Robotics mit dem Boe-Bot

Handbuch

VERSION 2.1 GER

AUTORISIERTE DEUTSCHE ÜBERSETZUNG

Hinweise zur deutschen Übersetzung:
Der vorliegende Text ist eine sinntreue Übersetzung des englischen Originaltexts "Robotics with the Boe-Bot, Version 2.1" für (junge) Robotiker mit eingeschränkten Englischkenntnissen. Übersetzt wurden alle Texte, welche für den Aufbau und die Programmierung des Boe-Bot nötig sind.
Nicht übersetzt wurden die PBASIC-Programme, sowie jene Teile des Originals, die entweder nur für den amerikanischen Markt relevant sind, oder welche ohne ausreichende Englischkenntnisse ohnehin nicht nutzbar sind (wie z.B. die englische Befehlsreferenz im Editor oder die Hinweise zu den vielfältigen englischsprachigen Internet-Ressourcen). Alle nicht übersetzten Texte liegen hier noch im Original vor und sind gelb markiert.
In seltenen Fällen hat der Übersetzer spezielle Hinweise für deutschsprachige Leser hinzugefügt. Diese sind alle an der blauen Textfarbe erkennbar.

Für die Übersetzung: Ariane & Thomas Ernst, CH-8427 Rorbas

PARALLAX INC

WARRANTY

Die folgenden Abschnitte befassen sich mit Garantien, Marken- und Konsumentenschutzrechten. Sie gelten für den amerikanischen Markt und sind hier daher im Original belassen.

Parallax warrants its products against defects in materials and workmanship for a period of 90 days from receipt of product. If you discover a defect, Parallax will, at its option, repair or replace the merchandise, or refund the purchase price. Before returning the product to Parallax, call for a Return Merchandise Authorization (RMA) number. Write the RMA number on the outside of the box used to return the merchandise to Parallax. Please enclose the following along with the returned merchandise: your name, telephone number, shipping address, and a description of the problem. Parallax will return your product or its replacement using the same shipping method used to ship the product to Parallax.

14-DAY MONEY BACK GUARANTEE

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a full refund. Parallax will refund the purchase price of the product, excluding shipping/handling costs. This guarantee is void if the product has been altered or damaged. See the Warranty section above for instructions on returning a product to Parallax.

COPYRIGHTS AND TRADEMARKS

ISBN# 1-928982-03-4

DISCLAIMER OF LIABILITY

Parallax, Inc. is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of

equipment or property, or any costs of recovering, reprogramming, or reproducing any data stored in or used with Parallax products. Parallax is also not responsible for any personal damage, including that to life and health, resulting from use of any of our products. You take full responsibility for your BASIC Stamp application, no matter how life-threatening it may be.

WEB SITE AND DISCUSSION LISTS

Die hier aufgeführten Webseiten und Diskussionsforen sind durchwegs in englischer Sprache gehalten. Die Hinweise sind deshalb nicht übersetzt.

The Parallax web site (www.parallax.com) has many downloads, products, customer applications and on-line ordering for the components used in this text. We also maintain several e-mail discussion lists for people interested in using Parallax products. These lists are accessible from www.parallax.com via the Support → Discussion Groups menu. These are the lists that we operate:

BASIC Stamps – With over 2,500 subscribers, this list is widely utilized by engineers, hobbyists and students who share their BASIC Stamp projects and ask questions.
Stamps in Class – Created for educators *and* students, this list has 500 subscribers who discuss the use of the Stamps in Class curriculum in their courses. The list provides an opportunity for both students and educators to ask questions and get answers.
Parallax Educators – This focus group of 100 members consists exclusively of educators and those who contribute to the development of Stamps in Class. Parallax created this group to obtain feedback on our curricula and to provide a forum for educators to develop Teacher's Guides.
Parallax Translators – Consisting of less than 10 people, the purpose of this list is to provide a conduit between Parallax and those who translate our documentation to languages other than English. Parallax provides editable Word documents to our translating partners and attempts to time the translations to coordinate with our publications.
Toddler Robot – A customer created this discussion list to discuss applications and programming of the Parallax Toddler robot.
SX Tech – Discussion of programming the SX microcontroller with Parallax assembly language tools and 3rd party BASIC and C compilers. Approximately 600 members.
Javelin Stamp – Discussion of application and design using the Javelin Stamp, a Parallax module that is programmed using a subset of Sun Microsystems' Java® programming language. Approximately 250 members.
HexCrawler – Parallax technical support and engineering staff are charter members and welcome all aboard whether you've already mastered your HexCrawler or if you are considering a purchase.

ERRATA

Es ist trotz grosser Anstrengungen der Übersetzer nicht ausgeschlossen, dass der vorliegende Text noch Fehler enthält. Falls Sie einen solche finden, danken wir Ihnen für eine kurze Mitteilung an editor@parallax.com unter dem Betreff "Errata in German Translation of Robotics with the Boe-Bot".

While great effort is made to assure the accuracy of our texts, errors may still exist. If you find an error, please let us know by sending an email to editor@parallax.com. We continually strive to improve all of our educational materials and documentation, and frequently revise our texts. Occasionally, an Errata sheet with a list of known errors and corrections for a given text will be posted to our web site, www.parallax.com. Please check the individual product page's free downloads for an errata file.

Inhaltsverzeichnis

Einleitung

VORWORT

Roboter werden im Automobilbau, in der Gesundheits- Konsum- und Industriegüterproduktion ebenso verwendet, wie in Erkundungsfahrzeugen und natürlich in vielen Science Fiction Filmen. Beim Bau und der Programmierung eines Roboters geht es um eine Mischung aus Mechanik, Elektronik und Problemlösungstechnik. Was Sie mit dem Boe-Bot hier erleben entspricht echten Anwendungen von Robotersteuerungen, mit dem einzigen Unterschied von Grösse und Komplexität. Die hier verwendeten elektronischen Steuerungsprinzipien, Beispielprogramme und Schaltungen, sind jenen in industriellen Anwendungen, die von professionellen Ingenieuren entwickelt wurden, sehr ähnlich (und manchmal identisch mit solchen).

Das Wort “robot” erschien erstmals 1920 im tschechischen satirischen Stück *Rossum's Universal Robots* von Karel Capek. Die Roboter in diesem Stück verhielten sich menschenähnlich. Es scheint, dass seit diesem Zeitpunkt in vielen Science Fiction Romanen Roboter vorkamen, die sich gegen die Menschen auflehnen. Das änderte sich, als General Motors 1961 den ersten Roboter in einem Automobilwerk installierte. Diese automatischen Maschinen unterschieden sich radikal von den “menschenähnlichen” Robotern der Science Fiction Romane.

Das Ziel dieses Buches ist, bei den Studenten Interesse und Begeisterung für das Themengebiet Engineering, Mechatronik und Software Entwicklung zu wecken, indem sie selbst autonome Roboter entwerfen, konstruieren und programmieren. Mit einer Serie von Experimenten werden anhand des “Board-of-Education-Robot” (kurz: Boe-Bot) grundlegende Konzepte der Robotik eingeführt. Ein Beispiel eines Boe-Bot mit Infrarot-Hinderniserkennung, aufgebaut auf dem lötfreien Prototypingfeld, ist in Bild P-1 abgebildet. Die Experimente beginnen mit einer Einführung in das Gehirn von Boe-Bot, der BASIC Stamp, und führen dann weiter zu Konstruktion, Test und Kalibrierung des Boe-Bot. Danach werden Sie den Boe-Bot für grundlegende Bewegungen programmieren, und anschliessend Sensoren anbauen, die es dem Boe-Bot ermöglichen, aktiv auf seine Umgebung zu reagieren.

Figur P-1
Parallax' Boe-Bot™
Selbständiger Rad-Roboter.

ZIELPUBLIKUM

Das Lehrbuch *Robotics with the Boe-Bot* wurde für die Altersstufe ab 13 als Folgetext zu *What's a Microcontroller?* entwickelt. Wie bei allen Experimenten aus dem *Stamps in Class* Lehrgang lehrt auch diese Serie von Experimenten neue Techniken und Schaltungen mit minimaler Überlappung zwischen den Büchern. Die allgemeinen Themen, die in dieser Serie eingeführt werden, sind: Grundlegende programmgesteuerte Boe-Bot-Navigation, Navigation mit verschiedenen Sensoren, Navigation mit Feedback und verschiedenen Steuerungstechniken und Navigation unter Verwendung von programmierter Künstlicher Intelligenz. Die Einführung jedes Themas ist so aufgebaut, dass der Student zusammen mit eigenhändiger Erfahrung auch ein konzeptionelles Verständnis für das Thema entwickelt. Wer tiefer in die industrielle Technologie, Elektronik oder Robotik eindringen will, wird wohl erheblich von den ersten Experimenten mit diesen Themen profitieren.

SUPPORT- UND DISKUSSIONSGRUPPEN

Die folgenden Yahoo! Diskussionsforen (in Englisch) bieten weiteren Support für Interessierte. Die Foren sind über www.parallax.com unter Support / Discussion Groups erreichbar. Es folgt hier eine Beschreibung der verschiedenen Support- und Diskussionsgruppen.

Stamps In Class Group: Open to students, educators, and independent learners, this forum allows members to ask each other questions and share answers as they work through the activities, exercises and projects in this text.

Parallax Educator's Group: This moderated forum provides support for educators and welcomes feedback as we continue to develop our Stamps in Class curriculum. To join this group you must have proof of your status as an educator verified by Parallax. The Teacher's Guide for this text is available as a free download through this forum.

Educational Support: stampsinclass@parallax.com Contact the Parallax Stamps in Class Team directly if you are having difficulty subscribing to either of these Yahoo! Groups, or have questions about the material in this text, our Stamps in Class Curriculum, our Educator's Courses, or any of our educational services.

Educational Sales: sales@parallax.com Contact our Sales Team for information about educational discount pricing and classroom packs for our Stamps in Class kits and other selected products.

Technical Support: support@parallax.com Contact our Tech Support Team for general questions regarding the set-up and use of any of our hardware or software products.

DER STAMPS IN CLASS LEHRGANG

Der vorliegende Lehrtext kann ohne jede Vorkenntnis erfolgreich durchgearbeitet werden. Allerdings empfehlen wir das Buch *What's a Microcontroller?* als besten Einstieg in unseren Stamps in Class Lehrgang.

"*What's a Microcontroller?*", Student Guide, Version 2.2, Parallax Inc., 2004

Nach Abschluss dieses Einführungswerkes können Sie ihre Studien mit jedem beliebigen der Lehrbücher oder Bausätze fortsetzen, die unten aufgeführt sind. Alle Bücher sind gratis als Download auf www.parallax.com erhältlich. Unten sind die zum Zeitpunkt der Drucklegung dieses Textes aktuellen Versionen. Bitte prüfen Sie gegebenenfalls bei www.parallax.com oder www.stampsinclass.com ob neuere Ausgaben verfügbar sind; Wir bemühen uns sehr, unser pädagogisches Programm laufend zu verbessern.

Stamps in Class Student Guides:

Wir empfehlen, die Aktivitäten und Projekte der folgenden „Student Guides" gründlich durchzuarbeiten, um zu einer abgeschlossenen Einführung in den Entwurfsmethoden für moderne Geräte und Maschinen zu gelangen.

"*Applied Sensors*", Student Guide, Version 1.3, Parallax Inc., 2003
"*Basic Analog and Digital*", Student Guide, Version 1.3, Parallax Inc., 2004
"*Industrial Control*", Student Guide, Version 1.1, Parallax Inc., 1999

Weitere Robotics Bausätze:

Wenn sie den vorliegenden Text durchgearbeitet haben, sind sie für jeden der folgenden beiden Aufbautexte und Bausätze gut gerüstet:

"*Advanced Robotics: with the Toddler*", Student Guide, Version 1.2, Parallax Inc., 2003
"*SumoBot*" Manual, Version 2.0, Parallax Inc., 2004

Weitere Bausätze für die Ausbildung:

Elements of Digital Logic, Understanding Signals und *Experiments with Renewable Energy* fokussieren stärker auf Themen der Elektronik, während *StampWorks* eine breite Auswahl verschiedener Projekte anbietet. Diese sind vor allem für vielseitig interessierte Hobby-Elektroniker, Erfinder und Produktentwickler nützlich, welche unterschiedliche Projekte ausprobieren wollen.

"*Elements of Digital Logic*", Student Guide, Version 1.0, Parallax Inc., 2003
"*Experiments with Renewable Energy*", Student Guide, Version 1.0, Parallax Inc., 2004
"*StampWorks*", Manual, Version 1.2, Parallax Inc., 2000-2001
"*Understanding Signals*", Student Guide, Version 1.0, Parallax Inc., 2003

Hinweis: Die vorstehend besprochenen Texte werden von Parallax grundsätzlich in englischer Sprache entwickelt und publiziert. Bitte prüfen Sie bei Bedarf, ob bzw. welche deutschen Übersetzungen dazu vorliegen. Kontaktieren Sie allenfalls den einschlägigen Fachhandel in den deutschsprachigen Ländern.

Literaturhinweise

Das wegweisende Handbuch: *The Basic Stamp Manual* liegt im Original zur Zeit nur in englisch vor. Man kann es gratis von der Parallax – Homepage downloaden. Für den Anwender deutscher Sprache gibt es aber von Claus Kühnel/Klaus Zahnert das deutsche Buch ***"Basic Stamp 2 Neue Eigenschaften – neue Projekte"*** (ISBN 3-907857-02-X). Hier findet man neben einer Fülle von Tipps und Ideen auch eine vollständige deutsche Befehlsreferenz.

The *BASIC Stamp Manual* book is an essential reference for all Stamps in Class Student Guides. It is packed with information on the BASIC Stamp microcontrollers, the Board of Education and our other carrier boards, the BASIC Stamp Editor, and our PBASIC programming language.

"*BASIC Stamp Manual*", Version 2.0c, Parallax Inc., 2000

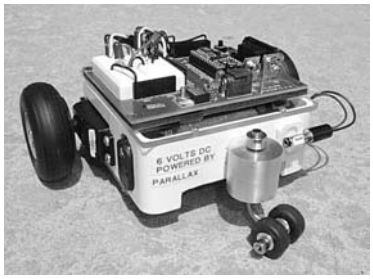
ÜBERSETZUNGEN

Der folgende Abschnitt richtet sich an Leute, welche Parallax-Dokumente in andere Sprachen übersetzen wollen:

Parallax educational texts may be translated to other languages with our permission (e-mail stampsinclass@parallax.com). If you plan on doing any translations please contact us so we can provide the correctly-formatted MS Word documents, images, etc. We also maintain a discussion group for Parallax translators that you may join. It's called the Parallax Translators Yahoo! Group, and directions for finding it are included on the inside cover of this text. See section entitled: WEB SITE AND DISCUSSION LISTS after the Title page.

DANKSAGUNG

Dr. Chuck Schoeffler verfasste in Zusammenarbeit mit Parallax Teile der Version 1.2 dieses Textes. Dr. Schoeffler war zu dieser Zeit Professor an der University of Idahoe, Abteilung Industrial Technology Education. Er entwarf auch den unten abgebildeten ursprünglichen Board of Education Robot (Boe-Bot) und weitere ähnliche Robotvarianten mit vielen besonderen Funktionen. Nach mehreren Überarbeitungen wurde Dr. Schoefflers Design als Basis für den Parallax Boe-Bot gewählt, auf dem der vorliegende Text basiert. Russ Miller von Parallax entwarf dann den Boe-Bot auf der Basis dieses Prototyps.



Figur P-2
Original Boe-Bot Prototyp

Andrew Lindsay, Chef-Robotiker von Parallax, hat inzwischen dieses Buch und die einzelnen Aktivitäten überarbeitet. Er hatte dabei drei Ziele vor Augen: Erstens sollten alle Aktivitäten in diesem Text von sorgfältig verfassten Vorgehens-Beschreibungen begleitet werden. Zweitens sollte der Leser und Student in jedem Kapitel neuen Schaltungen und neuen Konzepte der Programmierung, des Engineering und der Robotik begegnen. Und drittens sollte sichergestellt werden, dass die Experimente mit einer hohen Erfolgswahrscheinlichkeit durchgeführt werden können, indem die aktuellsten Parallax-Technologie eingesetzt wird. Für die vorliegende Version 2.0, die aktuellste Technologie ist das Board of Education Rev. C und das BASIC Stamp HomeWork Board.

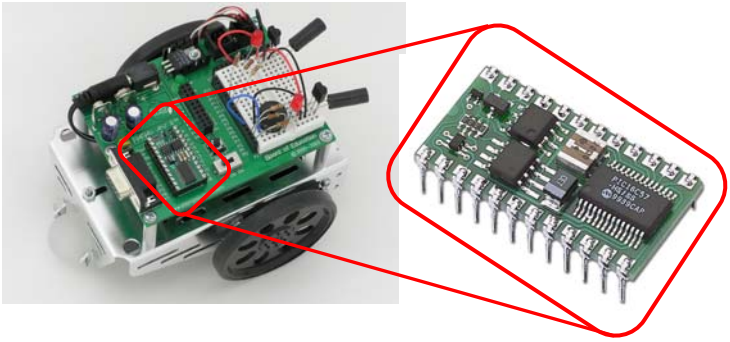
Ein besonderer Dank geht an Branden Gunn, der im Sommer 2000 bei Parallax eine Internship absolvierte, für die Illustrationen zur Version 1.3, und an Dale Kretzer für die redaktionelle Überarbeitung, welche in Version 1.4 integriert wurde. Ein Dankeschön ebenfalls an die Teilnehmer der Stamps in Class e-group für ihre Beiträge: Richard Breen, Robert Ang, Dwayne Tunnell, Marc Pierloz, and Nagi Babu. Sie alle haben mit Fehlerkorrekturen, Verbesserungsvorschlägen und neuem Material zur Version 1.4 beigetragen. Danke den Studenten Laura Wong und Rob Gerber für Ihren Input zur Version 1.5. Ein spezieller Dank gilt den Mitarbeitern von Parallax. Jeder von Ihnen hat in der einen oder anderen Weise zum Erfolg des Stamps in Class Programmes beigetragen.

Version 2.0 dieses Student Guide ist Resultat einer grundlegenden inhaltlichen und textlichen Überarbeitung, bei der neue Aktivitäten sowie Unterstützung für PBASIC 2.5 und das BASIC Stamp HomeWork Board eingeführt wurden. Diese Überarbeitung wäre ohne die Unterstützung der folgenden Leute nicht möglich gewesen. Seitens Parallax: Andy Lindsay – Autor, Rich Allred – technische Illustrationen, Stephanie Lindsay – technische Redaktion, Kris Magri – Reviewer und Robotik Guru. Seitens Stamps in Class die Referenten Robert Ang und Sid Weaver.

Wenn Sie Vorschläge haben, glauben, einen Fehler gefunden zu haben oder sonst wie zu Aktivitäten oder Kapiteln für weitere Versionen von *Robotics with the Boe-Bot* beitragen möchten, kontaktieren Sie uns in englisch unter stampsinclass@parallax.com. Abonnieren Sie die StampsInClass Diskussionsgruppe bei Parallax. Für Beiträge zu *Robotics with the Boe-Bot* winken laufend aktualisierte Anerkennungsprämien in Form von gratis Parallax-Produkten. Konsultieren Sie den Abschnitt “Web Site und Diskussionsforen” in der Einleitung.

Kapitel 1: Das Gehirn Ihres Boe-Bot

Parallax' Boe-Bot™ Roboter steht im Zentrum der Aktivitäten, Projekte und Wettbewerbe in diesem Buch. Ein Boe-Bot und eine Nahaufnahme seines programmierbaren BASIC Stamp® 2 Mikrocontroller-Hirns werden in Figur 1-1 gezeigt. Das BASIC Stamp 2 Modul ist gleichzeitig leistungsfähig und einfach zu benutzen, vor allem mit einem Roboter.



Figur 1-1 BASIC Stamp® 2 Modul auf einem Boe-Bot™ Roboter.

Im folgenden Text werden Sie einfache Programme schreiben, mit denen Sie der BASIC Stamp und Ihrem Boe-Bot vier grundlegenden Roboteraufgaben beibringen:

1. Sensoren zu überwachen, um die Umgebung zu erkennen
2. Entscheidungen zu treffen basierend auf dem Wahrgenommenen
3. Seine Bewegung zu steuern (mit den Motoren an den Rädern)
4. Information auszutauschen mit seinem Robotiker (das sind Sie!)

Die Programmiersprache die wir dazu benutzen heisst **PBASIC**, das ist eine Abkürzung für:

- Parallax – Die Firma die BASIC Stamp Mikrocontroller erfunden hat und herstellt.
- Beginners – Gemacht für Einsteiger um Computer programmieren zu lernen.
- All-purpose – Zweckmässig und nützlich für verschiedenste Arten von Aufgaben.
- Symbolic – Verwendung von Symbolen (Begriffe, die Englische Wörder ähneln)
- Instruction – Um einem Computer zu befehlen, wie die Aufgaben zu lösen sind
- Code – In Begriffen, die Sie und der Computer verstehen.

Was ist ein Mikrocontroller? Das ist ein programmierbarer Chip, der in Ihre digitale Armbanduhr, Handy, Taschenrechner, Funkuhr etc. eingebaut ist. In diesen Geräten wurde der Mikrocontroller programmiert zu merken, wenn Sie eine Taste drücken, Piepstöne zu produzieren und die Digitalanzeige zu steuern. Sie werden ebenso in Maschinen, Autos, Unterseeboote und Raumschiffe eingebaut, weil sie dafür programmiert werden können, Sensordaten zu lesen, Entscheidungen zu treffen und das Zusammenwirken von Einrichtungen zu orchestrieren, welche bewegliche Teile steuern.

Das Buch *What's a Microcontroller?* Student Guide wird als erster Text für Einsteiger empfohlen. Es ist voller Beispiele, wie man Mikrocontroller benutzt, und wie man die BASIC Stamp zum Gehirn seiner eigenen Erfindungen macht. Man kann es gratis von www.parallax.com herunterladen (in Englisch).

HARDWARE UND SOFTWARE

Mit der Programmierung einer BASIC Stamp anzufangen läuft etwa so, wie Sie einen neuen PC oder Laptop in Betrieb nehmen. Das erste, was die meisten Leute tun, wenn sie einen neuen Computer haben ist, ihn auszupacken, einzustecken, etwas Software zu installieren und auszuprobieren und vielleicht sogar ein paar eigene Programmzeilen in einer Programmiersprache zu schreiben. Wenn Sie den BASIC Stamp Mikrocontroller zum ersten Mal einsetzen, werden Sie etwa dieselben Dinge tun. Dieses Kapitel zeigt wie Sie die BASIC Stamp Programmierung zum Laufen bringen:

- Finden und Installieren der Programmierungs- – Software
- Anschluss der BASIC Stamp an die Batterie-Stromversorgung damit sie läuft
- Anschluss des BASIC Stamp Moduls an den PC für die Programmierung.
- Schreiben der ersten paar PBASIC Programme
- Abtrennen der Stromversorgung wenn Sie fertig sind.

Wenn Sie in der Schule sind, ist die BASIC Stamp vielleicht bereits für Sie aufgebaut. In diesem Fall hat Ihr Lehrer möglicherweise andere Anweisungen für Sie. Sonst wird Sie dieses Kapitel durch alle Schritte führen, um Ihren neuen BASIC Stamp Computer zum Laufen zu bringen.

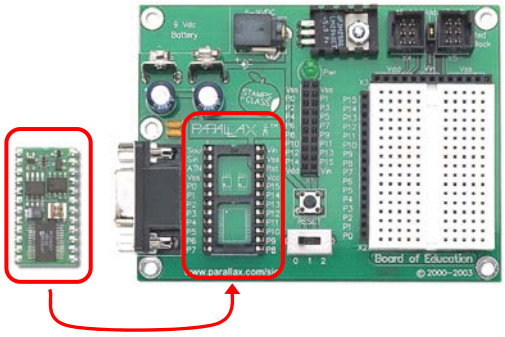
Sowohl dieses Buch als auch *What's a Microcontroller?* enthalten im Kapitel 1 die Instruktionen, wie man die BASIC Stamp Hardware und Software aufsetzt. Diese Anweisungen sind in beiden Büchern praktisch identisch.

