda el empleo de una brújula que nos ayude a localizar los cuatro puntos cardinales con la mayor precisión posible y, poder así, orientar el módulo debidamente durante el proceso de calibración. Este proceso se resume a seguidamente:

1.- Colocar el módulo paralelo, horizontal y orientado al NORTE. Activar / desactivar el pulsador.
2.- Colocar el módulo paralelo, horizontal y orientado al ESTE. Activar / desactivar el pulsador.
3.- Colocar el módulo paralelo, horizontal y orientado al SUR. Activar / desactivar el pulsador.
4.- Colocar el módulo paralelo, horizontal y orientado al OESTE. Activar / desactivar el pulsador.

Método software

Este método emplea el propio interface I2C y su correspondiente protocolo para proceder al calibrado del módulo. Efectivamente, basta con escribir el valor 0xFF sobre el registro interno 0x0F en cada uno de los cuatro puntos cardinales. También se recomienda el empleo de una brújula que permita orientar el módulo en cada uno de esos puntos.

1.- Colocar el módulo paralelo, horizontal y orientado al NORTE. Escribir 0xFF en la posición 0x0F
2.- Colocar el módulo paralelo, horizontal y orientado al ESTE. Escribir 0xFF en la posición 0x0F
3.- Colocar el módulo paralelo, horizontal y orientado al SUR. Escribir 0xFF en la posición 0x0F
4.- Colocar el módulo paralelo, horizontal y orientado al OESTE. Escribir 0xFF en la posición 0x0F

Aplicaciones

El comportamiento del módulo CMPS03 es similar o equivalente al de un compás o brújula digital. Las aplicaciones pueden ser todas aquellas en las que sea necesario llevar a cabo una acción, en función de la posición u orientación de un objeto.

Por ejemplo, un robot móvil puede desplazarse siguiendo una trayectoria previamente establecida, localizar un objeto en una determinada posición, orientarse de forma automática, etc.

En la sección "Downloads" de www.microcontroladores.com se puede encontrar mas información técnica del compás CMPS03.

AN3-12 EL SINTETIZADOR SP03

Durante bastante tiempo todos los sistemas dedicados a la síntesis de voz han sido sistemas complejos y de elevado coste que impedían su empleo en aplicaciones populares y de consumo como puede ser la microbótica. El empleo de robots móviles dotados de sistemas de voz, puede tener un gran interés general.

La aparición del dispositivo WTS701 de la firma Winbomd ha cambiado radicalmente este panorama. Consiste en un único circuito integrado que incluye un procesador de textos a voz completo. Aunque la versión actual tiene un marcado acento de inglés norte americano, el fabricante asegura futuras versiones en las que el dispositivo sea capaz de reproducir diferentes tonos y acentos que se adapten a diferentes lenguajes.

La firma Devantech Ltd. ha diseñado el módulo de síntesis de voz SP03, objeto de la presente documentación. Este módulo integra el dispositivo WTS701, un amplificador de audio, regulador de 3 V, un

microcontrolador PIC que facilita la comunicación con la aplicación final del usuario y un altavoz de 40 mm. El interface con el usuario se realiza mediante un canal serie RS232 estándar, un bus I2C y un puerto paralelo que permite reproducir en el altavoz hasta 30 frases previamente definidas y grabadas en el módulo.

Está disponible también un software para PC, el SP03.EXE, que permite la edición y el volcado sobre el módulo de hasta 30 frases predefinidas.

Características generales

- Tensión de alimentación: 5 Vcc
- Consumo: 35mA en modo standby y de 60 a 100mA en modo de reproducción.
- Sintetizador: Basado en el dispositivo WTS701EM/T.
- Control tipo 1: Mediante un interface RS232 estándar con el canal serie de un PC (38400 baudios).
- Control tipo 2: Mediante bus I2C para facilitar la transferencia de datos con cualquier microcontrolador.
- Control tipo 3: Mediante una puerta paralelo de entrada de 5 pines.
- Almacenamiento: Hasta 32 frases predefinidas con un máximo de 1925 caracteres en total.
- Conversión texto a voz: Ilimitada mediante los interfaces RS232 o bus I2C.
- Altavoz: de 40 mm integrado en el propio módulo SP03.
- Amplificador: Integrado en el módulo, basado en el LM386 con una potencia de 325mW
- Dimensiones: 40mm x 40mm excluyendo los conectores.