√ Wenn Sie bereits das Lehrbuch *What's a Microcontroller?* durchgearbeitet haben, gehen Sie gleich zum nächsten Kapitel.

√ Auch wenn Sie Ihre BASIC Stamp und Ihr Board of Education oder Ihr BASIC Stamp HomeWork Board bereits gut kennen, können Sie dieses Kapitel überspringen.

Einführung in die BASIC Stamp und das Board of Education

Ein BASIC Stamp 2 Modul und eine Board of Education® Trägerplatine sehen Sie in Figur 1-2. Wie erwähnt ist ein BASIC Stamp Modul wie ein sehr kleiner Computer. Diesen sehr kleinen Computer steckt man in das Board of Education, das man als "*carrier board*" (Trägerplatine) bezeichnet. Wie Sie gleich sehen werden macht es das Board of Education einfach, eine Stromversorgung und ein Serielles Kabel an die BASIC Stamp anzuschliessen. In späteren Schritten werden Sie auch sehen, wie das Board of Education den Aufbau von Schaltungen und ihren Anschluss an die BASIC Stamp vereinfacht.



Figur 1-2
BASIC Stamp® 2 Modul (links)

Board of Education® Trägerplatine (rechts)

Einführung in das BASIC Stamp HomeWork Board

Das BASIC Stamp® HomeWork Board™ als Projektplattform ist in Figur 1-3 unten gezeigt. Dieses Board ist wie ein Board of Education mit eingebautem BASIC Stamp 2 Mikrocontroller. Sie können als Basis für die Projekte in diesem Text entweder ein BASIC Stamp 2 Modul auf einer Board of Education Trägerplatine verwenden, oder das BASIC Stamp HomeWork Board. Denken Sie daran, die Anweisungen für die jeweilige Projektplattform zu befolgen, da sich diese in einigen wenigen Stellen unterscheiden.

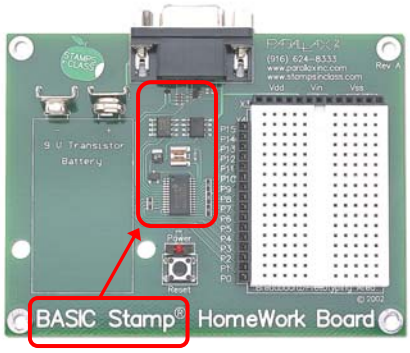


Figure 1-3
BASIC Stamp®
HomeWork Board™
Projektplattform.

Was ist der Unterschied?

Bei Verwendung eines Board of Education carrier board und eines BASIC Stamp Moduls gibt es einige zusätzliche Optionen, wie zum Beispiel Steckplätze zum Einstecken von Servomotoren, Wahl der Stromversorgung für die Servos und ein nützlicher 3fach-Schalter zur selektiven Stromversorgung der verschiedenen Systemteile. Das BASIC Stamp 2 Modul ist steckbar und kann ausgetauscht werden.

Das BASIC Stamp HomeWork Board hat keine Servo-Ports, keine Buchse für eine externe Stromversorgung und keinen Hauptschalter. Dafür kostet es auch weniger. Sie müssen die Servoanschlüsse selber aufbauen und die Stromversorgung durch Ausziehen des Steckers abschalten, oder sich einen eigenen Hauptschalter anbauen. Der BASIC Stamp 2 Mikrocontroller ist fest auf die Trägerplatine gelötet und jeder I/O-Pin wird durch einen oberflächenmontierten 220 Ω Widerstand geschützt.

Siehe dazu auch: Appendix B: BASIC Stamp and Carrier Board Components and Features

AKTIVITÄT 1: BESCHAFFUNG DER SOFTWARE

Der BASIC Stamp Editor (Version 2.0 oder höher) ist die Software, die Sie in den meisten Aktivitäten und Projekten in diesem Buch verwenden. Diese Applikation ermöglicht Ihnen, auf Ihrem Computer Programme zu schreiben und diese in das BASIC Stamp Gehirn Ihres Boe-Bot zu laden. Sie zeigt auch auf Ihrem Computerdisplay Meldungen der BASIC Stamp an, wodurch der Boe-Bot eine Möglichkeit hat, Ihnen, seinem Robotiker, zu berichten, was er tut und spürt.

Der **BASIC Stamp Editor ist gratis Software.** Es gibt zwei einfache Möglichkeiten, ihn zu bekommen:

- Download aus dem Internet: Suchen Sie den "BASIC Stamp Windows Editor Version 2.0…" auf www.parallax.com → Downloads → BASIC Stamp Software.
- Auf der Parallax CD: Folgen Sie dem Software Link auf der Welcome Seite. Überprüfen Sie, ob das Datum auf der CD jünger ist als April 2003.

In Eile? Besorgen Sie sich eine Kopie des BASIC Stamp Windows Editor Version 2.0 (oder höher) und installieren Sie diese auf Ihrem PC oder Laptop. Dann gehen Sie direkt zu Aktivität 3: .

Wenn Sie ein paar Fragen dazu haben, kann Aktivität 1: Beschaffung der Software als Schritt-für-Schritt Anleitung genutzt werden, um die Software zu beschaffen, und Aktivität 2 dient als Referenz für die Installation der Software auf Ihrem PC oder Laptop.

Anforderungen an das Computersystem

Sie benötigen entweder einen PC oder einen Laptop, um die BASIC Stamp Editor Software laufen zu lassen. Der Start ist am einfachsten, wenn Ihr Rechner folgende Eigenschaften aufweist:

- Windows 95 oder höheres Betriebssystem
- Serielle Schnittstelle oder USB Port (USB 1.1 genügt)
- Ein CD-ROM Laufwerk, Zugang zum Internet oder beides

USB Port Adapter: Wenn Ihr Computer nur USB Ports hat, benötigen Sie einen USB nach Seriell Adapter. . Vergleichen Sie dazu die Information auf Seite 14 für weitere Einzelheiten.

Download der Software aus dem Internet

Es ist ganz einfach, die Software des BASIC Stamp Editor von der Parallax Website herunterzuladen. Die in Figur 1-4 gezeigte Website kann anders aussehen als die Seite, die Sie erhalten, wenn Sie im Internet Parallax besuchen. Trotzdem werden sich die Schritte für den Software-Download nicht wesentlich unterscheiden.

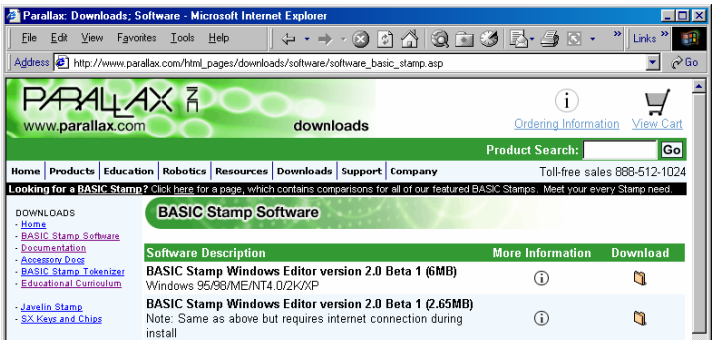
√ Verwenden Sie einen Browser und gehen Sie zu www.parallax.com (wie in Figur 1-4).
√ Zeigen Sie auf das Downloads Menu um die Optionen anzuzeigen.
√ Klicken Sie auf den BASIC Stamp Software Link.

Figur 1-4
Die Parallax Website:

www.parallax.com

√ Wenn Sie auf die BASIC Stamp Software Seite kommen, suchen Sie einen BASIC Stamp Windows Editor Download mit Versionsnummer 2.0 oder höher.

√ Klicken Sie das Download Icon. In Figur 1-5 sieht das Download Icon aus wie ein Dateiordner, rechts vom Beschreibungstext: "BASIC Stamp Windows Editor Version 2.0 Beta 1 (6MB)".



Figur 1-5
Die Parallax Website Downloads Seite

√ Wenn das File Download Fenster aus Figur 1-6 erscheint, wählen Sie: Save this program to disk.

√ Klicken Sie den OK Knopf.

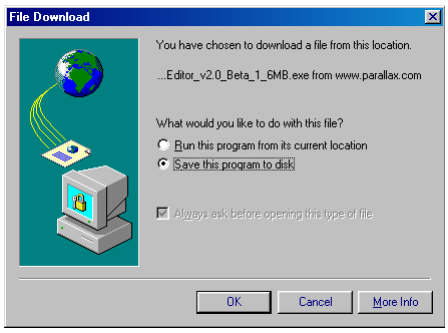


Figure 1-6
File Download Window

Figur 1-7 zeigt das Save As (speichern als) Fenster das als nächstes erscheint. Sie können das Save in (Speichern unter) Feld verwenden, um für die Datei einen passenden Platz auf Ihrem Harddisk zu finden.

√ Nachdem Sie den Speicherort gewählt haben, klicken Sie den Save Button.

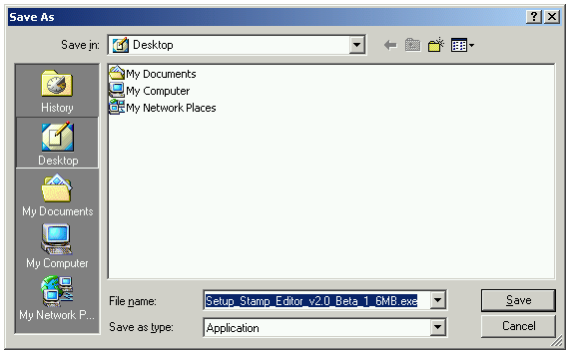


Figure 1-7
Save As Window

Wählen Sie einen Speicherort für die Datei

√ Warten Sie, bis das BASIC Stamp Editor Installationsprogramm übertragen wurde (siehe Figur 1-8). Je nach Art Ihrer Verbindung zum Internet kann das einige Zeit dauern.

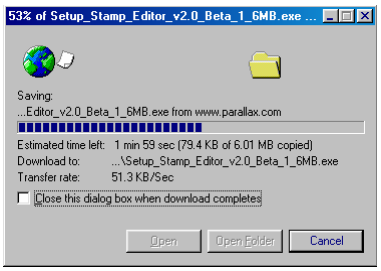


Figure 1-8
Download
Progress Window

√ Wenn der Download abgeschlossen ist, lassen Sie das Fenster aus Figur 1-9 offen und gehen Sie direkt zum nächsten Abschnitt.

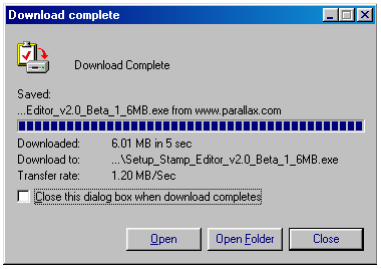


Figure 1-9
Download
Complete
Window

Weitere gratis Download-Angebote von der Parallax Website:

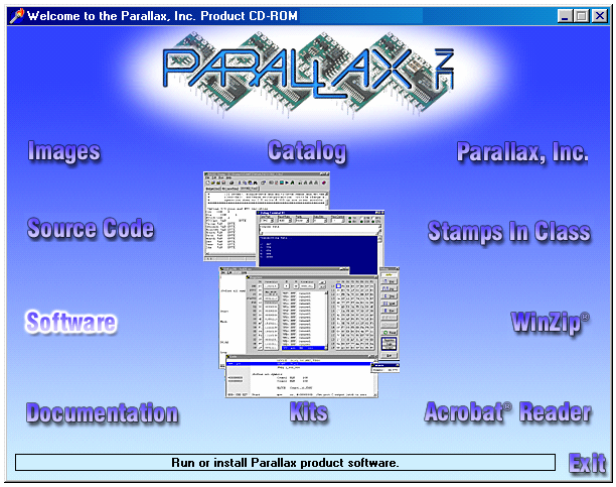
- Dieser Text (in englisch) und andere Stamps in Class Texte
- Robot Videos
- Weitere Gratissoftware
- Hunderte von Applikationen und Experimenten zum Ausprobieren!

Die Software auf der Parallax CD finden

Sie können den BASIC Stamp Editor auch von der Parallax CD installieren, aber die CD muss neuer sein als April 2003, damit Sie die mit den Beispielen in diesem Text kompatible Version des BASIC Stamp Editor haben. Jahr und Monat der Parallax CD sind auf der Vorderseite der CD aufgedruckt.

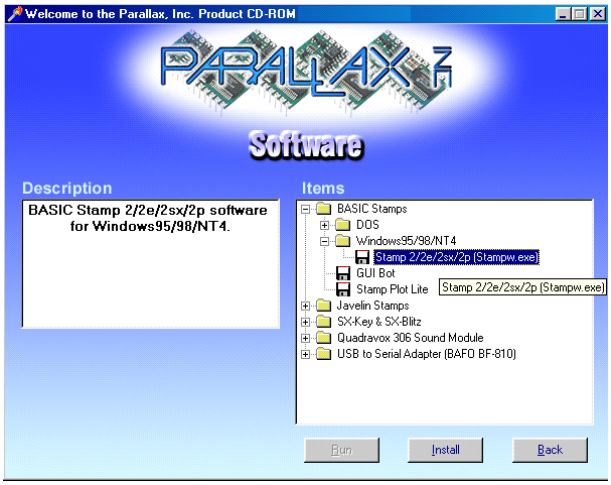
√ Legen Sie die Parallax CD ins CD Laufwerk Ihres PC. Der Browser der Parallax CD heisst Welcome Applikation. Sie ist in Figur 1-10 abgebildet und sollte selbständig starten, sobald sie die CD in den PC eingelegt haben.

√ Wenn die Welcome Applikation nicht automatisch startet, doppelklicken Sie Arbeitsplatz, dann Ihr CD Laufwerk und dann Welcome.
√ Klicken Sie auf den Software Link wie in Figur 1-10.



Figur 1-10
Der Parallax CD Browser

√ Klicken Sie auf das + neben dem BASIC Stamps Ordner in Figur 1-11.
√ Klicken Sie auf das + neben dem Windows Ordner.
√ Klicken Sie auf das Disketten-Icon "Stamp 2/2e/2sx/2p/2pe (stampw.exe)".
√ Folgen Sie den Anweisungen von Aktivität 2



Figur 1-11
Der Parallax CD Browser

Wählen Sie das BASIC Stamp Editor Installationsprogramm aus der Software Seite.

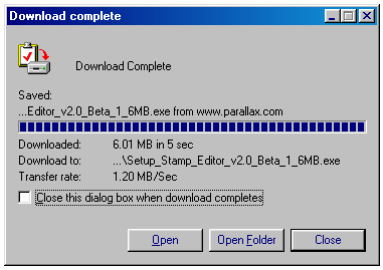
	**Die gratis Downloadangebote der Parallax Website sind auf der Parallax CD enthalten**, aber nur bis zu dem Datum, als die CD erstellt wurde. Das Datum auf der CD nennt das Erstellungsdatum. Wenn Ihre CD nur ein oder zwei Monate alt ist, haben Sie höchstwahrscheinlich den aktuellsten Stand. Wenn es eine ältere CD ist, sollten Sie vielleicht eine neue CD bestellen oder die benötigten Dateien frisch von der Parallax Website herunterladen.

AKTIVITÄT 2: INSTALLATION DER SOFTWARE

Sie haben nun den BASIC Stamp Editor Installer entweder von der Parallax Website heruntergeladen, oder ihn auf der Parallax CD gefunden. Dann können wir nun den BASIC Stamp Editor Installer starten.

Installation der Software – Schritt für Schritt

√ Wenn Sie den BASIC Stamp Editor Installer aus dem Internet heruntergeladen haben, klicken Sie den Open Button auf dem Download Complete Fenster wie in Figur 1-12.



Figur 1-12
Download Complete Fenster

Wenn sie vom Abschnitt Download hierher gesprungen sind, klicken Sie den Open Button.

√ Wenn Sie die Software auf der Parallax CD gefunden haben, klicken Sie den Install Button gemäss Figur 1-13.

Figure 1-13
Der Parallax CD Browser

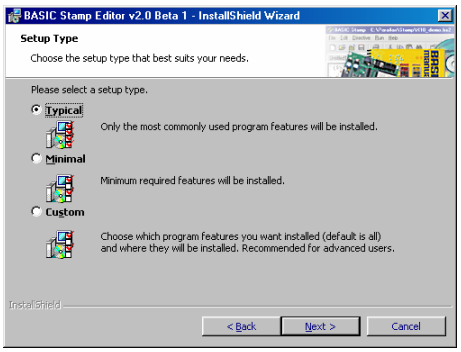
Install Button am unteren Rand des Fensters.

√ Wenn sich das BASIC Stamp Editor…InstallShield Wizard Fenster öffnet, klicken Sie den Next Button wie in Figur 1-14.

Figur 1-14
InstallShield Wizard für den BASIC Stamp Editor

√ Wählen Sie Typical als Setup Typ aus wie in Figur 1-15.

√ Klicken Sie den Next Button.



Figur 1-15
Setup Typ

√ Wenn der InstallShield Wizard meldet, er sei "Ready to Install the Program", klicken Sie den Install Button wie in Figur 1-16.

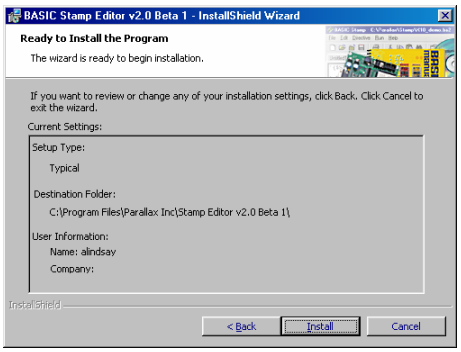
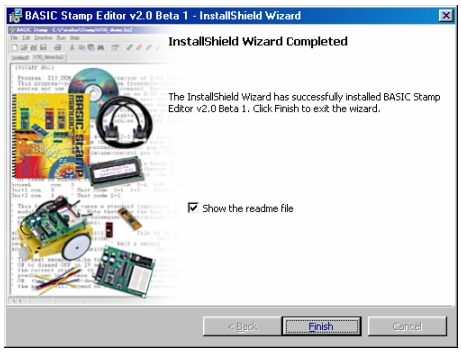


Figure 1-16
Ready to Install

Klicken Sie den Install Button.

√ Wenn das InstallShield Wizard Fenster meldet "InstallShield Wizard Completed" wie in Figur 1-17, klicken Sie Finish.



Figur 1-17
InstallShield Wizard abgeschlossen

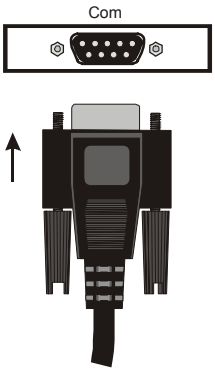
AKTIVITÄT 3: EINRICHTEN DER HARDWARE UND SYSTEMTEST

Die BASIC Stamp benötigt Strom, damit sie läuft. Sie muss auch an einen PC angeschlossen werden, damit man sie programmieren kann. Sobald Sie diese Anschlüsse gemacht haben, können sie den BASIC Stamp Editor benutzen, um das System zu testen. Der folgende Arbeitsschritt zeigt, wie das geht.

Einrichten des seriellen Computerkabels

Das Board of Education oder BASIC Stamp HomeWork Board wird entweder mit einem seriellen Kabel (*serial cable*) oder mit einem USB-nach-Seriell-Adapter (*UBS to serial adapter*) an Ihren PC oder Laptop angeschlossen.

√ Wenn Sie ein serielles Kabel verwenden, stecken Sie dieses in einen freien COM Port Ihres Computers ein (siehe Figur 1-18).



Figur 1-18
PC oder Laptop COM Port

Stecken Sie das serielle Kabel in einen freien COM-Port an Ihrem PC oder Laptop.

√ Wenn Sie einen USB-nach-Seriell-Adapter verwenden, halten Sie sich für die Installation bitte an die Instruktionen, die vom entsprechenden Hersteller mit dem Produkt geliefert werden.

US232B/LC USB to Serial Adapter von FTDI:

Zum Zeitpunkt der Drucklegung dieses Textes empfiehlt Parallax die Verwendung des US232B/LC *USB to Serial Adapter* von Future Technology Devices International für den Gebrauch mit Parallax-Produkten. Der US232B/LC wird in den USA mit der in Figur 1-19 gezeigten Hardware und einer Mini-CD ROM mit Treibern für die Verwendung unter den verschiedenen Betriebssystemen, inkl. Microsoft Windows®, geliefert.

In Europa sind vergleichbare Produkte im einschlägigen Fachhandel erhältlich.

US232B/LC Treiber Software Downloads: Die Software-Treiber und andere Produktinformationen können bezogen werden bei: http://www.ftdichip.com/FT232.htm.

Figur 1-19

US232B/LC USB nach Seriell Adapter von FTDI

Dieser Adapter hat die Parallax Stock# 800-00030. Er wird mit einer Software-CD ausgeliefert (nicht im Bild).

Nachdem nun das Programmierungskabel an Ihrem Computer angeschlossen ist, wird es Zeit, die Hardware zusammenzubauen.

√ Wenn Sie eine BASIC Stamp und ein Board of Education haben, befolgen Sie bitte die Instruktionen im nächsten Abschnitt.

√ Wenn Sie ein BASIC Stamp HomeWork board haben, gehen Sie bitte direkt zu den Instruktionen für den Anschluss des BASIC Stamp HomeWork Board auf Seite 19.

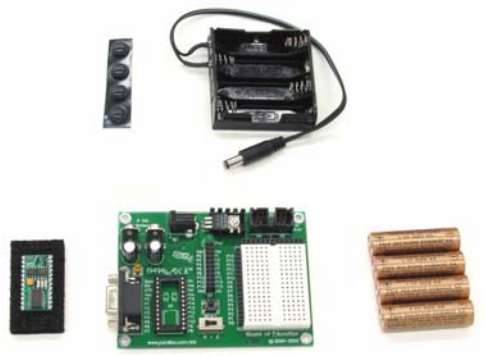
√ Wenn Ihre Ausrüstung bereits zusammengesetzt ist, gehen Sie bitte direkt zum Abschnitt Testing for Communication auf Seite 21.

Anschluss des Board of Education

Wenn Sie eine BASIC Stamp und ein Board of Education haben, so sehen Sie in Figur 1-20 die benötigten Bauteile.

Benötigte Hardware

(1) Streifen mit vier Gummifüssen
(1) Batteriehalter für 4 Batterien AA
(1) BASIC Stamp 2
(1) Board of Education
(4) Neue AA Alkalibatterien (nicht im Bausatz enthalten)



Figur 1-20

Für den Anfang: Hardware für die BASIC Stamp und das Board of Education

Anschluss der Hardware

Die Gummifüsse sehen sie in Figur 1-21. Diese werden an der Unterseite Ihres Board of Education angeklebt. Das Board of Education hat aufgedruckte Kreise auf der Unterseite, welche die genaue Platzierung jedes Gummifusses vorgeben.

√ Nehmen Sie jeden Gummifuss vom Trägerstreifen und kleben Sie ihn auf die Unterseite des Board of Education.

Figur 1-21
Gummifüsse (links) Auf die Unterseite des Board of Education geklebt (rechts)

Das Board of Education Rev C hat einen Dreifachschalter (siehe Figur 1-22). Position-0 schaltet die Stromversorgung für das Board of Education vollständig ab. Egal ob sie Batterien oder eine Stromversorgung angeschlossen haben oder nicht, wenn der 3er-Schalter auf 0 steht, ist das Gerät ausgeschaltet.

√ Schalten sie den Dreifachschalter (*3-position-switch*) auf dem Board of Education auf Position 0.

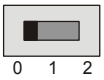


Figure 1-22
3-position Switch

Set to position-0 to turn off the power.

Nur das Board of Education Rev C hat einen 3-Positionen-Schalter.

Wenn Sie ein Board of Education Rev A or B haben:

- Wenn Sie aufgefordert werden, den Dreifachschalter in Stellung 0 zu bringen, schalten Sie die Stromversorgung durch Ausziehen des Batteriepacks aus (das Umgekehrte von Figur 1-24, Schritt 3).
- Wenn Sie aufgefordert werden, den Dreifachschalter in Stellung 1 oder 2 zu bringen, stecken Sie das Batteriepack wie in Figur 1-24, Schritt 3 gezeigt ein.