La figura AN3-33 muestra ambas caras del módulo SP03

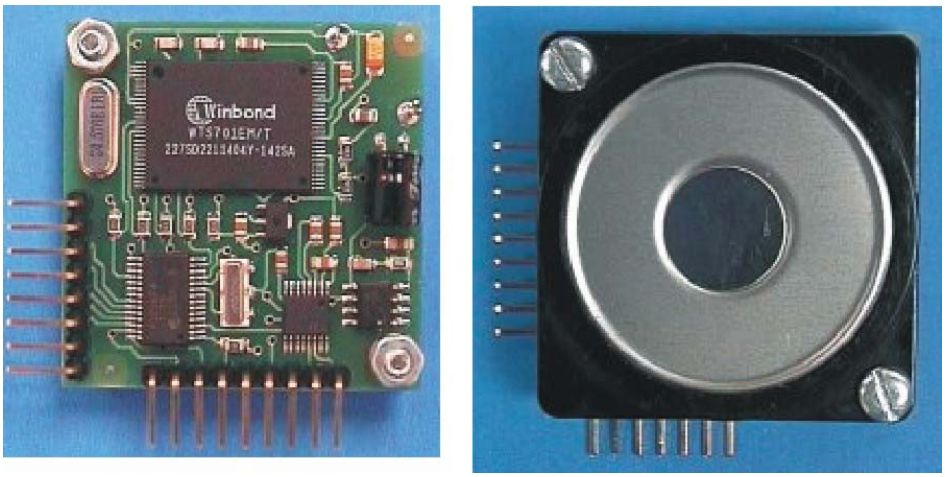


Figura AN3-33. *El sintetizador SP03*

Conexiones del sintetizador SP03

El módulo SP03 dispone de dos conectores, PL1 y PL2, para la conexión con la aplicación final del usuario. La tensión de alimentación de +5Vcc se puede aplicar por cualquiera de ellos. La figura AN3-34 muestra la distribución de las señales disponibles en ambos conectores, cuya descripción se resumen en las siguientes tablas.

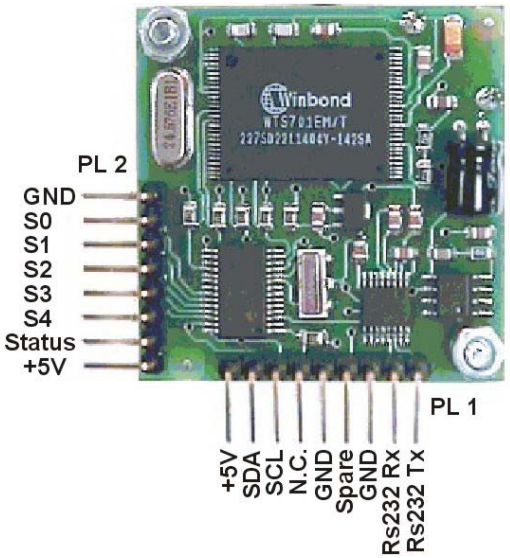


Figura AN3-34. *Los conectores PL1 y PL2*

CONECTOR PL1	
SEÑAL	**DESCRIPCIÓN**
+5V	Tensión de alimentación de +5Vcc / 100mA
SDA	Línea de datos del bus I2C
SCL	Línea de reloj del bus I2C
N.C.	No usado, no conectar
GND	Línea de tierra de alimentación
Spare	Línea indefinida, no conectar
GND	Línea de tierra de alimentación
RS232 Rx	Línea de recepción de datos, se conecta con la de transmisión del PC (Tx)
RS232 Tx	Línea de transmisión de datos, se conecta con la de recepción del PC (Rx)

CONECTOR PL2	
SEÑAL	**DESCRIPCIÓN**
+5V	Tensión de alimentación de +5Vcc / 100mA
Status	Se pone a "1" cuando hay actividad en el altavoz (reproducción de voz), en caso contrario se mantiene a "0"
S4	Entrada S4 para selección de frases predefinidas
S3	Entrada S3 para selección de frases predefinidas
S2	Entrada S2 para selección de frases predefinidas
S1	Entrada S1 para selección de frases predefinidas
S0	Entrada S0 para selección de frases predefinidas
GND	Línea de tierra de alimentación

En la sección "Download" de la página www.microcontroladores,com se puede encontrar información técnica mas exhaustiva así como software de comunicación con el PC.

Anexo 4

Relación de materiales del kit Home Boe-Bot

AN4.1 COMPONENTES ELECTRICOS/ELECTRONICOS

CANTIDAD	COD. PROVEEDOR	VALOR	DESCRIPCION
1	28158		Tarjeta de control Home Work con el Basic Stamp de Parallax
2	900-00008		Servos Parallax de rotación continua para la tracción del robot Home Boe-Bot
8	30R1C221	220	Resistencia de ¼ w y 5%
4	30R1C471	470	Resistencia de ¼ w y 5%
2	30R1C102	1K	Resistencia de ¼ w y 5%
2	30R1C202	2K	Resistencia de ¼ w y 5%
2	30R1C472	4K7	Resistencia de ¼ w y 5%
2	30R1C103	10K	Resistencia de ¼ w y 5%
2	350-00009		Células LDR (EG&G Vactec VT935G grupo B)
2	42PF103	10n	Condensador cerámico
2	43SR21104	100n	Condensador cerámico
2	05L5R		Diodos LED rojos de 5 mm
2	350-00014		Receptor IR (Panasonic PNA4602M)
2	350-00003		Leds emisores de IR
1	900-00001		Piezo eléctrico
1	79PTP4R6		Porta pilas para 4 pilas R6 de 1,5V
1	83C2M9P9S		Cable 2M sud D9 macho D9 hembra

AN4.2 PIEZAS Y ACCESORIOS VARIOS

CANTIDAD	COD. PROVEEDOR	VALOR	DESCRIPCION
1	700-00022		Chasis de aluminio del robot Home Boe-Bot
2	721-00001		Ruedas delanteras de plástico
4	721-00002		Bandas elásticas para las ruedas (neumáticos)
1	700-00009		Bola perforada para la rueda trasera
1	700-00023		Pasador de 1/16" para sujeción de la rueda trasera
2	350-90000		Porta leds
2	350-90001		Tapas para el porta leds 350-90000
4	451-00303		Tira de 3 pines macho-macho paso 2.54mm
2	700-00056		Alambres de acero para los bumpers o "bigotes"
2	800-00016		Bolsas de 10 cables para montaje y cableado en proto-board
2	79AB14		Arandelas de baquelita de 3mm de diámetro
8	79T6M3		Tornillos M3 x 6 mm
8	79T10M3		Tornillos M3 x 10 mm
2	79T10M3AVPH		Tornillos M3 x 10 mm con cabeza plana avellanada
2	79T15M3		Tornillos M3 x 15 mm
10	79TRM3		Tuercas M3
2	79SS10		Separadores h-h M3 x 10 mm (huecos con rosca interior)
4	79SS20		Separadores h-h M3 x 20 mm (huecos con rosca interior)
1	79PSHG27		Goma pasa chasis de 10 mm
1	CD-ROM		CD-ROM con manual y ejemplos del robot Home Boe-Bot

Nota: Ingeniería de Microsistemas S.L se reserva el derecho de cambiar y/o modificar cualquiera de los componentes anteriores, por otros similares o equivalentes.