√ Legen Sie die Batterien ein. (Figur 1-23). Überprüfen Sie, dass das negative (flache) Ende jeder Batterie guten Kontakt mit der Feder des Batteriehalters hat. Die korrekte Polarität jeder Batterie ist auch im Batteriegehäuse eingeprägt.

Figur 1-23
Batteriepack

Eingeprägte Polaritätskennzeichnungen (links) und mit korrekt eingesetzten Batterien (rechts).

√ Wenn Ihre BASIC Stamp nicht bereits in Ihrem Board of Education steckt, fügen Sie es vorsichtig in den dafür vorgesehenen Sockel ein (Figur 1-24, Schritt 1). Dabei muss die Kerbe "*reference notch*" an der oberen Kante der BASIC Stamp zur Kerbe am Sockel zeigen, damit Sie sie nicht verkehrt herum einstecken. Überprüfen Sie, dass die Pins sauber auf die Buchsen des Sockels ausgerichtet sind, und drücken Sie dann die BASIC Stamp hinein.

√ Stecken Sie das serielle Kabel wie in Figur 1-24, Schritt 2 gezeigt in das Board of Education.

√ Stecken Sie das Batteriepack in die 6-9V = (Gleichspannung) Batteriebuchse gemäss Figur 1-24, Schritt 3.

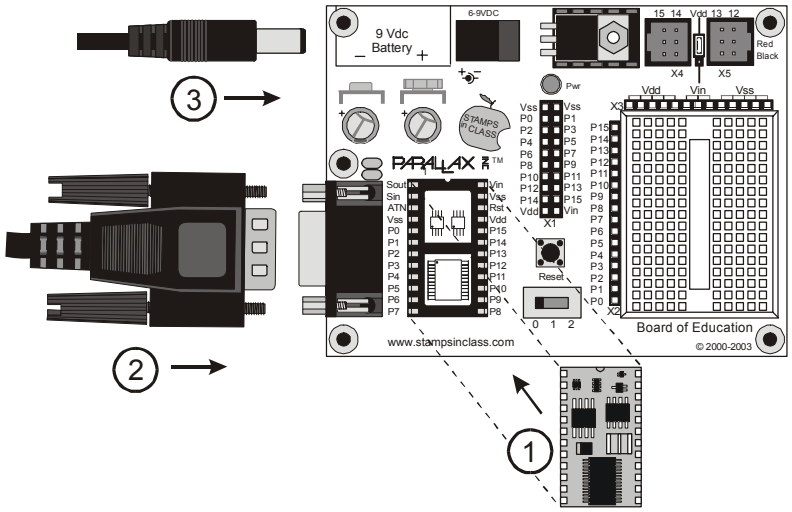
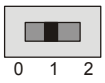


Figure 1-24
Board of Education, BASIC Stamp, Batterie, und serielles Kabel

Schliessen Sie die Teile in der angegebenen Reihenfolge an.

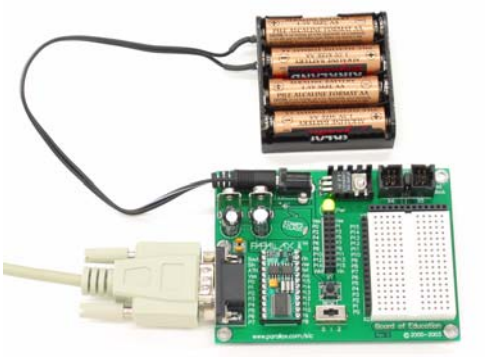
√ Schalten Sie nun den Dreifachschalter von 0 auf 1 um die Stromversorgung wieder einzuschalten.



Figur 1-25
Dreifachschalter

In Stellung 1um den Strom wieder anzuschalten.

√ Das grüne Licht mit der Bezeichnung “Pwr” (“Power”= Strom) am Board of Education sollte nun leuchten.



Figur 1-26
BASIC Stamp und Board of Education angeschlossen und bereit für die Programmierung

√ Überspringen Sie das Folgende und gehen Sie direkt zu Abschnitt Testing for Communication auf Seite 21.

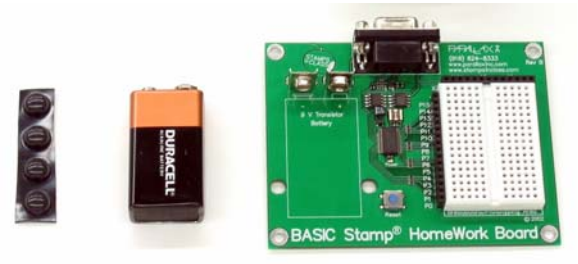
BASIC Stamp HomeWork Board Anschluss Anleitung

Dieser Abschnitt zeigt Ihnen, wie Sie die Basic Stamp und eine Batterie bzw. ein Netzteil mit einem BASIC Stamp Home Work Board anschliessen.

Benötigte Hardware

√ Nehmen Sie folgende Teile aus dem Kit Figur 1-27

(1) Streifen mit vier Gummifüssen
(1) Neue 9 V Batterie (nicht beinhaltet)
(1) BASIC Stamp HomeWork Board

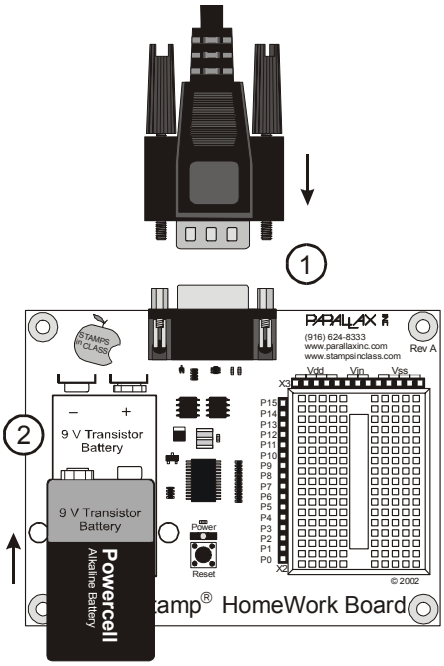


Figur 1-27
Für den Anfang: Hardware für das BASIC Stamp HomeWork Board

√ Nehmen Sie jeden Gummifuss vom Trägerstreifen und kleben Sie ihn auf die Unterseite des HomeWork Board jeweils neben das Loch in den Ecken wie in Figur 1-28 gezeigt.

Figur 1-28
Gummifüsse

√ Verbinden Sie das serielle Kabel und stecken Sie die Batterie an das HomeWork Board (Figur 1-29, Schritte 1 und 2).



Figur 1-29
HomeWork Board und serielles Kabel

Verbinden Sie das serielle Kabel und stecken Sie die Batterie an das HomeWork Board.

Figur 1-30 zeigt das BASIC Stamp HomeWork Board mit dem seriellen Kabel verbunden und Batterie eingesteckt. Beachten Sie, dass nach Anschluss der Batterie das grüne Pwr

Licht nicht angeht; es wird nur leuchten wenn ein Programm läuft. Jetzt sind Sie bereit, die Programmierverbindung zwischen BASIC Stamp und Ihrem PC/Laptop zu testen.

Figur 1-30
BASIC Stamp HomeWork Board bereit zur Programmierung

Testing for Communication
Verbindungstest

√ Starten Sie zuerst das Programm auf dem PC, indem Sie auf das BASIC Stamp Editor Symbol (Icon) auf Ihrem Desktop (doppel-)klicken. Es sollte etwa so aussehen wie in Figur 1-31.

Figur 1-31
BASIC Stamp Editor Shortcut

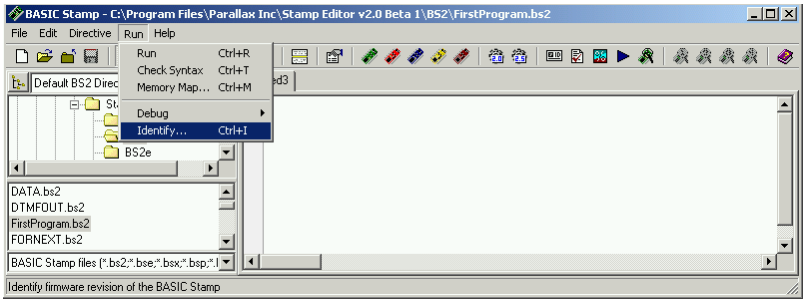
Suchen Sie einen Shortcut wie diesen auf dem Desktop Ihres Computers.

Sie können auch das Windows Start Menu verwenden um den BASIC Stamp Editor zu starten. Klicken Sie dazu auf den Windows *Start* Knopf, wählen Sie *Programme → Parallax, Inc. → Stamp Editor 2…*, dann klicken sie auf die *BASIC Stamp Editor* Icon.

Ihr BASIC Stamp Editor Fenster sollte etwa so aussehen wie das in Figur 1-32.

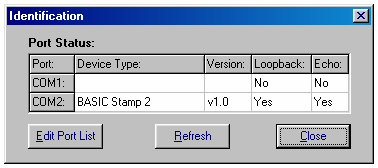
Beim ersten Start des BASIC Stamp Editor können diverse Meldungen und eine Liste der gefundenen COM-Ports erscheinen.

√ Um zu überprüfen, dass Ihre BASIC Stamp korrekt mit Ihrem Computer kommuniziert, klicken sie im Menü Run (Los!) die Auswahl Identify (Identifiziere).



Figur 1-32
BASIC Stamp Editor

Es erscheint ein Identifikationsfenster ähnlich wie das in Figur 1-33. Das abgebildete Beispiel zeigt, dass der Editor eine BASIC Stamp 2 an COM2 entdeckt hat.



Figur 1-33
Identification Window

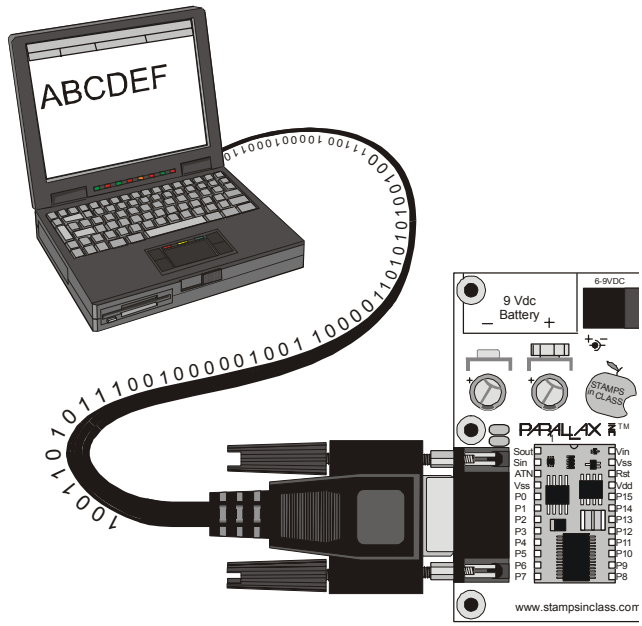
Beispiel: BASIC Stamp 2 wurde auf COM2 gefunden.

√ Prüfen Sie das Identifikationsfenster um sicherzustellen, dass eine BASIC Stamp 2 an einem der COM Ports gefunden wurde. Wenn die BASIC Stamp 2 gefunden wurde sind Sie bereit für den nächsten Schritt: Aktivität 4: .

√ Wenn das Identifikationsfenster an keinem der COM-Ports eine BASIC Stamp findet, gehen Sie zu Seite 293 (Anhang A:).

AKTIVITÄT 4: IHR ERSTES PROGRAMM

Das erste Programm das Sie schreiben und testen werden befiehlt der BASIC Stamp, eine Meldung (*message*) an Ihren PC oder Laptop zu senden. Figur 1-34 zeigt, wie die BASIC Stamp eine Folge von Einsen und Nullen sendet, um die Buchstaben des Textes zu übermitteln, welche der PC anzeigt. Die Einsen und Nullen werden als Binärzahlen (*binary numbers*) bezeichnet. Wie Sie gleich sehen werden, ist die BASIC Stamp Editor Software in der Lage, diese Meldungen zu erkennen und anzuzeigen.



Figur 1-34
Meldungen von der BASIC Stamp an Ihren Computer

Ihr erstes Programm

Die Beispiele, die Sie in den BASIC Stamp Editor eingeben und zur BASIC Stamp downloaden werden sind immer grau unterlegt. Hier ist ein Muster:

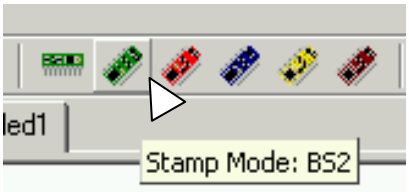
Beispielprogramm: HelloBoeBot.bs2

```
' Robotics with the Boe-Bot - HelloBoeBot.bs2
' BASIC Stamp sends a text message to your PC/laptop.

' {$STAMP BS2}
```

```
' {$PBASIC 2.5}

DEBUG "Hello, this is a message from your Boe-Bot."

END
```

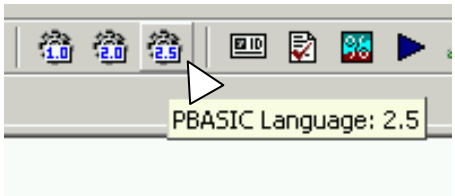
Wir werden nun dieses Programm in den BASIC Stamp Editor eingeben. Einige Programmzeilen werden automatisch eingetragen indem man die entsprechenden Buttons auf der Toolbar (Werkzeugleiste) klickt. Andere Zeilen werden manuell über die Tastatur eingegeben.

- √ Beginnen Sie, indem Sie das BS2 Icon (den grünen diagonalen Chip) auf der Toolbar klicken, (Figur 1-35). Wenn Sie mit dem Cursor über diesen Button fahren erscheint ein kurzer Hilfetext “Stamp Mode: BS2”.
- √ Klicken Sie nun auf das Zahnrad-Icon mit der Bezeichnung “2.5” wie in Figur 1-36 gezeigt. Sein Hilfetext ist “PBASIC Language: 2.5”.



Figur 1-35
BS2 Icon

Ein Klick auf diesen Button fügt automatisch “ ‘ {$STAMP BS2}” *am Anfang Ihres Programms ein.*



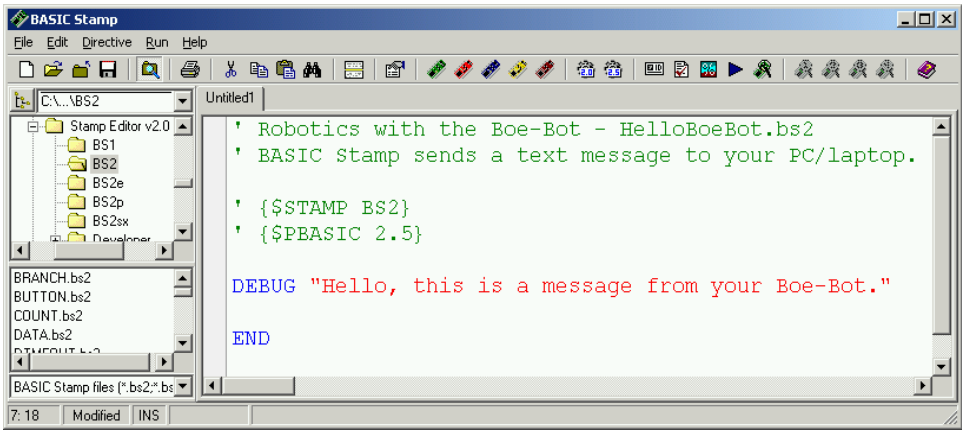
Figur 1-36
PBASIC 2.5 Icon

Ein Klick auf diesen Button fügt automatisch “ ‘ {$PBASIC 2.5}” *am Anfang Ihres Programms ein.*

Benutzen Sie IMMER diese Toolbar Buttons um diese zwei Zeilen als Anfang jedes Programms einzufügen!

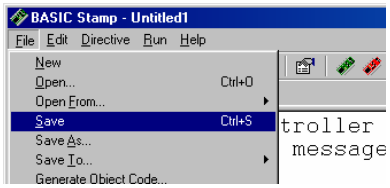
Compiler Direktiven (*Compiler directives*) benutzen geschweifte Klammern { }. Wenn Sie versuchen, diese als Teil Ihres Programms einzutippen könnten Sie versehentlich normale ( ) oder eckige [ ] Klammern verwenden. Wenn das passiert, wird Ihr Programm nicht funktionieren.

√ Geben Sie den Rest des Programms genau so in den BASIC Stamp Editor ein wie in Figur 1-37 gezeigt. Beachten Sie, dass die ersten zwei Zeilen oberhalb der Compiler Direktiven erscheinen, und der Rest der Programms unterhalb.



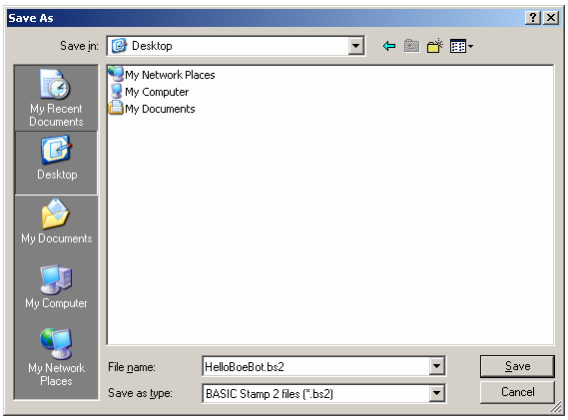
Figur 1-37 HelloBoeBot.bs2 im BASIC Stamp Editor erfasst

√ Speichern Sie Ihre Arbeit indem Sie File (Datei) klicken und Save (Speichern) auswählen wie in Figur 1-38.



Figur 1-38
Speichern des Programms HelloBoeBot.bs2

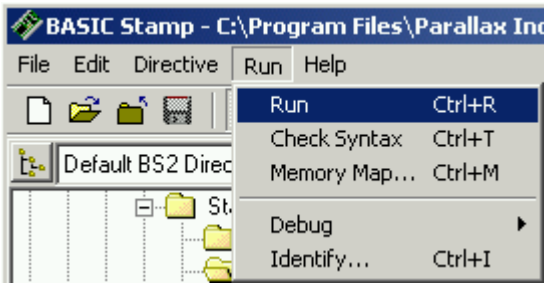
√ Geben Sie den Namen”HelloBoeBot.bs2” in das Feld File name unten im Save As Window ein (Figur 1-39).
√ Klicken Sie den Save Button.



Figur 1-39
Eingeben des Dateinamens

Das nächste Mal wenn Sie speichern, benutzt der BASIC Stamp Editor automatisch den selben Dateinamen (HelloBoeBot.bs2), es sei denn Sie wählen selbst einen anderen Dateinamen indem Sie *File* klicken und *Save As* (statt nur Save) wählen.

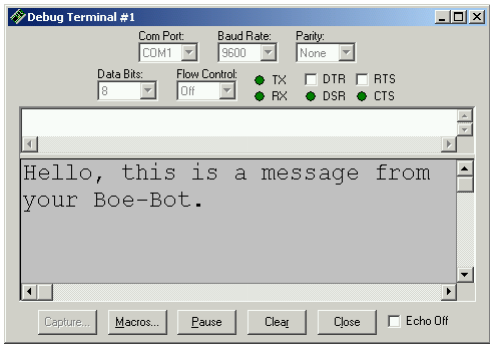
√ Klicken Sie Run, and wählen Sie (durch draufklicken) Run vom Menu das erscheint (Figur 1-40).



Figur 1-40
Ihr erstes Programm laufen lassen HelloBoeBot.bs2

Ein Download Progress Fenster erscheint kurz während das Programm vom PC oder Laptop an Ihre BASIC Stamp übermittelt wird. Figur 1-41 zeigt das Debug Terminal das erscheinen sollte, sobald der Download beendet ist. Sie können sich überzeugen, dass dies eine Meldung der BASIC Stamp ist, indem Sie den Reset button (Reset-Knopf) auf Ihrem Board of Education oder HomeWork Board drücken. Jedes Mal, wenn Sie den Knopf drücken und loslassen wird das Programm neu gestartet, und Sie werden wieder eine Meldung in Ihrem Debug Terminal angezeigt bekommen.

√ Drücken Sie den Reset Button kurz. Haben Sie gesehen, dass eine zweite "Hello…" –Meldung im Debug Terminal erscheint?



Figur 1-41
Debug Terminal

Das Debug Terminal zeigt Meldungen an, welche die BASIC Stamp an Ihren PC/Laptop sendet.

Der BASIC Stamp Editor hat Shortcuts (Tastatur-Kombinationen) für die häufigsten Aufgaben. Um z.B. ein Programm ausführen zu lassen, können Sie die Tasten 'Ctrl' ('Strg' auf gewissen Tastaturen) and 'R' gleichzeitig drücken. Sie können auch den *Run* Button klicken. Das ist das blaue Dreieck in Figur 1-42, das aussieht wie der Abspielknopf an einem CD-Player. Der Hilfetext (Run) erscheint, wenn Sie den *Run* Button mit Ihrer Maus berühren. Mit den Hilfetexten können Sie herausfinden, was die anderen Buttons bewirken, indem Sie mit der Maus darauf zeigen.

Figur 1-42
BASIC Stamp Editor Shortcut Buttons

Wie das Programm funktioniert

Die ersten beiden Zeilen im Programmbeispiel werden als Kommentar (*comment*) bezeichnet. Ein Kommentar ist eine Textzeile, die vom BASIC Stamp Editor ignoriert wird, weil sie für den menschlichen Leser des Programms vorgesehen ist, nicht für die BASIC Stamp. In PBASIC wird alles, was rechts von einem Apostroph (') steht, vom BASIC Stamp Editor normalerweise als Kommentar betrachtet. Der erste Kommentar sagt, aus welchem Buch das Beispiel stammt und wie der Name der Programmdatei lautet. Der zweite Kommentar enthält eine nützliche, einzeilige Kurzbeschreibung, was das Programm macht.

```
' Robotics with the Boe-Bot - HelloBoeBot.bs2
' BASIC Stamp sends a text message to your PC/laptop.
```

Es gibt Ausnahmen: Es gibt verschiedene spezielle Anweisungen, die Sie dem BASIC Stamp Editor senden können, indem Sie diese in einen Kommentar verpacken (rechts vom Apostroph einer Programmzeile. Diese Spezialanweisungen heissen Compiler Direktiven (*compiler directives*). Jedes Programm in diesem Text hat die folgenden zwei Direktiven:

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

Die erste Direktive heisst *Stamp directive*. Sie sagt dem BASIC Stamp Editor, dass Sie das Programm in eine BASIC Stamp 2 downloaden werden. Die zweite Direktive heisst *PBASIC directive*. Sie sagt dem BASIC Stamp Editor, dass Sie die Version 2.5 der PBASIC Programmsprache benutzen.

Ein Programmbefehl (*command*) ist ein Wort das Sie benutzen können, um die BASIC Stamp eine gewisse Aufgabe ausführen zu lassen. Der erste der zwei Befehle in diesem Programm ist der **DEBUG** Befehl:

```
DEBUG "Hello, this is a message from your Boe-Bot."
```

Das ist der Befehl der die BASIC Stamp anweist, eine Meldung über das serielle Kabel an den PC zu senden.

Der zweit Befehl heisst **END**:

```
END
```

Dieser Befehl ist praktisch, weil er die BASIC Stamp in Stromsparmodus (*low power mode*) schaltet, sobald sie mit der Programmausführung fertig ist. Im Stromsparmodus wartet die BASIC Stamp darauf, dass entweder der Reset Knopf (*reset button*) gedrückt (und losgelassen) wird, oder dass sie vom BASIC Stamp Editor mit einem neuen Programm geladen wird. Wenn der Reset Knopf auf Ihrem Bord gedrückt wird, führt die BASIC Stamp das bereits geladene Programm erneut aus. Wenn ein neues Programm geladen wird, wird das alte überschrieben und das neue startet.

Your Turn – DEBUG Formatters and Control Characters
Sie sind dran: DEBUG Formatanweisungen und Steuerzeichen

Eine **DEBUG** Formatanweisung (DEBUG *formatter*) ist ein Codewort, das Sie benützen können, um die Darstellung einer Meldung der BASIC Stamp im Debug Terminal zu

beeinflussen. **DEC** ist ein Beispiel einer Formatanweisung, welche eine Anzeige von Dezimalzahlen im Debug Terminal bewirkt. Ein Beispiel für ein Steuerzeichen (*control character*) ist **CR.** **CR** bewirkt, dass die nachfolgenden Daten am Anfang einer neuen Zeile ausgegeben werden. ("Carriage Return" = Wagenrücklauf) Der Begriff *Wagenrücklauf* stammt aus der Zeit, wo Computerprogramme mit mechanischen Schreibmaschinen geschrieben wurden. Wenn eine Zeile voll war, musste der bewegliche Teil, der "Wagen", wieder an den Zeilenanfang zurückgefahren werden. Seither heisst dieser Vorgang, mit dem eine neue Zeile begonnen wird, unverändert Wagenrücklauf.) Sie können Ihr Programm so abändern, dass es zusätzliche **DEBUG** Befehle zusammen mit einigen Formatanweisungen und Steuerzeichen enthält. Hier ist ein Beispiel, wie das geht:

- √ Speichern Sie zuerst das Programm unter einem neuen Namen, indem Sie File klicken und Save As wählen.
- √ Ein guter Name für die neue Datei wäre etwa HelloBoeBotYourTurn.bs2.
- √ Ändern Sie die Kommentare am Programmkopf wie folgt ab:

```
' Robotics with the Boe-Bot - HelloBoeBotYourTurn.bs2
' BASIC Stamp does simple math, and sends the results
' to the Debug Terminal.
```

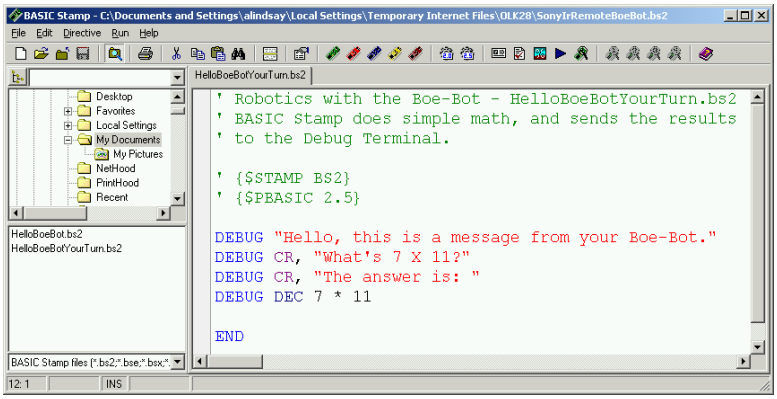
- √ Fügen Sie die folgenden drei Zeilen zwischen dem ersten **DEBUG** Befehl und dem **END** Befehl ein:

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

- √ Speichern Sie die Änderungen, indem Sie File klicken und Save wählen.

Ihr Programm sollte nun aussehen wie das in Figur 1-43.

Starten Sie das modifizierte Programm. Hinweis: Sie müssen entweder Run aus dem Run Menu klicken, wie in Figur 1-40, oder den Run Button, wie in Figur 1-42.



Figur 1-43
Geändertes Programm HelloBoeBot.bs2

Vergleichen Sie Ihr Programm mit diesem Beispiel.

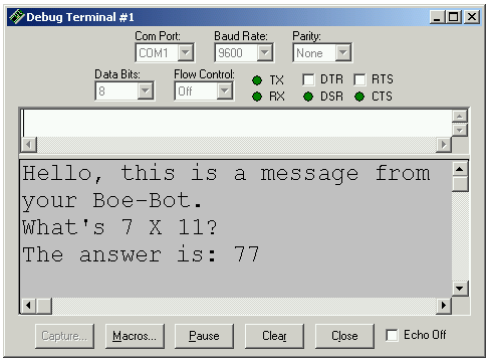
Wohin ist mein Debug Terminal verschwunden? Manchmal wird das Debug Terminal hinter dem BASIC Stamp Editor Fenster verdeckt. Sie können es wieder nach vorne bringen indem Sie wahlweise das *Run* Menu wie links in Figur 1-44 verwenden, den *Debug Terminal 1* Shortcut Button rechts im Bild, oder die F12 Taste Ihrer Tastatur.

Figur 1-44

Debug Terminal 1 in den Vordergrund

Über das Menu (links) und über den Shortcut Button (rechts).

Ihr Debug Terminal sollte nun etwa so aussehen wie Figur 1-45.



Figur 1-45
Terminal Output des modifizierten Programms HelloBoeBot.bs2 Debug

Kontrollieren Sie, ob Sie beim Neustart des Programms die erwarteten Ergebnisse erhalten.

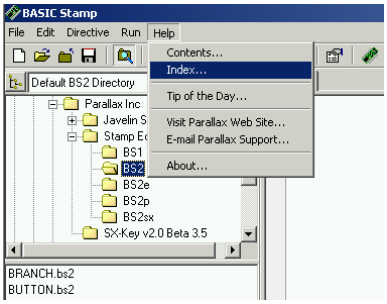
AKTIVITÄT 5: ANTWORTEN FINDEN

Der BASIC Stamp Editor hat eine Hilfefunktion eingebaut, in der jeder Befehl aufgeführt und erläutert ist – allerdings zur Zeit nur in Englisch. Eine vollständige Liste aller PBASIC-Befehle und Hinweise zu Ihrer Verwendung in Deutsch finden Sie aber auch in dem Buch von Claus Kühnel/Klaus Zahnert ***"Basic Stamp 2 Neue Eigenschaften – neue Projekte"*** (ISBN 3-907857-02-X). -et-

The example program you just finished introduced two PBASIC commands: **DEBUG** and **END**. You can find out more about these commands and how they are used by looking them up, either in the BASIC Stamp Editor's Help or in the *BASIC Stamp Manual*. This activity guides you through an example of looking up **DEBUG** using the BASIC Stamp Editor's Help and the *BASIC Stamp Manual*.

Using the BASIC Stamp Editor's Help

√ In the BASIC Stamp Editor, Click Help, then select Index as shown in Figur 1-46.



Figur 1-46
Selecting Index from the Help Menu

√ Type **DEBUG** into the field labeled Type in the keyword to find: (shown in Figure 1-47).

√ When the word **DEBUG** appears in the list below where you are typing, click it, then click the Display button.

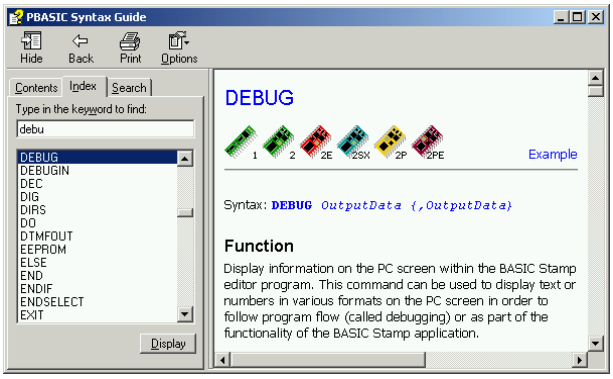


Figure 1-47
Looking up the DEBUG Command Using Help

Your Turn

- √ Use the scrollbar to review the **DEBUG** command's write-up. Notice that it has lots of explanations and example programs you can try.
- √ Click the Contents tab, and find **DEBUG** there.
- √ Click the Search tab, and run a search for the word **DEBUG**.
- √ Repeat this process for the **END** command.

Getting and Using the BASIC Stamp Manual

The *BASIC Stamp Manual* is available for free download from the Parallax web site, and it's also included on the Parallax CD. It can also be purchased as a bound and printed manual.

Downloading the *BASIC Stamp Manual* from the Parallax Web Site

- √ Using a web browser, go to www.parallax.com.
- √ Point at the Downloads menu to display the options.
- √ Point at the Documentation link and click to select it.
- √ When you get to the BASIC Stamp Documentation page, find the *BASIC Stamp Manual*.
- √ Click the Download icon that looks like a file folder to the right of the description: "BASIC Stamp Manual Version 2.0 (3.2 MB)".

Viewing the *BASIC Stamp Manual* on the Parallax CD

- √ Click the Documentation link.
- √ Click the + next to the BASIC Stamps folder.
- √ Click the *BASIC Stamp Manual* book icon.
- √ Click the View button.

√ Figure 1-48 shows an excerpt from the *BASIC Stamp Manual* Contents section (Page 2). It shows that information on the **DEBUG** command can be found on page 97.

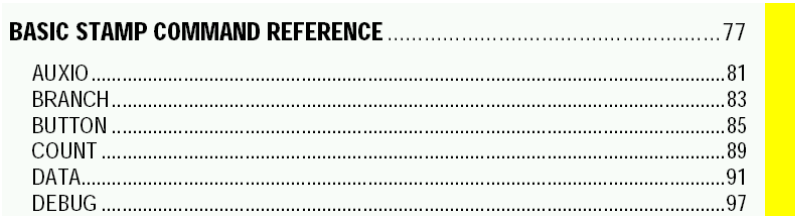
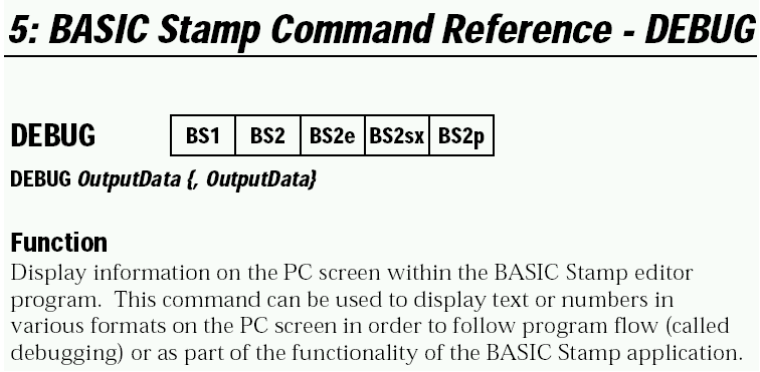


Figure 1-48 Finding the DEBUG Command in the Table of Contents

Figure 1-49 shows an excerpt from the *BASIC Stamp Manual*. The **DEBUG** command is explained in detail here.

- √ Briefly look over the *BASIC Stamp Manual* explanation of the **DEBUG** command.
- √ Count the number of example programs in the **DEBUG** section. How many are there?



5: BASIC Stamp Command Reference - DEBUG

DEBUG | BS1 | BS2 | BS2e | BS2sx | BS2p

DEBUG *OutputData {, OutputData}*

Function

Display information on the PC screen within the BASIC Stamp editor program. This command can be used to display text or numbers in various formats on the PC screen in order to follow program flow (called debugging) or as part of the functionality of the BASIC Stamp application.

Figure 1-49 Reviewing the **DEBUG** Command in the *BASIC Stamp Manual*

Your Turn

- √ Use the *BASIC Stamp Manual* index to look up the **DEBUG** command.
- √ Look up the **END** command in the *BASIC Stamp Manual*.

AKTIVITÄT 6: EINFÜHRUNG IN DEN ASCII CODE

In Aktivität 4: benutzen wir die **DEC** Formatanweisung mit dem **DEBUG** Befehl um eine Dezimalzahl im Debug Terminal anzuzeigen. Aber was passiert, wenn Sie bei einer Zahl keine **DEC** Formatanweisung verwenden? Wenn Sie den **DEBUG** Befehl gefolgt von einer Zahl ohne Formatanweisung verwenden wird die BASIC Stamp diese Zahl als einen ASCII Code betrachten.

Programmieren mit ASCII Codes

ASCII ist die Abkürzung für **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. Die meisten Mikrocontroller und Personalcomputer verwenden diesen Code um jeder Taste oder Funktion auf der Tastatur eine Zahl zuzuweisen. Gewisse Zahlen gehören zu Tastaturbewegungen wie Cursor Auf, Cursor Ab, Leerschlag oder Löschen. Andere Zahlen gehören zu druckbaren Zeichen und Symbolen. Die Zahlen 32 …126 gehören zu jenen Zeichen und Symbolen, welche die BASIC Stamp im Debug Terminal anzeigen kann. Das folgende Programm benutzt ASCII-Code um die Wörder “BASIC Stamp 2” im Debug Terminal anzuzeigen.

Beispielprogramm – ASCIIName.bs2

- √ Geben Sie das Programm ASCIIName.bs2 im Editor ein und führen sie es aus.

Denken Sie daran, die Icons aus der Toolbar zu verwenden, um Compiler Direktiven in Ihre Programme einzufügen!

'{$STAMP BS2} – Benutzen Sie das grüne schräge Chip-Icon .
'{$PBASIC 2.5} – Benutzen Sie das Zahnrad-Icon mit Aufschrift 2.5.

Sie können sich die Ikonen auf Seite 24 nochmals ansehen..

```
'What's a Microcontroller - ASCIIName.bs2
'Use ASCII code in a DEBUG command to display the words BASIC Stamp 2.

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

DEBUG 66,65,83,73,67,32,83,116,97,109,112,32,50

END
```

Wie das Programm funktioniert

Jede Zahl im **DEBUG** Befehl gehört zu einem ASCII-Code Symbol das im Debug Terminal erschien.

```
DEBUG 66,65,83,73,67,32,83,116,97,109,112,32,50
```

66 ist der ASCII Code für ein grosses “B”, 65 ist der Code für ein grosses “A” und so weiter. 32 ist der Code für einen Leerschlag zwischen zwei Zeichen. Beachten Sie, dass jede Codezahl mit einem Komma abgetrennt war. Die Kommata ermöglichen es dem einen Befehlswort **DEBUG** jede Zahl als separaten Befehl auszuführen. Das ist viel einfacher, als 12 separate **DEBUG** Befehle einzutippen.

Sie sind dran – ASCII Code ausprobieren

- √ Speichern Sie ASCIIName.bs2 als ASCIIRandom.bs2
- √ Wählen Sie 12 Zufallszahlen zwischen 32 und 127.
- √ Ersetzen Sie die ASCII Codezahlen im Programm mit Ihren Zahlen.
- √ Führen Sie das geänderte Programm aus, um zu sehen, was rauskommt.

Das *BASIC Stamp Manual* Appendix A enthält eine Tabelle der ASCII Codezahlen und den dazugehörenden Zeichen. Sie können damit die ASCII-Codes zum Buchstabieren Ihres Namens zusammenstellen:

- √ Speichern Sie ASCIIRandom.bs2 als YourASCIIName.bs2
- √ Suchen Sie eine ASCII Tabelle, z.B. im *BASIC Stamp Manual*.
- √ Ändern Sie das Programm so, dass es Ihren Namen buchstabiert.
- √ Starten Sie das Programm um zu kontrollieren, ob der Name richtig ist.
- √ Wenn Sie alles richtig gemacht haben, speichern Sie das Programm!

ACTIVITY 7: WENN SIE FERTIG SIND

Es ist aus verschiedenen Gründen wichtig, die Stromversorgung Ihrer BASIC Stamp und Ihres Board of Education auszuziehen. Erstens halten die Batterien länger, wenn das System keinen Strom verbrauchen kann, solange Sie es nicht benutzen. Zweitens werden wir in künftigen Experimenten Zusatzschaltungen auf dem Prototyping-Feld aufbauen.

Schaltungs-Prototypen sollten niemals unbeaufsichtigt mit angeschlossener Batterie oder Stromversorgung gelassen werden. Man kann nie wissen, was passieren könnten, wenn Sie nicht da sind.

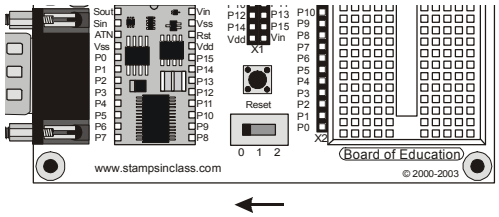
Ziehen Sie immer die Stromversorgung aus Ihrem Board of Education oder HomeWork Board, sogar wenn Sie eigentlich nur für ein, zwei Minuten rausgehen

Wenn Sie in der Schule sind, hat Ihr Lehrer möglicherweise andere oder zusätzliche Anweisungen für das Vorgehen, etwa das serielle Kabel auszuziehen oder das Board zu versorgen. Abgesehen von solchen Einzelheiten ist das wichtigste, dass Sie immer den Strom ausschalten, wenn Sie fertig sind.

Strom ausschalten

Mit dem Board of Education Rev C, ist es besonders einfach, die Stromversorgung abzuschalten:

- √ Wenn Sie ein Board of Education Rev C verwenden, schalten Sie den Dreifachschalter in Position 0 (Nach links schieben wie in Figur 1-50).



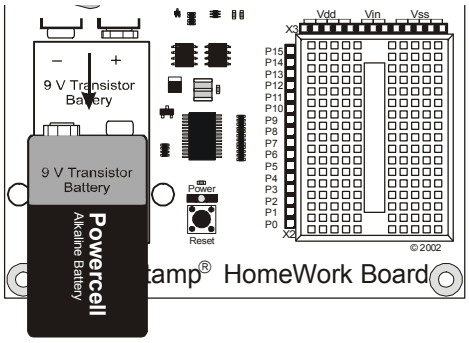
Figur 1-50
Strom ausschalten am Board of Education Rev C

Nehmen Sie die BASIC Stamp NICHT aus ihrem Sockel auf dem Board of Education!

Widerstehen Sie jeder Versuchung, das Board of Education und die BASIC Stamp getrennt aufzubewahren. Jedes Mal wenn die BASIC Stamp aus dem Sockel genommen und wieder eingesetzt wird können Fehler passieren und die BASIC Stamp könnte beschädigt werden. Obwohl die BASIC Stamp in grösseren Projekten manchmal von einem Sockel zu einem anderen bewegt wird, ist das bei keiner der Aktivitäten in diesem Buch nötig.

Strom am BASIC Stamp HomeWork Board ausschalten ist genau so einfach:

√ Wenn Sie das BASIC Stamp HomeWork Board einsetzen, trennen Sie die Batterie wie in Figur 1-51 gezeigt.



Figur 1-51
Strom am HomeWork Board ausschalten

Beim Board of Education Rev A oder B muss ebenfalls der Strom abgestellt werden, entweder durch Ausziehen des Steckers oder Herausnehmen einer Batterie aus dem Batteriefach.

Sie sind dran

√ Schalten Sie jetzt die Stromversorgung aus.

ZUSAMMENFASSUNG

Dieses Kapitel führte Sie durch folgende Themen:

- Eine Einführung in die BASIC Stamp
- Wo man den BASIC Stamp Editor gratis bekommt, den Sie in so ziemlich allen Experimenten dieses Buches verwenden werden
- Wie man die BASIC Stamp Editor Software installiert
- Eine Einführung in die BASIC Stamp, das Board of Education und das HomeWork Board
- Wie man die BASIC Stamp Hardware aufsetzt
- Wie Sie Ihre Software und Hardware testen
- Wie man ein PBASIC Programm schreibt und startet
- Die Verwendung der Befehle **`DEBUG`** und **`END`**
- Die Verwendung des **`CR`** Steuerzeichens und der **`DEC`** Formatanweisung
- Wie man die Help Funktion des BASIC Stamp Editors und das *BASIC Stamp Manual* verwendet (englisch)
- Wie man am Schluss die Stromversorgung Ihres Board of Education oder HomeWork Board abschaltet.

Fragen

1. Welches Bauteil ist das Gehirn Ihres Boe-Bot?
2. Was ist der Zweck des Seriellen Kabels?
3. Welche Art von Zahlen benutzt die BASIC Stamp um ein Zeichen durch das Serielle Kabel an Ihren PC/Laptop zu senden?
4. Was sollten Sie tun zwischen dem Erfassen des Programms in den BASIC Stamp Editor und dem Start des Programms?
5. Wie heisst das Fenster, das die Nachrichten anzeigt, die von der BASIC Stamp an Ihren PC/Laptop gesendet werden?
6. Was bedeutet ein Apostroph (Hochkomma) am Anfang einer Zeile PBASIC Programmcode?
7. Welche PBASIC Befehle lernten Sie in diesem Kapitel?
8. Nehmen wir an, Sie wollen während Ihres BASIC Stamp Projektes eine Pause machen, um eine Erfrischung zu holen, oder vielleicht brauchen Sie einen längeren Unterbruch und werden erst in ein paar Tagen wieder zu Ihrem Projekt zurückkehren. Was sollten Sie immer tun, bevor Sie eine Pause machen?

Übungen

1. Erklären Sie, was Sie mit jedem PBASIC Befehl machen können, den Sie in diesem Kapitel gelernt haben.
2. Erklären Sie, was passieren würde, wenn Sie alle **CR** Steuerzeichen aus den nachfolgenden **DEBUG** Befehlen entfernen würden und schreiben Sie auf, wie das im Debug Terminal aussehen würde.

```
DEBUG "Hello, this is a message from your Boe-Bot!"
DEBUG CR, "What's 7 X 11?"
DEBUG CR, "The answer is: "
```

3. Erklären Sie, was der Stern in diesem Befehl bewirkt:

```
DEBUG DEC 7 * 11
```

4. Was würde das Debug Terminal anzeigen, wenn Sie diesen Befehl ausführen lassen:

```
DEBUG DEC 7 + 11
```

5. Es gibt ein Problem mit den folgenden zwei Befehlen. Wenn Sie den Code ausführen lassen, erscheinen die Zahlen im Display zusammengeklebt und sehen aus wie eine einzige grosse Zahl, statt wie zwei Kleine. Modifizieren Sie die zwei Befehle so, dass die Ergebnisse auf separaten Zeilen im Debug Terminal erscheinen.

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

Projekte

1. Verwenden Sie **DEBUG,** um die Lösung des folgenden Rechenproblems anzuzeigen: 1 + 2 + 3 + 4.
2. Benutzen Sie dazu den Abschnitt “Sie sind dran: DEBUG Formatanweisungen und Steuerzeichen” als Muster, um die Datei unter einem neuen Namen abzuspeichern und zu modifizieren. Verwenden Sie den Dateinamen HelloBoeBotCh01Project02.bs2. Fügen Sie die folgende Zeile in das Programm ein und starten sie es:

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

   Fügen Sie nun die **DEC** Formatanweisung vor jeder dieser Zahl ein, so dass die Zeile so lautet:

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

Starten Sie das Programm erneut und beschreiben Sie, was die **DEC** Formatanweisung bewirkt. Erklären Sie anschliessend in einer kurzen Beschreibung, was jede dieser Zeilen macht:

```
DEBUG "Hello!"
DEBUG 32, 32, 32, 32
DEBUG "Hello again!"
DEBUG 13
DEBUG "Goodbye."
```

3. Erklären Sie, was passieren wird, wenn Sie die **DEC** Formatanweisung aus der folgenden Zeile löschen. Benutzen Sie ein PBASIC Programm um Ihre Vorhersage zu überprüfen.

   ```
   DEBUG DEC 7 * 11
   ```

4. Schauen Sie sich nochmals Figur 1-34 auf Seite 23 an. Können Sie die Zahl 65 (entspricht einem 'A') senden, wenn Sie auf Nullen und Einsen beschränkt sind? Die Antwort lautet, dass die Zahl 65 tatsächlich allein mit Nullen und Einsen dargestellt werden kann. Sie werden später mehr über das Binärsystem (Zahlen mit Basis 2) erfahren. Für den Moment ändern Sie einfach ein Programm ab, indem Sie das nachfolgende Codesegment einfügen und kontrollieren, dass es dasselbe tut, wie das Codesegment in Projekt 2.

   ```
   ' Send the ASCII codes 65, 66, 67, 68, 69, and 70.
   ' Debug Terminal will display "ABCDEF".

   DEBUG %01000001, %01000010, %01000011
   DEBUG %01000100, %01000101, %01000110
   ```

5. Welche Zeilen können Sie in HelloBoeBotYourTurn.bs2 löschen, wenn sie die untenstehenden Befehle gerade oberhalb des **END** Befehls ins Programm einfügen? Überprüfen Sie Ihre Hypothese (Ihre Vorhersage, was passieren wird). Vergessen Sie nicht, HelloBoeBotYourTurn.bs2 unter einem neuen Namen, wie z.B. HelloBoeBotCh01Project05.bs2 abzuspeichern. Machen Sie dann Ihre Änderungen, speichern und starten Sie Ihr Programm.

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

Weitere Hinweise

In diesem Kapitel wurde der Abschnitt Software der Parallax Website bzw. der Parallax CD aufgesucht, um eine Kopie des BASIC Stamp Editors zu installieren. Sie finden in den Abschnitten Documentation sowohl der Parallax Website als auch der Parallax CD kostenlose Kopien von *Robotics with the Boe Bot* (englisches Original dieses Textes) als auch des *BASIC Stamp Manual* (ebenfalls englisch). Gedruckte Exemplare können auch direkt bei Parallax oder beim lokalen Distributor in Ihrem Land gekauft werden.

"*BASIC Stamp Manual*", Version 2.0c, Parallax Inc., 2000
Sie können noch viel mehr über die Befehle **DEBUG** und **END** lernen, wenn Sie diese im *BASIC Stamp Manual* nachschlagen. Sie finden diese über das Inhaltsverzeichnis. Im *BASIC Stamp Manual* hat es viele weitere Beispiele, die Sie zusammen mit ähnlichen Aufgaben wie Sie oben im Abschnitt Projekte gelöst haben ausprobieren können.

Kapitel 2: Die Servomotoren Ihres Boe-Bot

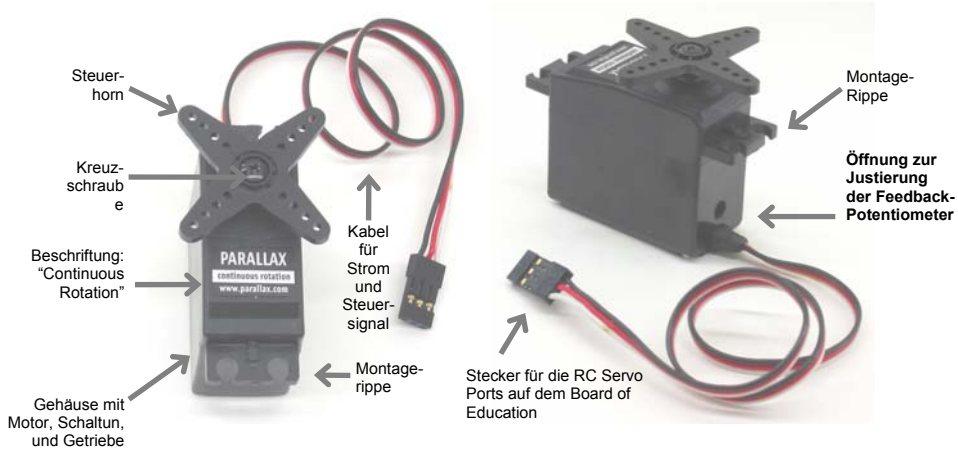
2

Dieses Kapitel führt Sie durch das Anschliessen, Justieren und Testen der Boe-Bot-Motoren. Damit Sie das tun können, müssen Sie gewisse PBASIC-Befehle und Programmiertechniken verstehen, mit denen man Richtung, Geschwindigkeit und Dauer der Servo Bewegungen steuert. Die Arbeitsschritte 1, 2 und 5 führen Sie in diese Programmierwerkzeuge ein, und in den Arbeitsschritten 3, 4 und 6 wenden Sie diese auf die Servos an. Da eine präzise Servosteuerung entscheidend für die Leistungsfähigkeit des Boe-Bot ist, wird es ebenso wichtig wie nötig, dass Sie diese Arbeiten vollständig durchführen, bevor sie die Servos am Chassis des Boe-Bot montieren.

VORSTELLUNG DER FREILAUFSERVOS

Die in Figur 2-1 gezeigten Parallax Freilaufservos (*continuous rotation servos*) sind die Antriebe, um die Räder des Boe-Bot drehen. Dieses Bild zeigt einige der äusseren Teile der Servos. Auf manche dieser Teile wird Bezug genommen, wenn Sie sich durch die Aufgaben dieses und des folgenden Kapitels arbeiten.

Figur 2-1
Parallax Freilaufservo



TIP: Es kann nützlich sein, diese Seite mit einem Buchzeichen zu markieren um sie später einfach wiederfinden.

Standard Servos im Vergleich zu Freilaufservos: Standard Servos sind so konstruiert, dass sie beim Empfang eines elektronischen Signals eine bestimmte Position einnehmen. Standard Servos werden normalerweise für ferngesteuerte Flugzeugklappen, Bootsruder oder Autosteuerungen verwendet. Freilaufservos verwenden ähnliche elektronische Signale, aber statt eine bestimmte Position zu halten drehen sie kontinuierlich in eine bestimmte Richtung in einer bestimmten Geschwindigkeit.

AKTIVITÄT 1: WIE MAN ZEIT MISST UND AKTIONEN WIEDERHOLT

Zur Steuerung von Geschwindigkeit und Drehrichtung eines Servomotors gehört ein Programm, das dafür sorgt, dass die BASIC Stamp dieselbe Meldung immer und immer wieder sendet. Die Meldung muss sich etwa 50 mal pro Sekunde wiederholen, damit der Servo seine Geschwindigkeit und Drehrichtung beibehält. In diesem Abschnitt sehen wir ein paar PBASIC Beispielprogramme die zeigen, wie man dieselbe Meldung immer und immer wiederholt und wie man die Zeitsteuerung der Meldungen kontrolliert.

Meldungen in menschlichen Geschwindigkeiten anzeigen

Sie können den **PAUSE** Befehl benutzen, um der BASIC Stamp zu sagen, dass sie ein Weilchen warten soll, bevor sie den nächsten Befehl ausführt.

PAUSE ***Duration***

Die Zahl die sie hinter den **PAUSE** Befehl schreiben nennt man das ***Duration*** (Dauer) Argument, und es ist diese Zahl, die der BASIC Stamp sagt, wie lange sie warten soll, bevor sie zum nächsten Befehl geht. Die Einheiten für das ***Duration*** Argument sind Millisekunden (ms). Wenn Sie also eine Sekunde warten wollen, setzen Sie den Wert 1000 ein. So sollte der Befehl aussehen:

```
PAUSE 1000
```

Wenn Sie doppelt so lange warten wollen, versuchen Sie:

```
PAUSE 2000
```

Eine Sekunde wird abgekürzt mit “s”. Wenn Sie in diesem Text “1 s” sehen, dann bedeutet das eine Sekunde.

Eine Millisekunde ist eine Tausendstelsekunde, und wird mit “ms” abgekürzt. Der Befehl PAUSE 1000 verzögert das Programm um 1000 ms, was 1000/1000 einer Sekunde entspricht, was einer Sekunde entspricht, oder 1 s. Ok?

Beispielprogramm: TimedMessages.bs2

Es gibt viele verschiedene Möglichkeiten den **PAUSE** Befehl einzusetzen. Dieses Beispiel nutzt **PAUSE** um eine Verzögerung zwischen der Anzeige von Meldungen zu bewirken, welche zeigen, wie viel Zeit vergangen ist.. Das Programm sollte eine Sekunde warten, bevor es die "One Second elapsed…" (Eine Sekunde vergangen…)-Meldung sendet, und weitere zwei Sekunden bevor die Meldung ""Three seconds elapsed…" erscheint.

- √ Geben Sie das folgende Programm in den BASIC Stamp Editor ein.
- √ Speichern Sie das Programm unter dem Namen "TimedMessages.bs2".
- √ Starten Sie das Programm und überwachen Sie die Verzögerung zwischen den Meldungen.

```
' Robotics with the Boe-Bot - TimedMessages.bs2
' Show how the PAUSE command can be used to display messages at human speeds.

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

DEBUG "Start timer..."

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

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

DEBUG CR, "Done."

END
```

Von hier an werden wir die drei Instruktionen oben am Programm wie folgt formulieren:

Eingeben, Speichern und Starten von TimedMessages.bs2.

Sie sind dran: Verschiedene Pausenlängen

Sie können die Verzögerung zwischen Meldungen ändern, indem Sie die ***Duration*** Argumente des **PAUSE** Befehls variieren.

- √ Versuchen Sie zum Beispiel, die ***Duration*** Argumente von **PAUSE** von 1000 und 2000 auf 5000 und 10000 zu ändern:

```
DEBUG "Start timer..."

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

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

√ Starten Sie das modifizierte Programm.
√ Versuchen Sie das auch mit Zahlen wie 40 and 100 für die ***Duration*** Argumente; die gehen ziemlich schnell.

Endlose Wiederholungen

Etwas vom Besten an Computern und Mikrocontrollern ist, dass sie sich nie darüber beschweren, wenn sie immer wieder dieselben langweiligen Dinge tun müssen. Hier ist ein Beispiel eines Programms, das ständig dieselben Befehle ausführt, unermüdlich immer wieder.

Sie können Ihre Befehle zwischen die Wörter **DO** und **LOOP** schreiben, wenn sie unablässig immer wieder ausgeführt werden sollen.

DO...LOOP

Nehmen wir zum Beispiel an, Sie möchten eine Textmeldung immer wieder anzeigen lassen, einmal pro Sekunde. Dazu setzen Sie einfach Ihre **DEBUG** und **PAUSE** Befehle zwischen die Wörter **DO** und **LOOP**, wie hier:

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

Beispielprogramm: HelloOnceEverySecond.bs2

√ Eingeben, Speichern und Starten von HelloOnceEverySecond.bs2.
√ Überzeugen Sie sich, dass die “Hello!” Meldung einmal pro Sekunde angezeigt wird.

```
' Robotics with the Boe-Bot - HelloOnceEverySecond.bs2
' Display a message once every second.

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

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

Sie sind dran: Eine andere Meldung

Sie können Ihr Programm so abändern, dass ein Teil davon nur einmal, ein anderer Teil immer wieder ausgeführt wird.

√ Ändern Sie das Programm, so dass die Befehle so aussehen:

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

√ Starten Sie es und schauen Sie, was passiert. Haben Sie dieses Ergebnis erwartet?

AKTIVITÄT 2: ZEITMESSUNG UND WIEDERHOLUNG MIT EINEM SCHALTKREIS

In diesem Schritt werden sie eine Schaltung bauen, die leuchtet und Ihnen erlaubt die Signale zu "sehen", die zur Steuerung der Boe-Bot-Servos verwendet werden.

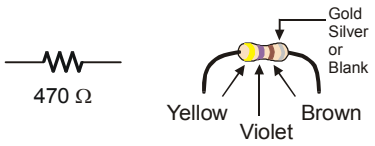
***What's a Microcontroller?* Auszüge –** Dieser Arbeitsschritt enthält ausgewählte Auszüge aus dem Buch *What's a Microcontroller?* Student Guide v2.0.

√ Auch wenn Sie dieses Material aus *What's a Microcontroller?* schon kennen, überspringen Sie diesen Schritt bitte nicht!

Im zweiten Teil dieser Aktivität werden Sie die Signale, die Ihre Servos steuern und die Timing-Diagramme aus einer anderen Perspektive untersuchen, als sie in *What's a Microcontroller?* dargestellt wurden.

Einführung: LED und Widerstand

Ein Widerstand (*resistor*) ist ein Bauteil, das dem Fluss der Elektrizität "Widerstand leistet". Dieser elektrische Fluss wird als Strom bezeichnet. . Jeder Widerstand hat einen Wert der sagt, wie stark er dem Stromfluss widersteht. Die Einheit des Widerstands ist das Ohm, und das Zeichen für das Ohm ist der griechische Buchstabe Omega - Ω. Der Widerstand, mit dem Sie hier arbeiten ist ein 470 Ω Widerstand (Figur 2-2). Der Widerstand hat zwei Anschlussdrähte, aus jedem Ende einen. Zwischen den Anschlussdrähten hat es ein Keramikgehäuse, und dieser Teil ist es, der dem Stromfluss Widerstand entgegensetzt.. Die meisten (angloamerikanischen) Schaltungsdiagramme, in denen Widerstände vorkommen, benutzen das Symbol links mit der Zickzacklinie, um der Person, welche die Schaltung baut zu sagen, dass sie einen 470 Ω Widerstand nehmen muss. Dies nennt man ein Symbol im Schaltschema. . Die Zeichnung rechts ist eine Bauteilzeichnung, wie sie in manchen Anfängertexten des Stamps-in-Class-Programms für die Bauanleitung von Schaltungen verwendet wird



Figur 2-2
470 Ω Widerstand

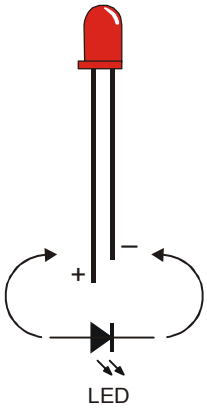
Schemasymbol (links) und Bauteilzeichnung (rechts)

Die farbigen Streifen geben den Widerstandswert an. Vgl. Anhang C für Informationen über die Bestimmung des Widerstandswerts anhand der Farbstreifen auf dem Keramikkörper.

Eine Diode ist ein Einweg-Stromventil, und eine Leuchtdiode (*light emitting diode*, LED) leuchtet, wenn sie von Strom durchflossen wird. Anders als die Farbringe auf den Widerständen zeigt die Farbe einer Leuchtdiode normalerweise nur, in welcher Farbe sie leuchtet, wenn sie vom Strom durchflossen wird.. Die wichtigen Markierungen an einer LED liegen in der Form. Weil eine LED ein Einweg-Stromventil ist, müssen Sie sicherstellen, dass sie richtig herum angeschlossen ist, sonst wird es nicht wie vorgesehen funktionieren.

Figur 2-3 zeigt das Schemasymbol einer LED und eine Bauteilzeichnung. . Die LED hat zwei Anschlüsse. Eine nennt man Anode (+), die andere Kathode (-). In diesem Abschnitt werden Sie die LED in eine Schaltung einbauen und müssen darauf achten, dass die Anoden- und Kathodenanschlüsse richtig in der Schaltung angeschlossen sind. In

der Bauteilzeichnung ist der Anodenanschluss (der längere) mit einem Plus-Zeichen (+) angeschrieben.. Im Schemasymbol ist die Anode die breite Seite des Dreiecks. In der Bauteilzeichnung ist der Kathodenanschluss mit einem Minus-Zeichen (-) beschriftet, und auf dem Symbol ist die Kathode die Linie quer vor der Spitze des Dreiecks.



Figur 2-3
LED Bauteilzeichnung und Schemasymbol

Bauteilzeichnung (oben) und Schemasymbol (unten).

Die LED Bauteilzeichnungen in späteren Bildern werden ein + neben dem Anodenanschluss haben.

Wenn sie Ihre Schaltung aufbauen, vergleichen Sie auf jeden Fall nochmals die Bauteilzeichnung und das Schemasymbol. Beachten Sie in der Bauteilzeichnung, dass die Anschlüsse verschieden lang sind. Der längere Anschluss ist die Anode (+), der kürzere die Kathode (-). Wenn Sie ausserdem das Plastikgehäuse der LED genau anschauen, so ist es zwar rund, doch hat es eine kleine flache Stelle gleich neben dem kürzeren Anschluss, der die Kathode (-) bezeichnet. Das ist wirklich nützlich, wenn jemand die Anschlussdrähte auf gleiche Länge gekürzt hat.

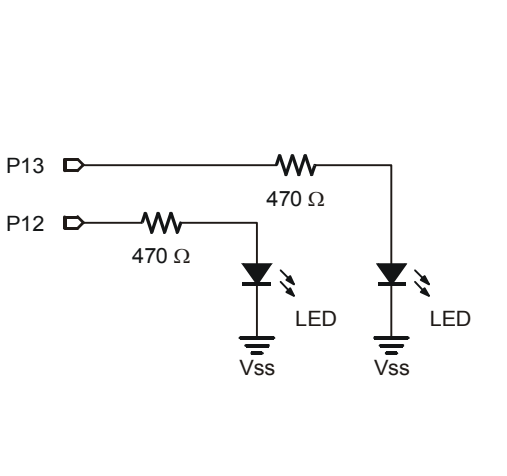
Bauteile der LED Testschaltung

(2) LEDs – rot
(2) Widerstände – 470 Ω (gelb-violett-braun)

LED Testschaltungen

Wenn Sie den Text *What's a Microcontroller?* schon bearbeitet haben, sind Sie zweifellos sehr vertraut mit der Schaltung von Figur 2-4. Links sehen Sie das Schaltschema, rechts die Skizze einer möglichen Verdrahtung der Schaltung, wie sie auf dem Prototyping-Feld Ihres Board aufgebaut wird.

- √ Bauen Sie die Schaltung von Figur 2-4.
- √ Kontrollieren Sie, dass die kürzeren Anschlüsse an jeder LED (die Kathoden) in schwarze Buchsen mit der Aufschrift Vss gesteckt sind.
- √ Kontrollieren Sie, dass die längeren Pins (die Anoden, +) genau so wie gezeigt in die weissen Buchsen gesteckt sind.

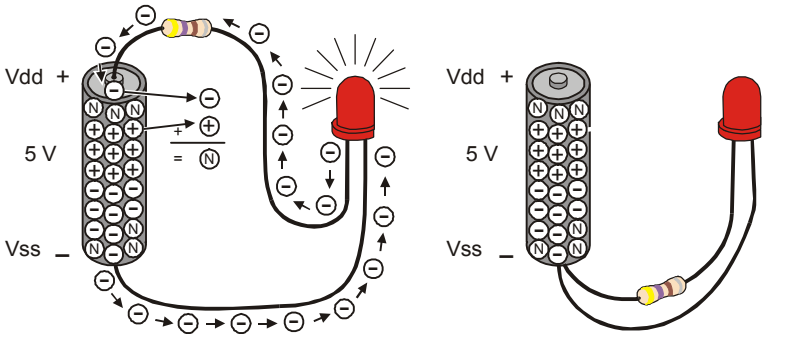


Figur 2-4 Zwei LEDs an I/O Pin P13 und P12 der BASIC Stamp

Schema (links) und Verdrahtungs -Diagramm (rechts).

Neuling im Schaltungsaufbau? Vgl. Anhang D.

Figur 2-5 zeigt, was sie die BASIC Stamp programmieren werden, mit der LED zu tun. . Stellen Sie sich vor, sie hätten eine 5 Volt (5V) Batterie. Obwohl 5V Batterien nicht üblich sind, enthält das Board of Education einen Spannungsregler-IC (*voltage regulator*), der die BASIC Stamp wie eine 5V Batterie mit Strom versorgt. Wenn Sie eine Schaltung an Vss anschliessen, so ist das, wie wenn Sie die Schaltung am negativen Pol der 5V Batterie kontaktieren. Wenn Sie das andere Ende der Schaltung an Vdd anschliessen, entspricht das einem Anschluss am positiven Pol der 5V Batterie.



Figur 2-5
BASIC Stamp Beschalten

Die BASIC Stamp kann darauf programmiert werden, die LED-Schaltungen intern mit Vdd oder Vss zusammenzuschliessen.

Volt wird abgekürzt mit V. Das bedeutet, 5 Volt ist abgekürzt 5V. Wenn Sie eine Schaltung unter Spannung setzen, ist das wie wenn Sie elektrischen Druck ausüben würden.

Strom bezieht sich auf die Rate, mit der Elektronen durch eine Schaltung fliessen. Sie werden häufig Strommessungen in Ampere, abgekürzt A, sehen. Die Strommenge (*current*), die ein elektrischer Motor verbraucht, wird oft in Ampere (*amps*) gemessen, z.B. 2A, 5A, etc. Allerdings werden die Ströme, die Sie im Board of Education verwenden in Tausendstel eines Ampere – Milliampere (mA) gemessen. . Durch die Schaltung von Figur 2-5 fliessen z.B. 10.5 mA.

Durch diese Verbindungen werden 5V elektrischen Drucks auf die Schaltung ausgeübt, was die Elektronen fliessen und die LED leuchten lässt. Sobald Sie den Widerstandsanschluss vom positiven Batterieanschluss trennen, hört der Strom auf zu fliessen und die LED hört zu leuchten auf. Sie können noch einen Schritt weitergehen und den Widerstandsanschluss mit Vss verbinden, was denselben Effekt hat. Dies ist genau, was Ihr Programm in der BASIC Stamp bewirkt und so die LED anschalten (leuchten) oder abschalten (nicht leuchten) lässt.

Programme zur Kontrolle der LED Testschaltung

Die **`HIGH`** und **`LOW`** Befehle können benutzt werden, damit die BASIC Stamp die LED abwechselnd an Vdd und Vss anschliesst. Das ***`Pin`*** Argument ist eine Zahl zwischen 0 und 15 die der BASIC Stamp sagt, welchen I/O-Pin sie mit Vdd oder Vss verbinden soll..

HIGH *`Pin`*

LOW *`Pin`*

Wenn Sie zum Beispiel den Befehl

```
HIGH 13
```

Benutzen, sagen Sie der BASIC Stamp, den I/O-Pin P13 nach Vdd zu schalten, was dann die LED einschaltet.

Wenn Sie entsprechend den Befehl

```
LOW 13
```

Benutzen, sagen Sie damit der BASIC Stamp sie soll diesen I/O-Pin P13 mit Vss verbinden, was die LED abschaltet. Wir wollen das ausprobieren.

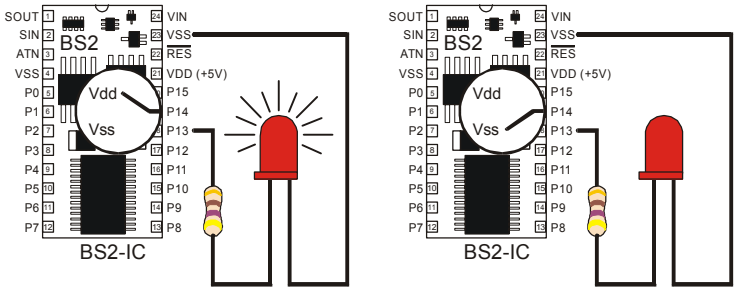
Beispielprogramm: HighLowLed.bs2

- √ Erfassen, Speichern und Starten Sie HighLowLed.bs2.
- √ Kontrollieren Sie, dass der LED Schaltkreis an P13 angeschlossen ist und einmal pro Sekunde an- und abstellt..

```
' Robotics with the Boe-Bot – HighLowLed.bs2
' Turn the LED connected to P13 on/off once every second.

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

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

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

Wie das Programm funktioniert

Figur 2-6 zeigt, wie die BASIC Stamp eine LED-Schaltung abwechselnd nach Vdd und Vss schalten kann. Wenn die LED mit Vdd verbunden ist, leuchtet sie. Wenn sie mit Vss verbunden ist, leuchtet sie nicht. Der Befehl **`HIGH 13`** veranlasst die BASIC Stamp P13 mit Vdd zu verbinden. Der Befehl **`PAUSE 500`** sorgt dafür, dass die BASIC Stamp die Schaltung für 500 ms unverändert in diesem Zustand zu belassen. Der Befehl **`LOW 13`** veranlasst die BASIC Stamp die LED an mit Vss zu verbinden. Wiederum sorgt der Befehl **`PAUSE 500`** dafür, dass die BASIC Stamp für weitere 500 ms keine Änderungen vornimmt. Weil diese Befehle zwischen **`DO`** und **`LOOP`** stehen werden sie immer wieder ausgeführt.



Figur 2-6
Umschalten mit der BASIC Stamp

Die BASIC Stamp kann darauf programmiert werden, die LED-Schaltungen intern mit Vdd oder Vss zusammenzu-schliessen.

Ein Diagnosetest für Ihren Computer

Ganz wenige Computer, wie z.B. gewisse Laptops, halten das PBASIC Programm nach dem ersten Durchlauf durch die **DO...LOOP** Schleife an. Diese Computer haben einen nicht dem Standard entsprechenden seriellen Port. Wenn man einen **DEBUG** Befehl in das Programm LedOnOff.bs2 einfügt, verhindert das offene Debug Terminal, dass das passieren kann. Wir werden als nächstes das Programm ohne den **DEBUG** Befehl neu starten um zu prüfen, ob Ihr Computer dieses Problem am seriellen Port zeigt. Das ist zwar nicht wahrscheinlich, aber nötigenfalls wichtig zu wissen.

- √ Öffnen Sie HighLowLed.bs2.
- √ Löschen Sie den ganzen **DEBUG** Befehl.
- √ Starten sie das modifizierte Programm und beobachten Sie die LED.

Wenn die LED fortwährend blinkt, genauso wie als Sie das Originalprogramm mit dem **DEBUG** Befehl laufen liessen, dann hat Ihr Computer dieses Problem nicht.

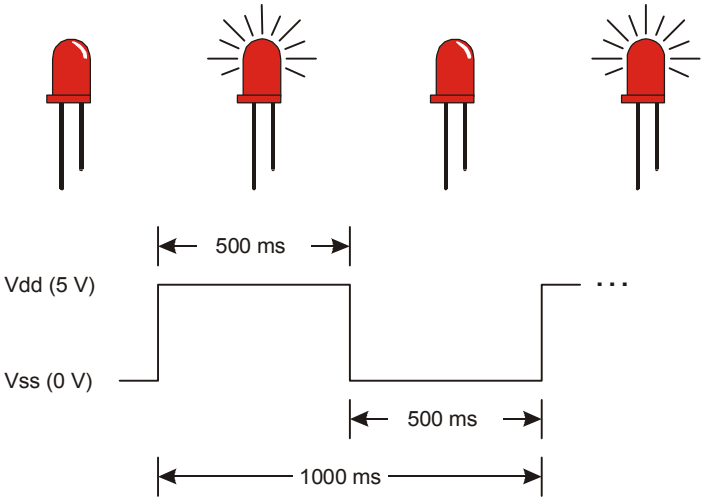
Wenn die LED aber nur einmal ein- und ausschaltet und sich dann nichts mehr tut, dann haben Sie einen Computer mit diesem non-standard seriellen Port. Wenn Sie nun das serielle Kabel am Board ausziehen und den Reset-Knopf drücken, startet die BASIC Stamp das Programm korrekt und ohne einzufrieren. In Programmen, die Sie selbst schreiben, sollten Sie folgenden zusätzlichen Befehl:

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

direkt nach den Compiler Direktiven einfügen. Das öffnet das Debug Terminal und hält den COM Port offen. Damit wird verhindert, dass Ihre Programme nach einem Durchlauf durch die `DO...LOOP` Schleife oder irgend einen anderen Schleifenbefehl stehen bleibt, die Sie in späteren Kapiteln noch kennen lernen werden. Sie werden diesen Befehl in gewissen Beispielprogrammen antreffen, die sonst keinen `DEBUG` Befehl benötigen würden.. Damit sollten Sie in der Lage sein, alle übrigen Programme in diesem Buch laufen zu lassen, auch wenn Ihr Computer beim Diagnosetest durchgefallen ist.

Einführung in das Timing Diagramm

Ein Timing Diagramm ist eine Grafik, der High (Vdd) und Low (Vss) – Signale mit dem Zeitablauf in Beziehung setzt. Die Zeit läuft von links nach rechts, und die High und Low-Signale richten sich mit Vdd (5V) oder Vss (0V) aus. Das Timing Diagramm in Figur 2-7 zeigt einen 1000 ms Abschnitt des High/Low-Signals, mit dem Sie eben experimentiert haben. Die gepunktete Linie (**...**) rechts des Signals ist eine Möglichkeit um anzuzeigen, dass das Signal sich wiederholt.



Figur 2-7
Timing Diagramm für HighLowLed.bs2

Die LED ein/aus-Statis sind über dem Timing Diagramm angegeben.

Sie sind dran: Die andere LED blinkt

Um die andere LED (angeschlossen an P12) zum Blinken zu bringen braucht man nur das ***`Pin`*** Argument in den `HIGH` und `LOW` Befehlen auszuwechseln und das Programm neu zu starten.

√ Modifizieren Sie das Programm, dass die Befehle so aussehen:

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

√ Starten Sie das modifizierte Programm und kontrollieren Sie, ob es die andere LED zum Blinken bringt.

Sie können auch beide LEDs gleichzeitig blinken lassen.

√ Ändern Sie das Programm so ab, dass die Befehle so aussehen:

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

√ Starten Sie das modifizierte Programm und kontrollieren Sie, ob die LEDs nun etwa gleichzeitig ein und aus blinken.

Sie können das Programm noch anders abändern, so dass die LEDs abwechselnd ein- und ausschalten, und Sie können noch die Blinkgeschwindigkeit ändern, indem Sie das ***Duration*** Argument des **PAUSE** Befehls höher oder tiefer setzen.

√ Probieren Sie das doch mal aus!

Anzeigen eines Servo Kontrollsignals mit einer LED

Die High- und Low-Signale, die Ihre BASIC Stamp aufgrund Ihres Programms zur BASIC Stamp sendet um die Servomotoren zu steuern müssen zeitlich sehr genau bemessen sein. Das liegt daran, dass die Servomotoren die Zeitdauer messen, während welcher das Signal High ist, und daraus ableiten, in welche Richtung sie drehen sollen. Für eine präzise Servomotorsteuerung muss die Dauer, während der das Signal High bleibt viel genauer sein, als was Sie mit einem **HIGH** und einem **PAUSE** Befehl erreichen können. Sie können das ***Duration*** Argument des **PAUSE** Befehls nur um jeweils

mindestens 1 ms (Sie erinnern sich, das ist 1/1000 Sekunde) ändern. Das ist zu grob. Dafür gibt es einen anderen Befehl, genannt **PULSOUT**, der zeitlich sehr genaue bemessene High-Signale liefern kann. Die Zeitdauer wird im ***Duration*** Argument, von **PULSOUT** angegeben und seine Schrittweite beträgt zwei Millionstel Sekunden.

PULSOUT ***Pin, Duration***

A Eine Mikrosekunde ist eine Millionstel Sekunde, abgekürzt µs. Seine sie vorsichtig, wenn Sie diesen Buchstaben schreiben, es ist nicht das kleine "u" aus unserem Alphabet, sondern der griechische Buchstabe 'µ' (gesprochen "mü").

8 Mikrosekunden werden zum Beispiel als 8 µs abgekürzt.

Sie können ein **HIGH** Signal senden, das die LED von P13 für 2 µs (also 2 Millionstel einer Sekunde) anstellt, indem Sie diesen Befehl verwenden:

```
PULSOUT 13, 1
```

Dieser Befehl schaltet die LED für 4 µs ein:

```
PULSOUT 13, 2
```

Dieser Befehl sendet ein High-Signal, das Sie tatsächlich sehen können:

```
PULSOUT 13, 65000
```

Wie lange bleibt die LED-Schaltung mit P13 verbunden, wenn Sie diesen Puls senden? Wir rechnen das kurz aus. Die Zeit, während der sie eingeschaltet bleibt ist 65000 mal 2µs. Also:

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

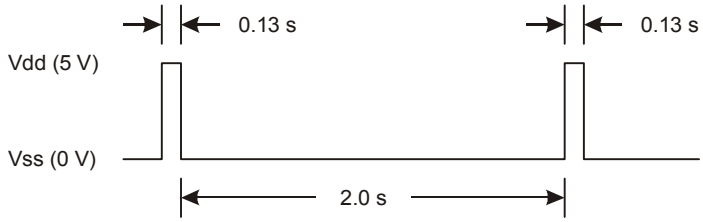
Das ist immer noch ziemlich schnell, 13 Hundertstelsekunden.

Der grösste Wert, den Sie in einem ***Duration*** Argument verwenden können, ist 65535.

Sie können einen Low-Puls senden statt eines High-Pulses, wenn Sie den Befehl **LOW 13** vor dem **PULSOUT** Befehl benutzen.

Beispielprogramm: PulseP13Led.bs2

Das Timing-Diagramm in Figur 2-8 zeigt die Pulsfolge die mit diesem neuen Programm zur LED gesendet wird. Diesmal dauert das High-Signal 0.13 Sekunden, und das Low-Signal dauert 2 Sekunden. Das ist 100 mal langsamer als das Signal, dass der Servo benötigt um seine Bewegung zu steuern.



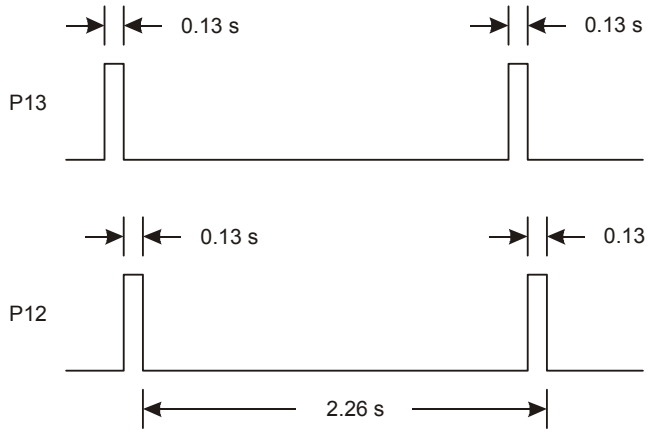
Figur 2-8
Timing Diagramm für PulseP13Led.bs2

√ Erfassen, Sichern und Starten Sie PulseP13Led.bs2.
√ Kontrollieren Sie, ob der an P3 angeschlossene LED-Schaltkreis Pulse von etwa 13/100 einer Sekunde einmal pro zwei Sekunden erhält.

```
' Robotics with the Boe-Bot – PulseP13Led.bs2
' Send a 0.13 second pulse to the LED circuit connected to P13 every 2 s.

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

DEBUG "Program Running!"

DO
  PULSOUT 13, 65000
  PAUSE 2000
LOOP
```

Beispielprogramm: PulseBothLeds.bs2

Dieses Beispielprogramm sendet einen Puls zur LED an P13, und dann sendet es einen Puls zur LED an P12 (Figur 2-9). Danach pausiert es für 2 Sekunden.



Figur 2-9
Timing Diagramm für PulseBothLeds.bs2

Die LEDs leuchten für 0.13 Sekunden während das Signal High ist.

Die Spannungen (Vdd and Vss) in diesem Timing-Diagramm sind nicht beschriftet. Mit einer BASIC Stamp geht man davon aus, dass das High Signal 5V (Vdd) ist und das Low Signal 0V (Vss).

Das ist in Dokumenten die Zeitsteuerung von High und Low-Signalen erklären so üblich. Häufig gibt es ein oder mehrere solcher Dokumente für jede Komponente einer Schaltung, die ein Ingenieur entwirft. Die Ingenieure, welche die BASIC Stamp entwickelt haben, mussten sich durch viele solcher Dokumente hindurcharbeiten, um die benötigten Entscheidungsgrundlagen während der Produktentwicklung zu finden.

Manchmal sind auch die Zeitangaben weggelassen oder haben nur Bezeichnungen wie t_{high} und t_{low}. In diesem Fall sind die benötigten Werte für die Zeiten t_{high} und t_{low} in einer Tabelle irgendwo nach dem Timing Diagramm aufgeführt. Eine vertiefte Diskussion dieses Ansatzes (in englisch) findet sich im Buch *Basic Analog and Digital*, das zum Parallax *Stamps in Class* Programm gehört.

√ Erfassen, Sichern und Starten Sie PulseBothLeds.bs2.
√ Kontrollieren Sie, ob beide LED-Schaltungen gleichzeitig einmal alle zwei Sekunden während etwa 13/100 einer Sekunde Aufleuchten.

```
' Robotics with the Boe-Bot - PulseBothLeds.bs2
' Send a 0.13 second pulse to P13 and P12 every 2 seconds.

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

```
DEBUG "Program Running!"

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

Sie sind dran: Anzeige des Servosignals bei vollem Tempo

Erinnern Sie sich, dass das Servosignal 100 mal schneller ist, als das Programm, das Sie gerade gestartet haben. Zunächst versuchen wir mal, das Programm 10x schneller laufen zu lassen. Das heisst, wir teilen alle ***Duration*** Argumente (**PULSOUT** and **PAUSE**) durch 10.

√ Ändern Sie das Programm so ab:

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

√ Starten Sie das modifizierte Programm und kontrollieren Sie, ob die LEDs zehn mal schneller blinken.

Nun versuchen wir es 100 mal schneller (1/100 der Dauer). Statt zu Flackern wird die LED einfach ein bisschen weniger hell scheinen, als wenn Sie ein einfaches High-Signal senden. Das liegt daran, dass die LED so schnell und so kurz aufblitzt und wieder dunkel wird, dass das menschliche Auge das tatsächliche ein/aus-Flackern nicht unterscheiden kann und nur eine Helligkeitsänderung registriert.

√ Ändern Sie das Programm so ab:

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

√ Starten Sie das Programm und kontrollieren Sie, ob es die beiden LEDs etwa gleich hell leuchten lässt.

√ Versuchen Sie mal 850 im ***Duration*** Argument für den **PULSOUT** Befehl für P13 einzusetzen.

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

√ Starten Sie das modifizierte Programm. Nun sollte die LED von P13 etwas heller sein als die LED von P12. Vielleicht müssen sie mit der Hand eine Röhre um die beiden LEDs machen und hineinlinsen um den Unterschied festzustellen. Der Helligkeitsunterschied kommt von der unterschiedlichen Zeitdauer Die LED von P13 ist länger eingeschaltet als die LED von P12.

√ Versuchen Sie mal 750 im ***Duration*** Argument für den **PULSOUT** Befehl beider LEDs einzusetzen.

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

√ Starten Sie das modifizierte Programm und überzeugen Sie sich, dass die Helligkeit beider LEDs wieder gleich ist. Man mag es vielleicht nicht sehen, aber die Helligkeit liegt zwischen jenen der ***Duration*** Argumente 650 und 850.

AKTIVITÄT 3: ANSCHLUSS DER SERVOMOTOREN

In diesem Abschnitt werden Sie eine Schaltung aufbauen, welche die Servos mit einer Stromversorgung und einem BASIC Stamp I/O-Pin verbindet.

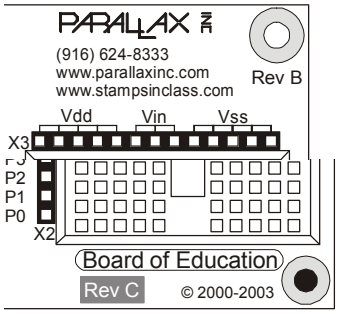
Bauteile für den Anschluss der Servos

(2) Parallax Freilaufservos

Wie Sie die Anschluss-Instruktionen für Ihr Carrier Board finden

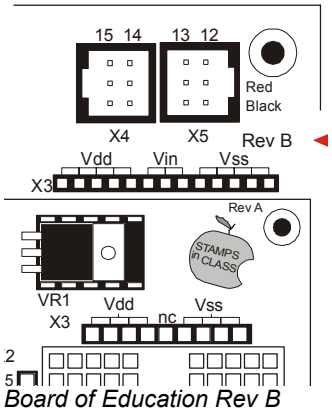
Es gibt drei verschiedene Versionen (“Revisions”, kurz “Rev”) des Board of Education und eine Version des BASIC Stamp HomeWork Board. Die Versionen des Board of Education heissen Rev A, B, oder C. Das HomeWork Board ist Rev B. Figur 2-10 zeigt Beispiele des Aufdrucks, den Sie auf Ihrem Board sehen können.

√ Untersuchen Sie die Bezeichnungen auf Ihrer Trägerplatine und finden Sie heraus, ob Sie ein BASIC Stamp HomeWork Board Rev B oder ein Board of Education Rev C, B, oder A haben.



BASIC Stamp HomeWork Board Rev B

Board of Education Rev C



Board of Education Rev B

Board of Education Rev A

Figur 2-10
Beispiele von Versionsbezeichnungen auf dem BASIC Stamp HomeWork Board und dem Board of Education

√ Nachdem Sie nun die Version Ihrer Trägerplatine kenne, gehen Sie direkt zu dem für Ihre Version gültigen Instruktionen für den Anschluss der Servos an das Board.

Board of Education Rev B

Wenn Sie ein Board of Education Rev B haben, befolgen Sie generell die Anweisungen für das Board of Education Rev C, wobei Sie folgende zwei Punkte im Kopf behalten sollten:

- Das Board of Education Rev B hat keinen 3-Positionen-Schalter. Sie müssen jeweils den Stecker des Batteriepacks ausziehen, wenn im Text stehen, Sie sollen den Dreifachschalter auf 0 stellen. Wenn dort steht, man solle den Dreifachschalter auf 1 oder 2 schieben, müssen Sie den Stromstecker einstecken.
- Das Board of Education Rev B weist keine Jumper (Steckbrücken) zur Einstellung der Stromversorgung auf. Benutzen Sie ausschliesslich das 6V Batteriepack als Stromquelle für Board of Education Rev B Boe-Bot Projekte.

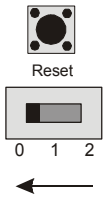
Board of Education Rev A

Wenn Sie ein Board of Education Rev A haben, befolgen Sie generell die Anweisungen für das BASIC Stamp HomeWork Board.

√ Wenn Sie fertig sind, gehen Sie zu Aktivität 4: Einmitten der Servos auf Seite 69.

Anschliessen der Servos an das Board of Education Rev C

√ Schalten Sie den Strom aus indem Sie den 3-Positionen-Schiebeschalter auf dem Board of Education in Position 0 bringen(vgl. Figur 2-11).



Figur 2-11
Strom abschalten

Figur 2-12 zeigt den Servo-Port (Anschlussstecker) auf dem Board of Education Rev C. Diese Platine weist einen Jumper (Steckbrücke) auf, mit der Sie die Stromversorgung des Servos wahlweise an Vin oder Vdd anschliessen können. Um den Jumper umzustecken, müssen Sie ihn senkrecht nach oben von den Pins wegziehen und ihn dann auf jene beiden Pins stecken, wo Sie ihn haben wollen..

√ Wenn Sie das 6V Batteriepack verwenden, überprüfen Sie, dass der Jumper zwischen den Servo-Ports auf dem Board of Education auf Vin steht, wie links in Figur 2-12 gezeigt.

Verwenden Sie ausschliesslich Alkaline AA (1.5 V) Batterien. Vermeiden Sie den Einsatz von Akkus, da diese nur 1.2 V statt 1.5 V Nennspannung haben.

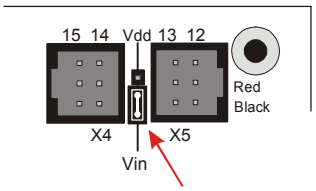
√ Wenn Sie eine 7.5 V, 1000 mA (Mittelanschluss positiv) Gleichstromversorgung (Steckernetzteil) verwenden, stecken Sie den Jumper auf Vdd, wie in Figur 2-12 rechts gezeigt.

ACHTUNG – Falsche Netzteile können Ihre Servos beschädigen.

Wenn Sie sich wenig mit Gleichstromnetzteilen auskennen, empfehlen wir Ihnen den ausschliesslichen Einsatz des 6V Batteriepacks, das zum Boe-Bot gehört.

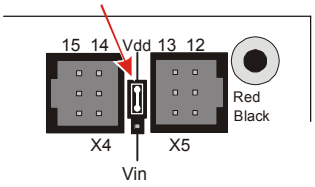
Benutzen Sie ausschliesslich Netzteile mit einer Nenngleichspannung von 6 … 7.5 V und einer Strombelastbarkeit von mindestens 800 mA.

Verwenden Sie ausschliesslich Netzteile mit denselben Steckern, wie sie auch das Boe-Bot Batteriepack hat (2.1 mm, center-positive).



Wählen Sie Vin, wenn Sie das Batteriepack des Boe-Bot Kits verwenden.

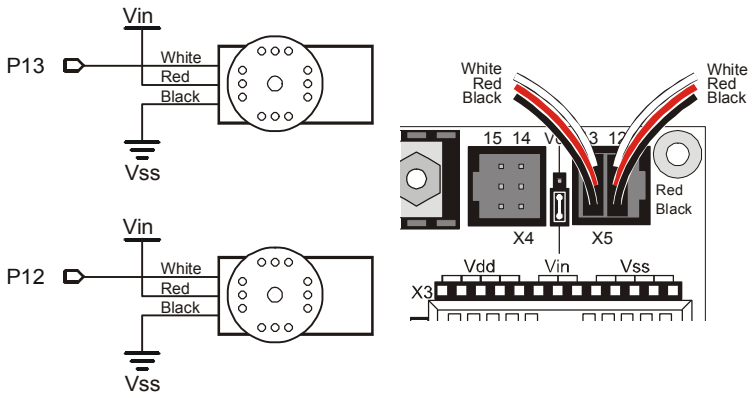
Wählen Sie Vdd, wenn Sie ein Steckernetzteil verwenden.



Figur 2-12
Wahl der Servo-Stromversorgung am Board of Education Rev C

Alle Beispiele und Anweisungen in diesem Buch benutzen ausschliesslich das Batteriepack. Figur 2-13 zeigt das Schema des Schaltkreises, den Sie auf dem Board of Education Rev C aufbauen werden. Der Jumper ist auf Vin gesetzt.

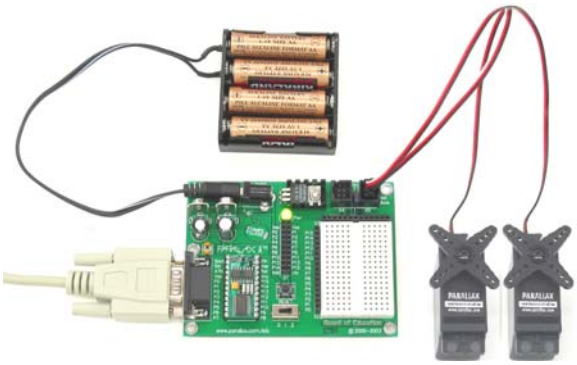
√ Schliessen Sie die Servos am Board of Education Rev C wie in Figur 2-13 gezeigt an.



Figur 2-13
Servo Anschluss Schema und Verdrahtungs-Diagramm für das Board of Education Rev C

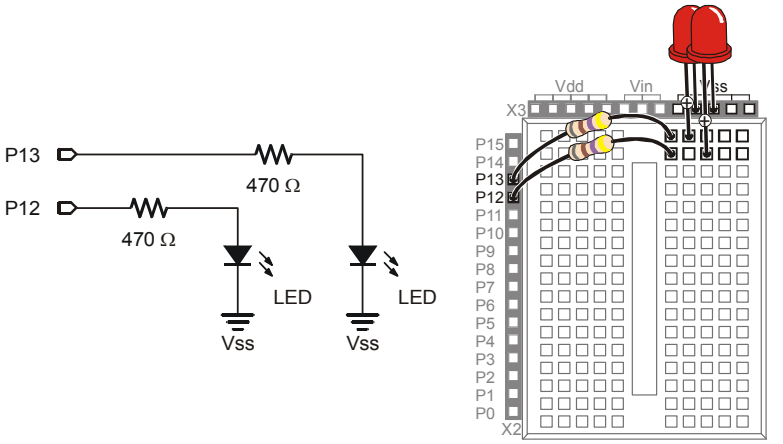
Wie kann ich unterscheiden, welcher Servo an P13 und welcher an P12 angeschlossen ist? Sie haben eben die Servos in die Servo-Steckbuchsen eingesteckt. Diese sind oben nummeriert. Wenn die Nummer oben am Port 13 ist, ist dieser Servo an P13 angeschlossen. Wenn die Zahl 12 ist, ist er mit P12 verbunden.

√ Wenn sie das System fertig zusammengesetzt haben, sollte es etwa aussehen wie Figur 2-14.



Figur 2-14
Board of Education mit angeschlossenen Servos und Batteriepack

√ Fügen Sie nun noch den Servo-Überwachungsschaltkreis Figur 2-15 hinzu.



Figur 2-15
LED Servo Signal Monitor Schaltung

Strom ausschalten – Spezielle Instruktionen für das Board of Education Rev C

Lassen Sie nie den Strom eingeschaltet, wenn Sie nicht mit dem System arbeiten..

√ Um den Strom am Board of Education Rev C abzuschalten, schieben Sie den Dreifachschalter auf Position 0.

√ Gehen Sie nun auf Seite 69 (Aktivität 4:).

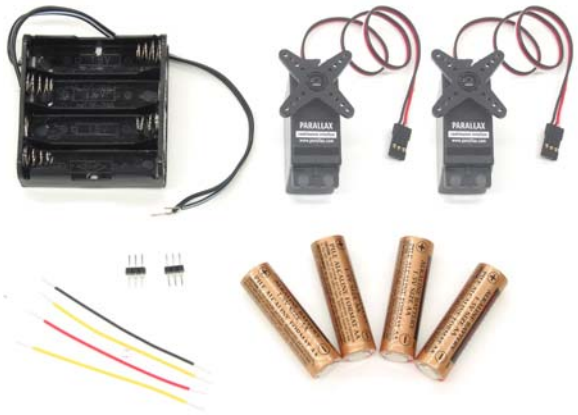
Anschluss der Servos an das BASIC Stamp HomeWork Board

Wenn Sie Ihre Servos an das BASIC Stamp HomeWork Board anschliessen wollen, brauchen Sie folgend gelistete und in Figur 2-16 gezeigten Teile:

Stückliste:

(1) Batteriepack mit verzinnten Drähten
(2) Parallax Freilauf Servos
(2) 3-pin Steckleisten
(4) Verbindungsdrähte
(4) AA Batterien – 1.5 V Alkaline
(2) LEDs – Rot
(2) Widerstände – 470 Ω (gelb-violett-braun)

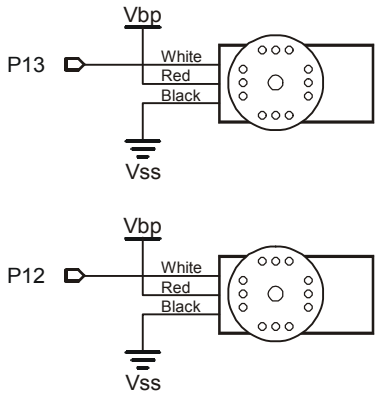
Bemerkung: Es mögen noch zwei LEDs/Widerstände auf der Experimentierplatine sein. Diese können Sie weiter verwenden ohne zusätzliche Bauteile zu benötigen.



Figur 2-16
Servo Zentrierteile für das HomeWork Board

Figur 2-17 zeigt das Schaltschema der Servos auf dem HomeWork Board. Bevor Sie mit dem Aufbau beginnen, trennen Sie die Stromversorgung vom BASIC Stamp HomeWork Board.

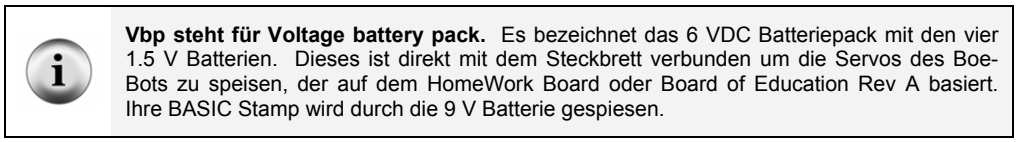
√ Die 9 V Batterie sollte vom Batterieclip getrennt sein und im Batteriepack mit verzinnten Drähten dürfen sich keine Batterien befinden.



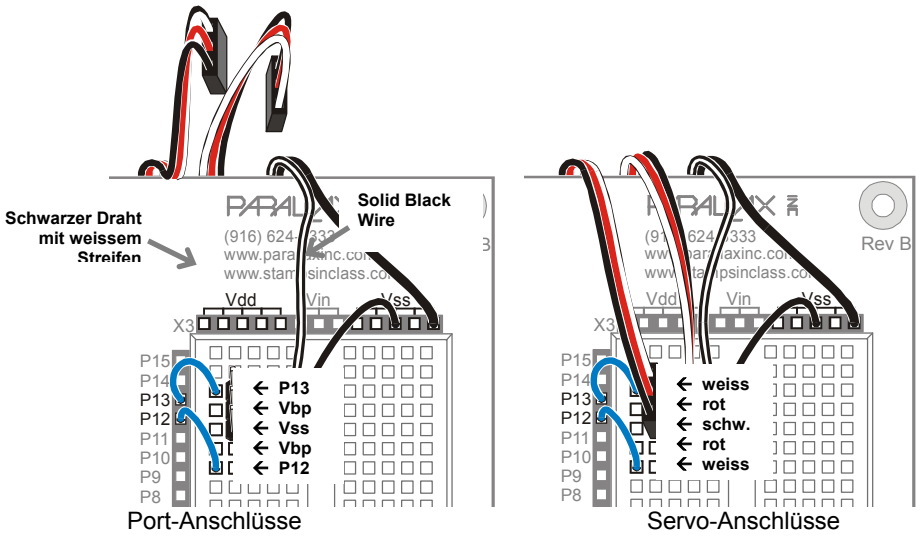
Figur 2-17
Servo Verbindungsschema für das BASIC Stamp HomeWork Board.

√ Entfernen Sie die zwei LEDS/Widerstände und legen Sie diese beiseite.

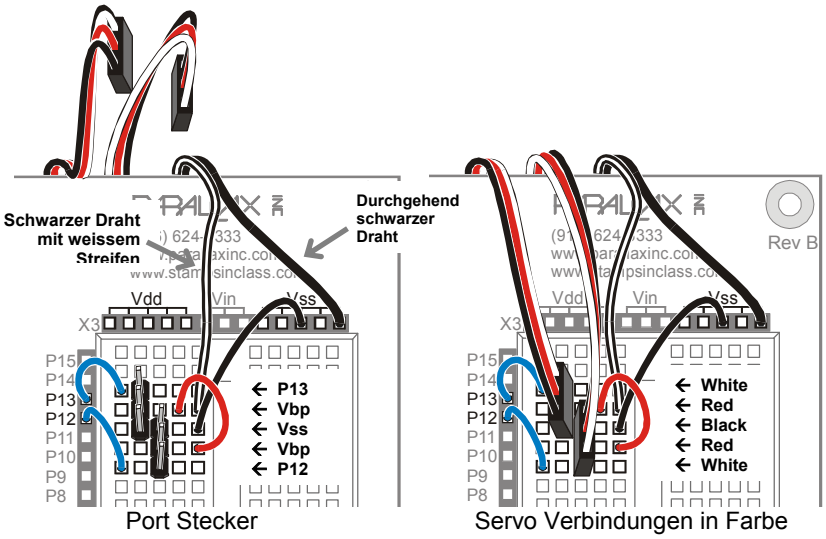
√ Bilden Sie die Servo Ports wie auf der linken Seite in Figur 2-18 gezeigt.
√ Vergewissern Sie sich, dass der schwarze Draht mit weissem Streifen mit Vbp verbunden und der durchgehend schwarze Draht mit Vss verbunden ist.
√ Stellen Sie sicher, dass alle Verbindungen für P13, Vbp, Vss, Vbp, und P12 exakt dem Verdrahtungsplan entsprechen.
√ Stecken Sie die Servoanschlüsse auf den entsprechenden 3-pin Stecker wie in der rechten Figur 2-18 gezeigt.
√ Vergewissern Sie sich, dass die Farben der Servoanschlüsse mit der Legende im Bild übereinstimmen.



Vbp steht für Voltage battery pack. Es bezeichnet das 6 VDC Batteriepack mit den vier 1.5 V Batterien. Dieses ist direkt mit dem Steckbrett verbunden um die Servos des Boe-Bots zu speisen, der auf dem HomeWork Board oder Board of Education Rev A basiert. Ihre BASIC Stamp wird durch die 9 V Batterie gespiesen.



Port-Anschlüsse Servo-Anschlüsse

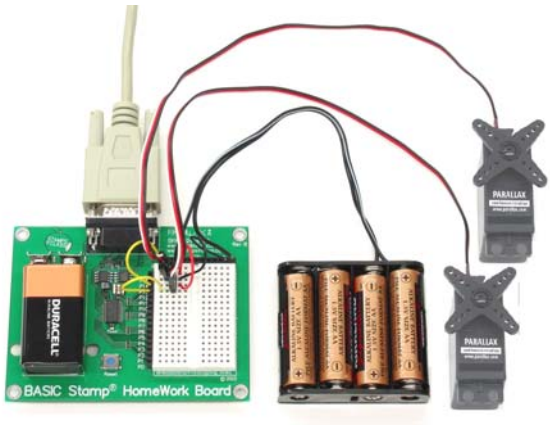


Figur 2-18
Servo Anschluss-diagramm für das BASIC Stamp HomeWork Board

Links (die Servoports).

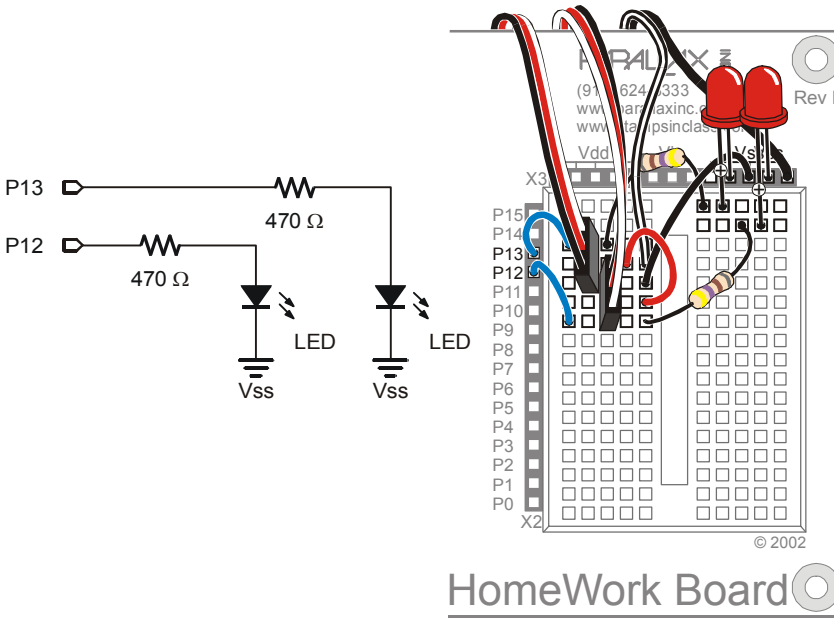
Rechts (Verbinde die Servos).

Ihre Schaltung sollte nun Figur 2-19 gleichen.



Figur 2-19
Beide Speisungen und Servos angeschlossen

√ Ergänzen Sie die LEDs/Widerstände gemäss Figur 2-20.



Figur 2-20
LED Servo Signal Überwachungs -schaltung

√ Nachdem Sie alle Verbindungen ausgeführt und auf Richtigkeit überprüft haben, setzen Sie die Batterien in das Pack und stecken Sie die 9 V Batterie in den HomeWork Board Batterie Clip.

Strom abschalten – Spezielle Anleitung für das HomeWork Board

Lassen Sie Ihr System bei Nichtgebrauch nie an der Stromquelle. Im Weiteren braucht es folgende zwei Schritte den Strom auszuschalten:

√ Ziehen Sie die 9 V Batterie aus dem Clip um das HomeWork Board abzutrennen. Dies trennt die Versorgung zur BASIC Stamp und den Spannungsanschlüssen auf dem Steckbrett (Vdd, Vin, and Vss).

√ Entfernen Sie eine Batterie aus dem Batteriepack. Dies trennt die Stromversorgung der Servos.

AKTIVITÄT 4: EINMITTEN DER SERVOS

In diesem Arbeitsschritt werden Sie ein Programm laufen lassen, das den Servos ein Signal sendet mit dem Befehl, still zu stehen. Weil die Servos vom Hersteller nicht

eingemittet wurden, werden sie sich aber beginnen zu drehen. Sie werden sie dann mit einem Schraubendreher soweit adjustieren, dass sie gerade stillstehen, wenn sie dieses spezielle Signal von der BASIC Stamp empfangen. Dies bezeichnet man als Einmitten der Servos. Nach der Adjustierung werden Sie die Servos testen um zu prüfen, ob sie richtig funktionieren. Die Testprogramme senden Signale für Rechts- und Linkslauf in verschiedenen Geschwindigkeiten.

Servo Werkzeuge und Bauteile

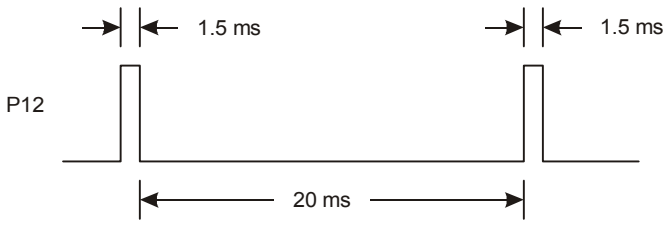
Der Parallax Schraubenzieher in Figur 2-21 ist das einzige Spezialwerkzeug, das Sie für diese Aktivität benötigen. Es geht aber auch mit jedem anderen Kreuzschlitz-Schraubenzieher Grösse 1 mit 3.18 mm (1/8") Schaft und einer Kreuzschlitz-Klinge verwenden.

Figur 2-21
Parallax
Schraubendreher

Das Center-Signal senden

Figur 2-22 zeigt das Signal, das man den Servos für die Kalibrierung senden muss. Dieses Signal wird als Center-Signal bezeichnet und mit diesem Signal werden die Servos angewiesen, in der Mittelposition ("Center") stillzuhalten. Der Befehl besteht aus einem 1.5 ms Puls, der vom selben **`PULSOUT`** Befehl generiert wird, den Sie für das LED-Blinken verwendet haben. Da die Servos so hergestellt sind, dass sie bei 1.5 ms Pulsen in der Mitte-Position verharren geben wir dem **`PULSOUT`** Befehl ein ***`Duration`*** Argument von 750.

Wenn die Servos dieses Center-Signal erhalten, werden sie wahrscheinlich zu drehen beginnen. Mit dem Schraubenzieher können Sie nun während das Signal läuft jeden Servo so lange adjustieren bis er aufhört zu drehen und seine Position hält.

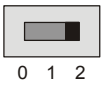


Figur 2-22
Timing Diagramm für CenterServoP12.bs2

Die 1.5 ms Pulse befehlen dem Servo, in der Mittenposition stillzuhalten.

Am besten mittet man nur einen Servo aufs mal ein, weil Sie so hören können, wenn der Motor stoppt, während sie die Justierschraube drehen. Das folgende Programm lässt nur den Servo an P12 drehen, und die Anweisungen führen Sie durch den Adjustierprozess. Wenn Sie mit dem Servo an P12 fertig sind, machen Sie dasselbe mit dem Servo an P13.

√ Wenn Sie ein Board of Education Rev C haben, sorgen Sie dafür, dass der Dreifachschalter auf Position 2 steht wie in Figur 2-23.



Figur 2-23
Schieben Sie den Dreifach-schalter auf Position-2

√ Wenn Sie ein anderes Board of Education oder das HomeWork Board haben, Überprüfen Sie die Stromversorgung sowohl für die BASIC Stamp als auch für die Servos.
√ Erfassen, speichern und starten Sie CenterServoP12.bs2.

Beispielprogramm: CenterServoP12.bs2

```
' Robotics with the Boe-Bot - CenterServoP12.bs2
' This program sends 1.5 ms pulses to the servo connected to
' P12 for manual centering.

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

DEBUG "Program Running!"

DO
  PULSOUT 12, 750
  PAUSE 20
LOOP
```

√ Kontrollieren Sie, ob der LED Signalkontrollschaltkreis an P12 Aktivitäten anzeigt. Die LED sollte leuchten und damit anzeigen, dass die Pulse zum Servo an P12 übermittelt werden.

Wenn der Servo noch nicht eingemittet ist, wird sich sein Montagekreuz zu drehen beginnen und Sie werden ein wimmerndes Motorgeräusch hören.

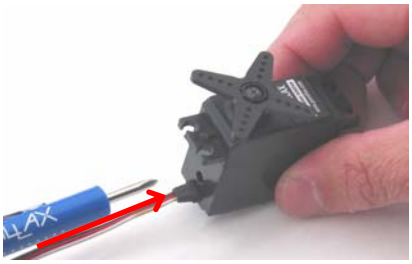
√ Wenn der Servo noch nicht eingemittet wurde, benutzen Sie einen Schraubenzieher um das Potentiometer im Servo vorsichtig, wie in Figur 2-24

gezeigt, zu adjustieren. Drehen Sie das Potentiometer so lange hin und her, bis der Servo aufhört, sich zu drehen.

Vorsicht: Drücken Sie nicht zu stark mit dem Schraubenzieher! Das Potentiometer im Servo ist ziemlich empfindlich. Seien Sie also vorsichtig, beim Adjustieren nicht mehr Druck als nötig auszuüben.

Wenn der Servo schon eingemittet ist, wird er sich nicht drehen. Es ist zwar unwahrscheinlich, aber ein beschädigter Servo würde auch nicht drehen. Diese Fehlermöglichkeit wird in der Aktivität 6 ausgeschlossen, bevor die Servos dann am Chassis des Boe-Bot installiert werden.

√ Wenn die Servos nicht drehen, gehen Sie zum Abschnitt "Sie sind dran", damit Sie den Servo an P13 testen und einmitten können.



Führen Sie die den Kreuzschraubenzieher in das Zugangsloch

Drehen Sie vorsichtig, um das Poti zu justieren

Figur 2-24
Einmitten eines Servos

Ihr Servo ist nun eingemittet.

Was ist ein Potentiometer? Ein Potentiometer ("Poti") ist so etwas wie ein veränderlicher Widerstand. Der Widerstandswert eines Potentiometers wird mit einem beweglichen Teil verändert. An gewissen Potentiometern ist dieser bewegliche Teil ein Knopf oder eine Gleitschiene, andere haben einen (Kreuz-)Schlitz und müssen mit einem Schraubenzieher adjustiert werden. Das Potentiometer im Parallax Freilaufservo wird mit der Spitze eines Kreuzschlitz-Schraubenziehers Grösse 1 gedreht. Mehr über Potentiometer können Sie in *What's a Microcontroller?* und *Basic Analog and Digital* nachlesen.

Sie sind dran: Einmitten des Servos an P13

√ Wiederholen Sie den Prozess für den Servo an P13.

Beispielprogramm: CenterServoP13.bs2

```
' Robotics with the Boe-Bot - CenterServoP13.bs2
' This program sends 1.5 ms pulses to the servo connected to
' P13 for manual centering.

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

DEBUG "Program Running!"

DO
  PULSOUT 13, 750
  PAUSE 20
LOOP
```

Denken Sie daran den Strom vollständig abzuschalten, wenn Sie fertig sind.

Wenn Sie ein Board of Education Rev C haben:

- √ Schieben Sie den Dreifachschalter auf Position 0

Wenn Sie ein BASIC Stamp HomeWork Board haben:

- √ Ziehen Sie die 9 V Batterie vom Batterieclip ab um die Stromversorgung des HomeWork Board zu unterbrechen.
- √ Nehmen Sie eine Batterie aus dem Batteriepack.

AKTIVITÄT 5: WIE MAN WERTE SPEICHERT UND ZÄHLT

Diese Aktivität führt Variablen ein, welche in PBASIC-Programmen zur Speicherung von Zahlen verwendet werden. Die Boe-Bot Programme weiter hinten in diesem Buch werden in grossem Stil von Variablen Gebrauch machen. Sobald Ihr PBASIC-Programm Variablen speichern kann, kann es sie nutzen, um zu zählen. Sobald Ihr Programm zählen kann, kann es steuern und überwachen, wie häufig etwas passiert.

Ihre Servos müssen für diese Aktivität nicht am Strom angeschlossen sein.

- √ Wenn Sie ein Board of Education Rev C haben, schieben Sie den Dreifachschalter auf Stellung 1. Dies unterbricht nur die Stromzufuhr zu den Servo-Ports. Die BASIC Stamp, Vdd, Vin, und Vss bekommen immer noch Strom.
- √ Wenn Sie ein BASIC Stamp HomeWork Board haben, entfernen Sie eine Batterie aus dem Batteriepack, lassen aber die 9V Batterie angeschlossen. Das unterbricht die Stromversorgung der Servo-Ports, lässt aber den Strom zur eingebauten BASIC Stamp, Vdd, Vin und Vss fliessen.

Verwendung von Variablen für das Speichern von Werten, Mathematische Operationen und Zählen

Variablen können benutzt werden, um Werte zu speichern. Bevor man in PBASIC eine Variable benutzen kann, muss man sie deklarieren. .

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

Sie können vier verschiedene Grössen von Variablen deklarieren in PBASIC:

Grösse	**–**	**Speichert**
Bit	–	0 bis 1
Nib	–	0 bis 15
Byte	–	0 bis 255
Word	–	0 bis 65535
		oder -32768 to + 32767

Das nächste Beispielprogramm benutzt nur ein paar Word Variablen.

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

Nachdem Sie eine Variable deklariert haben, können Sie diese auch initialisieren, das heisst, ihr einen Anfangs- oder Initialwert zu geben.

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

Defaultwert – Wenn Sie eine Variable nicht selbst initialisieren, wird das Programm automatisch die Zahl Null in diese Variable speichern. Das nennt man einen Defaultwert (Fehlwert).

Das "=" Zeichen in **`value = 500`** ist ein Beispiel eines Operators. Sie können auch andere Operatoren verwenden um mit Variablen zu rechnen. Hier sind ein paar Multiplikations-Beispiele:

```
value = 10 * value
large = large * medium
```

Beispielprogramm: VariablesAndSimpleMath.bs2

Dieses Programm demonstriert, wie man Variablen deklariert, initialisiert und Operationen auf ihnen ausführt.

- √ Bevor Sie das Programm starten, sagen Sie vorher, was jeder **DEBUG** Befehl anzeigen wird.
- √ Eingeben, Speichern und Starten von VariablesAndSimpleMath.bs2.
- √ Vergleichen Sie die Ergebnisse mit Ihren Vorhersagen und erklären Sie allfällige Unterschiede.

```
' Robotics with the Boe-Bot - VariablesAndSimpleMath.bs2
' Declare variables and use them to solve a few arithmetic problems.

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

value          VAR     Word                ' Declare variables
anotherValue   VAR     Word

value = 500                                ' Initialize variables
anotherValue = 2000

DEBUG ? value                              ' Display values
DEBUG ? anotherValue

value = 10 * anotherValue                  ' Perform operations

DEBUG ? value                              ' Display values again
DEBUG ? anotherValue

END
```

Wie das Programm funktioniert

Dieser Code deklariert die zwei Worte Variablen **value** und **anotherValue**.

```
value          VAR     Word      ' Declare variables
anotherValue   VAR     Word
```

Diese Befehle sind Beispiele für die Initialisierung von Variablen mit Werten die Sie bestimmen

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

Diese **DEBUG** Befehle erlauben Ihnen den Wert der Variablen nach der Initialisierung zu sehen. Da **value** den Wert 500 und **anotherValue** den Wert 2000 zugewiesen erhielt, senden diese **DEBUG** Befehle die Meldungen “value = 500” and “anotherValue = 2000” zum Debug Terminal.

```
DEBUG ? value                       ' Display values
DEBUG ? anotherValue
```

Die Formatanweisung (formatter) "?" des DEBUG Befehls kann vor einer Variablen verwendet werden, um damit im Debug Terminal den Variablennamen und den Dezimalwert ihres Inhalts anzuzeigen. Gleichzeitig wird der Cursor auf den nächsten Zeilenanfang gesetzt (CR). Das ist sehr praktisch um Variableninhalte anzuschauen.

Die Denksportaufgabe in den nächsten drei Zeilen ist, was wird angezeigt? Die Antwort ist dass **`value`** wird auf zehnmal **`anotherValue`** gesetzt. Da **`anotherValue`** 2000 ist, wird **`value`** auf 20,000 gesetzt. Die Variable **`anotherValue`** bleibt unverändert.

```
value = 10 * anotherValue          ' Perform operations

DEBUG ? value                       ' Display values again
DEBUG ? anotherValue
```

Sie sind dran: Rechnen mit negativen Zahlen

Wenn Sie Berechnungen mit negativen Zahlen machen wollen, können Sie die **`SDEC`** Formatanweisung , des **`DEBUG`** Befehls verwenden, um diese anzuzeigen. (SDEC steht für "Signed Decimal" = Dezimalzahl mit Vorzeichen). Hier ein Beispiel mit einem modifizierten VariablesAndSimpleMath.bs2.

√ Löschen Sie folgende Teile von VariablesAndSimpleMath.bs2:

```
value = 10 * anotherValue          ' Perform operations

DEBUG ? value                       ' Display values again
DEBUG ? anotherValue
```

√ Ersetzen Sie diese durch:

```
value = value - anotherValue       ' Answer = -1500

DEBUG "value = ", SDEC value, CR ' Display values again
DEBUG ? anotherValue
```

√ Starten Sie das modifizierte Programm und überzeugen Sie sich, dass **`value`** von 500 auf -1500 ändert.

Zählen und Steuern von Wiederholungen

Der bequemste Weg, wie man die Repetitionshäufigkeit eines Stücks Programmcode steuert ist mit einer **FOR**…**NEXT** Schleife. Da ist die Syntax:

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

Die drei Punkte … zeigen, wo man seine eigenen Befehle zwischen **FOR** und **NEXT** platziert. Vergessen Sie nicht, eine Variable für den Schleifenzähler, das ***Counter*** Argument, zu deklarieren. Die ***StartValue*** und ***EndValue*** Argumente can können Zahlen oder Variablen sein. Wenn Sie in einer Syntaxbeschreibung etwas zwischen geschweiften Klammern {} sehen, bedeutet das, ein optionales Argument. Mit anderen Worten, die **FOR**…**NEXT** Schleife funktioniert auch ohne, aber Sie können es für spezielle Zwecke einsetzen.

Die Schleifenvariable muss nicht notwendigerweise “Counter” oder “Zähler” heissen. Mann kann Sie z.B. “myCounter” nennen.

```
myCounter       VAR     Word
```

Nun ein Beispiel einer **FOR**…**NEXT** Schleife, die **myCounter** als Zählervariable benutzt.
Es zeigt ausserdem den Wert der Variable **myCounter** bei jedem Schleifendurchlauf an.

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

Beispielprogramm: CountToTen.bs2

√ Eingeben, Speichern und Starten von CountToTen.bs2.

```
' Robotics with the Boe-Bot - CountToTen.bs2
' Use a variable in a FOR...NEXT loop.

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

myCounter          VAR     Word

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

```
DEBUG CR, "All done!"

END
```

Sie sind dran: Verschiedene Start- und Endwerte und Zählen in Schritten

Sie können verschiedene Werte für die ***StartValue*** und ***EndValue*** Argumente verwenden.

√ Ändern Sie die **FOR**…**NEXT** Schleife so ab:

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

√ Starten Sie das modifizierte Programm. Haben Sie gesehen, dass Die BASIC Stamp den Wert der Variable auch reduzieren ("dekrementieren") kann, d.h. rückwärts zählen?

Erinnern Sie sich an das optionale **{STEP *StepValue*}** Argument? Sie können es benutzen, um **myCounter** in Schritten zählen zu lassen. Statt 9, 10, 11…, können Sie in Zweier- (9, 11, 13…) oder Fünferschritten (10, 15, 20…) zählen lassen, kurz in jeder Schrittweite, die Sie ***StepValue*** geben, vorwärts oder rückwärts. Hier ein Beispiel für Schrittweite 3:

√ Fügen Sie **STEP 3** zur **FOR**…**NEXT** Schleife hinzu, so dass sie so aussieht:

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

√ Starten Sie das modifizierte Programm und schauen Sie, ob es in 3er Schritten zählt.

AKTIVITÄT 6: TEST DER SERVOS

Eine letzte Aufgabe muss noch erledigt werden, bevor Sie Ihren Boe-Bot zusammensetzen können, und das ist der Test der Servos. In diesem Arbeitsschritt lassen wir Programme laufen, die Servos in verschiedenen Geschwindigkeiten und Drehrichtungen bewegen werden. Damit können Sie kontrollieren, ob Ihre Servos richtig funktionieren, bevor Sie den Boe-Bot zusammenschrauben.

Ihr ganzes System, inklusive Servos, muss für diesen Schritt Strom haben.

- √ Wenn Sie ein Board of Education Rev C haben, schieben Sie den Dreifachschalter auf Position 2. Die verbindet die Stromversorgung mit den Servo Ports, zusätzlich zur Stromversorgung für die BASIC Stamp, Vdd, Vin, und Vss, die Sie bereits mit Position 1 einschalten.
- √ Wenn Sie ein BASIC Stamp HomeWork Board haben, setzen Sie die Batterie, die Sie herausgenommen haben, wieder im Batteriepack ein. Dies stellt die Stromversorgung für die Servo Ports wieder her. Verbinden Sie ausserdem die 9V Batterie wieder mit dem Batterieclip. Damit stellen Sie die Stromversorgung für die eingebaute BASIC Stamp, Vdd, Vin, und Vss wieder her.

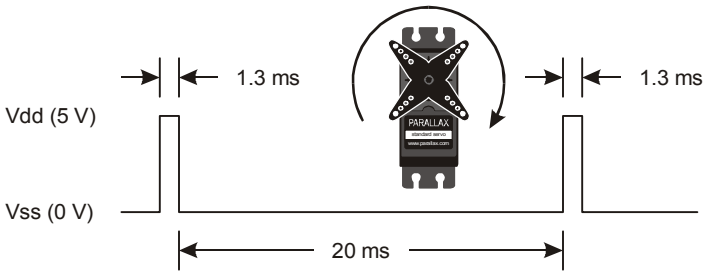
Die ist ein Beispiel, wie man Subsysteme testet. Es ist eine vernünftige Angewohnheit, Subsysteme zu testen, weil es nicht so lustig ist, einen Roboter wieder auseinander-nehmen zu müssen, um ein Problem zu beheben, das man auch hätte finden können, bevor man das ganze zusammenschraubt.

Der Test von Subsystemen ist die Gewohnheit, einzelne Komponenten zu testen bevor Sie in ein grösseres Gerät eingesetzt werden. Es ist eine sinnvolle Strategie, die helfen kann, Roboterwettbewerbe zu gewinnen. Es ist ausserdem eine wesentliche Fähigkeit von Ingenieuren, die weltweit verwendet wird um alles mögliche zu entwickeln, von Spielzeugen, Autos und Videospielen bis zu Space-Shuttles und Marsrobotern. Speziell bei komplexeren Geräten kann es fast unmöglich sein ein Problem zu finden, wenn die Einzelkomponenten vorher nicht getestet wurden. In Luftfahrtprojekten kann es z.B. Hunderttausende oder Millionen Euro kosten, einen Prototypen soweit zu zerlegen, dass ein Problem behoben werden kann. In solchen Projekten wird das Testen von Subsystemen ebenso gründlich wie lückenlos durchgesetzt.

Pulsbreiten steuern Tempo und Richtung

Erinnern Sie sich aus dem Einmitten der Servos, dass ein Signal mit Pulsbreite 1.5 ms die Servos gerade stillstehen lässt. Dies hatten wir mit einem `PULSOUT` Befehl mit einer *`Duration`* von 750 erreicht. Was passiert, wenn die Pulsbreite nicht 1.5 ms ist?

Im Abschnitt Sie sind dran von Aktivität 2, haben Sie die BASIC Stamp dahin programmiert, dass sie Serien von 1.3 ms-Pulsen an eine LED gesandt hat. Diese Pulsserie wollen wir mal näher anschauen und untersuchen, wie sie für die Servosteuerung eingesetzt werden können. Figur 2-25 zeigt, wie ein Parallax Freilaufservo mit Höchstgeschwindigkeit im Uhrzeigersinn dreht, wenn Sie ihm 1.3 ms Pulse senden. Höchstgeschwindigkeit geht etwa von 50 bis 60 Umdrehungen pro Minute (RPM).



Figur 2-25
Eine 1.3 ms Pulsfolge dreht den Servo mit Höchstgeschwindigkeit im Uhrzeigersinn.

Was sind RPM? RPM sind "Revolutions Per Minute" = Umdrehungen pro Minute. Das ist die Anzahl voller Umdrehungen die etwas in einer Minute ausführt.

Sie können ServoP13Clockwise.bs2 verwenden um diese Pulsfolge zum Servo an P13 zu senden.

Beispielprogramm: ServoP13Clockwise.bs2

- √ Erfassen, Speichern und Starten Sie ServoP13Clockwise.bs2.
- √ Kontrollieren Sie, ob sich das Steuerhorn des Servo mit 50 bis 60 RPM im Uhrzeigersinn dreht.

```
' Robotics with the Boe-Bot – ServoP13Clockwise.bs2
' Run the servo connected to P13 at full speed clockwise.

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

DEBUG "Program Running!"

DO
  PULSOUT 13, 650
  PAUSE 20
LOOP
```

Beachten Sie, dass ein 1.3 ms Puls ein ***Duration*** Argument von 650 im **PULSOUT** Befehl verlangt, was weniger ist als 750. Alle Pulsbreiten unter 1.5 ms, und damit **PULSOUT** ***Duration*** Argumente kleiner als 750, veranlassen den Servo im Uhrzeigersinn zu drehen.

Beispielprogramm: ServoP12Clockwise.bs2

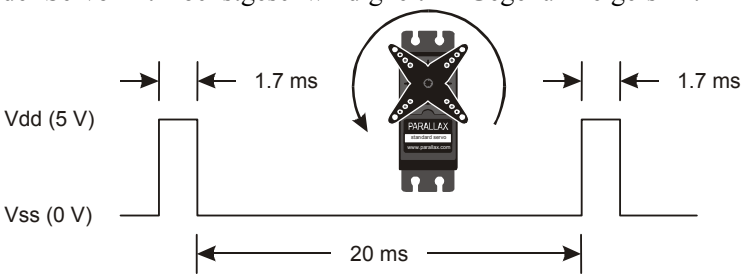
Wenn man das ***Pin*** Argument des **PULSOUT** Befehls von 13 auf 12 ändert, kann man den Servo an P12 mit Höchstgeschwindigkeit im Uhrzeigersinn drehen lassen.

- √ Speichern Sie ServoP13Clockwise.bs2 als ServoP12Clockwise.bs2.
- √ Ändern Sie das Programm, indem Sie die Kommentare nachführen und das ***Pin*** argument des **PULSOUT** Befehls auf 12 ändern.
- √ Starten Sie das Programm und kontrollieren Sie, ob der Servo an P12 jetzt mit 50 bis 60 RPM im Uhrzeigersinn dreht.

```
' Robotics with the Boe-Bot - ServoP12Clockwise.bs2
' Run the servo connected to P12 at full speed clockwise.

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

DEBUG "Program Running!"

DO
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

Beispielprogramm: ServoP12Counterclockwise.bs2

Sie haben Sich wohl schon gedacht, dass die Erhöhung des ***Duration*** Arguments des **PULSOUT** Befehls auf Werte grösser als 750 den Servo im Gegenuhrzeigersinn drehen. Eine ***Duration*** von 850 sendet 1.7 ms Pulse wie in Figur 2-26 gezeigt. Damit dreht sich der Servo mit Höchstgeschwindigkeit im Gegenuhrzeigersinn.



Figur 2-26
Eine 1.7 ms Pulsfolge lässt den Servo mit Vollgas im Gegenuhr-zeigersinn drehen

- √ Speichern Sie ServoP12Clockwise.bs2 als ServoP12Counterclockwise.bs2.

√ Ändern Sie das Programm, indem Sie das ***Duration*** Argument des **PULSOUT** Befehls von 650 auf 850 erhöhen.
√ Starten Sie das Programm und überzeugen Sie sich, dass der Servo an P12 nun mit 50 bis 60 RPM im Gegenuhrzeigersinn dreht.

```
' Robotics with the Boe-Bot – ServoP12Counterclockwise.bs2
' Run the servo connected to P12 at full speed counterclockwise.

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

DEBUG "Program Running!"

DO
  PULSOUT 12, 850
  PAUSE 20
LOOP
```

Sie sind dran: Test von Servo-Geschwindigkeit und Drehrichtung

√ Modifizieren Sie das ***Pin*** Argument des **PULSOUT** Befehls so, dass es den Servo an P13 im Gegenuhrzeigersinn drehen lässt.

Beispielprogramm: ServosP13CcwP12Cw.bs2

Sie können zwei **PULSOUT** Befehle verwenden damit beide Servos gleichzeitig rotieren, und Sie können Sie sogar in verschiedene Richtungen rotieren lassen..

√ Eingabe, Speichern und Laufen lassen ServosP13CcwP12Cw.bs2.
√ Prüfen, dass der an P13 angeschlossene Servo mit voller Geschwindigkeit im Gegenuhrzeigersinn dreht während der an P12 angeschlossene Vollgas im Uhrzeigersinn dreht.

```
' Robotics with the Boe-Bot - ServosP13CcwP12Cw.bs2
' Run the servo connected to P13 at full speed counterclockwise
' and the servo connected to P12 at full speed clockwise.

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

DEBUG "Program Running!"
```

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

Dies wird bald wichtig sein. Überlegen Sie: Wenn die Servos auf beiden Seiten des Chassis montiert sind, muss der eine im Uhrzeigersinn, der andere im Gegenuhrzeigersinn rotieren, damit der BoeBot in einer geraden Linie rollt. Scheint das merkwürdig? Wenn Sie sich das nicht recht vorstellen können, versuchen Sie das folgende:

√ Halten Sie die beiden Servos Rücken an Rücken zusammen und lassen Sie das Programm nochmals laufen.

Sie sind dran: Anpassung von Geschwindigkeit und Richtung

Es gibt vier verschiedene Kombinationen des **PULSOUT** ***Duration*** Arguments, welche in den späteren Kapiteln beim Programmieren der Bewegungen Ihres Boe-Bot häufig vorkommen. ServosP13CcwP12Cw.bs2 sendet eine dieser Kombinationen, 850 an P13 und 650 an P12. Wenn sie mehrere mögliche Kombinationen testen und die Beschreibungsspalte der Table 2-1 ausfüllen werden Sie damit vertraut werden und sich eine Programmbibliothek anlegen. Die "Verhalten"-Spalte werden wir ausfüllen, wenn der Boe-Bot vollständig zusammengesetzt ist; Sie können dann sehen, welche Bewegungen jede der Kombinationen bewirkt.

√ Probieren Sie die folgenden **PULSOUT** ***Duration*** Kombinationen aus und ergänzen Sie die Beschreibungs-Spalte mit Ihren Ergebnissen.

Table 2-1: PULSOUT Dauer Kombinationen			
Dauer		**Beschreibung**	**Verhalten**
P13	**P12**		
850	650	Volle Geschwindigkeit, P13 Servo Gegenuhrzeigersinn, P12 Servo im Uhrzeigersinn.	
650	850		
850	850		
650	650		
750	850		
650	750		
750	750	Beide Servos sollten stillstehen, da sie ja mit diesen Werten in Aktivität 4 eingemittet wurden.	
760	740		
770	730		
850	700		
800	650		

Steuerung der Servolaufzeit mit FOR…NEXT

Hoffentlich haben Sie bis hierher völlig verstanden, dass die Pulsbreite die Geschwindigkeit und Drehrichtung eines Parallax Freilaufservos steuert. Es ist eine ziemlich einfache Art, Geschwindigkeit und Drehrichtung eines Motors zu kontrollieren. Es gibt auch einen einfachen Weg, um die Zeitdauer zu bestimmen, während der eine Motor läuft, und zwar mit einer **FOR**…**NEXT** Schleife.

Hier ist ein Beispiel einer **FOR**…**NEXT** Schleife, die den Servo während einiger Sekunden laufen lässt.:

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

Wir wollen herausfinden, exakt wie lange diese Codezeilen den Servo drehen lassen. Bei jedem Schleifendurchlauf dauert der **PULSOUT** Befehl 1.7 ms, der **PAUSE** Befehl dauert 20 ms, and die Ausführung der Schleife benötigt etwa 1.3 ms.

Ein Schleifendurchlauf = 1.7 ms + 20 ms + 1.3 ms = 23.0 ms.

Da die Schleife 100 mal ausgeführt wird also 23.0 ms mal 100.

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

Nehmen wir an, Sie wollen die Servos 4.6 Sekunden laufen lassen. Dann muss Ihre **FOR**…**NEXT** Schleife doppelt so häufig ausgeführt werden

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

Beispielprogramm: ControlServoRunTimes.bs2

- √ Erfassen, speichern und starten Sie ControlServoRunTimes.bs2.
- √ Kontrollieren Sie, ob der P13 Servo sich für etwa 2.3 Sekunden im Gegenuhrzeigersinn dreht, gefolgt vom P12 Servo, der sich etwa doppelt so lange dreht.

```
' Robotics with the Boe-Bot - ControlServoRunTimes.bs2
```

```
' Run the P13 servo at full speed counterclockwise for 2.3 s, then
' run the P12 servo for twice as long.

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

DEBUG "Program Running!"

counter VAR Byte

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

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

END
```

Nehmen wir an Sie wollen beide Servos laufen lassen, den P13 Servo mit Pulsbreite 850 und den P12 Servo mit einer Pulsbreite von 650. Jetzt benötigt ein Schleifendurchlauf:

1.7ms – Servo an P13
1.3 ms – Servo an P12
20 ms – Pausendauer
1.6 ms – Code Verarbeitungsaufwand
--------- ----------
24.6 ms – Total

Wenn Sie die Servos für eine gewisse Zeitdauer laufen lassen wollen, können Sie folgende Berechnung verwenden:

Anzahl Pulse = Zeit in Sekunden / 0.0246s = Time / 0.0246

Nehmen wir an, Sie wollen die Servos für 3 Sekunden anschalten. Das ergibt

Anzahl Pulse = 3 / 0.0246 = 122

Nun können Sie den Wert 122 im ***EndValue*** der **FOR…NEXT** Schleife einsetzen, und das sieht dann so aus:

```
FOR counter = 1 TO 122
  PULSOUT 13, 850
```

```
    PULSOUT 12, 650
    PAUSE 20
  NEXT
```

Beispielprogramm: BothServosThreeSeconds.bs2

- √ Hier ein Beispiel, wie man die Servos für 3 Sekunden in die eine Richtung drehen lässt, und dann die Richtung umdreht..
- √ Erfassen, speichern und starten Sie BothServosThreeSeconds.bs2.

```
' Robotics with the Boe-Bot - BothServosThreeSeconds.bs2
' Run both servos in opposite directions for three seconds, then reverse
' the direction of both servos and run another three seconds.

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

DEBUG "Program Running!"

counter VAR Byte

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

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

END
```

Überzeugen Sie sich, dass jeder Servo während drei Sekunden in die eine Richtung, und dann weitere drei Sekunden in die andere Richtung rotierte. Haben Sie bemerkt, dass während die Servos gleichzeitig umdrehten, sie jederzeit in entgegengesetzte Richtungen gedreht haben? Wozu könnte das nützlich sein?

Sie sind dran – Wahl der Servolaufzeit

- √ Wählen Sie eine Zeitdauer (in Sekunden), während der die Servos drehen sollen.
- √ Dividieren Sie die Anzahl Sekunden durch 0.024.
- √ Das Ergebnis ist die Anzahl Schleifendurchläufe, die Sie benötigen.

- √ Ändern Sie BothServosThreeSeconds.bs2 so das beide Servos für die gewählte Zeitdauer rotieren.
- √ Vergleichen Sie die gewählte mit der tatsächlichen Laufzeit.
- √ Denken Sie daran, den Strom an Ihrem System (Platine und Servos) abzuschalten, wenn Sie fertig sind. Das heisst den Dreifachschalter in Position 0 zu schieben, wenn Sie ein Board of Education Rev C haben. Wenn Sie ein HomeWork Board haben, ziehen Sie die 9 V Batterie vom Batterieclip ab und entfernen eine Batterie aus dem Batteriepack.

TIP – Um eine Laufzeit zu messen, drücken Sie den Reset Knopf auf Ihrem Board of Education (oder BASIC Stamp HomeWork Board) und halten ihn gedrückt. Wenn Sie für die Zeitmessung bereit sind, lassen Sie den Reset Knopf los.

ZUSAMMENFASSUNG

Dieses Kapitel führte Sie durch das Anschliessen, Justieren und Testen der Parallax Freilaufservos. Dabei haben wir eine Auswahl von PBASIC Befehlen eingeführt. Der **PAUSE** Befehl lässt das Programm für kurze oder längere Zeitperioden anhalten, abhängig vom verwendeten ***Duration*** Argument. **DO**…**LOOP** vereinfacht das mehrfache Wiederholen von einzelnen oder Gruppen von PBASIC Befehlen. **HIGH** und **LOW** wurden eingeführt als eine Möglichkeit, wie man einen I/O Pin der BASIC Stamp auf Vdd oder Vss legen lassen kann. High und Low Signale wurden mit Hilfe einer LED-Schaltung sichtbar gemacht. Diese Signale wurden verwendet, um Timing Diagramme zu erläutern.

Mit dem **PULSOUT** Befehl wurde eine präzisere Möglichkeit gezeigt, wie man High oder Low Signale erzeugen kann, und eine LED Schaltung wurde benutzt um die vom **PULSOUT** Befehl gesendeten Signale anzusehen. **DO**…**LOOP**, **PULSOUT** und **PAUSE** wurden dann benutzt, um den Parallax Freilaufservos das Stillstandssignal zu senden, welches einem Puls von 1.5 ms alle 20 ms entspricht. Die Servos wurden mit einem Schraubendreher während des Empfangs der 1.5 ms Pulse so justiert, dass sie gerade stillstanden. Diesen Prozess nennt man "Einmitten" des Servos.

Nachdem alle Servos eingemittet waren haben wir Variablen eingeführt als eine Möglichkeit, Werte zu speichern. Variablen können in mathematischen Operationen und zum Zählen verwendet werden. Mit den **FOR**…**NEXT** Schleifen wurde ein Weg zum Zählen eingeführt. **FOR**…**NEXT** Schleifen bestimmen, wie oft der Programmcode zwischen dem **FOR** und dem **NEXT** Befehl ausgeführt wird. Die **FOR**…**NEXT** Schleifen wurden dann verwendet, um die Anzahl Pulse für einen Servo zu steuern, welche wiederum bestimmen, wie lange sich der Servo dreht.

Fragen

1. Wozu werden normale Servos gewöhnlich verwendet?
2. Inwiefern unterscheiden sich die Parallax Freilaufservos von normalen Servos?
3. Was bewirkt der **PAUSE** Befehl? Was bewirkt **PAUSE 3000**? Was ist mit **PAUSE 20**?
4. Wie lange dauert eine Millisekunde? Wie kürzt man sie ab?
5. Welche PBASIC Befehle können Sie verwenden, um andere PBASIC Befehle wiederholt ausführen zu lassen?
6. Was bewirkt ein Widerstand in einer Schaltung? Wozu dienen die farbigen Streifen auf einem Widerstand?

7. Was bedeutet die Abkürzung LED? Was bewirkt eine LED in einer Schaltung? Was sind die zwei Anschlüsse der LED?
8. Welcher Befehl veranlasst die BASIC Stamp, einen I/O Pin intern auf Vdd zu legen? Welcher Befehl bewirkt dieselbe Art von Verbindung, aber an Vss?
9. Welchen Befehl können Sie verwenden, um ein High-Signal für eine gewisse Anzahl Mikrosekunden zu erzeugen? Wie lange dauert eine Mikrosekunde, und wie wird sie abgekürzt?
10. Wie heissen die Variablen der verschiedenen Grössen, die in einem PBASIC Programm deklariert werden können? Welche Grössen von Werten kann jede Grösse der Variablen speichern?
11. Was ist der hauptsächliche Unterschied zwischen einer **FOR**…**NEXT** Schleife und einer **DO**…**LOOP** Schleife?
12. Woher weiss eine **FOR**…**NEXT** Schleife, wo sie mit Zählen anfangen und wo aufhören muss?
13. Was ist der Schlüssel zur Steuerung von Drehrichtung und Geschwindigkeit eines Freilaufservos? Wie verhalten sich diese zu einem Timing Diagramm? Wie verhalten sich diese zu den PBASIC Befehlen? Welchen Befehl und welches Argument können Sie anpassen, um Richtung und Geschwindigkeit eines Freilaufservos zu steuern?
14. Was ist die Schlüsselgrösse, mit der man bestimmt, wie lange ein Servo läuft? Welche Befehle werden benötigt, um die Laufzeit eines Servos zu steuern?

Übungen

1. Schreiben Sie einen **PAUSE** Befehl, der bewirkt, dass die BASIC Stamp während 10 Sekunden nichts tut.
2. Zeichnen Sie ein Timing Diagramm für folgenden **DO**…**LOOP**:

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

3. Ändern Sie folgende **FOR**…**NEXT** Schleife so, dass sie von 6 bis 24 in 3er Schritten zählt. Schreiben Sie ausserdem die Variablendeklaration auf, die Sie benötigen, damit dieses Programm funktioniert.

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

```
NEXT
```

4. Ändern Sie diesen **DO**…**LOOP** so, dass beide Servos im Gegenuhrzeigersinn drehen.

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

5. Nehmen Sie Ihre Antwort zur vorhergehenden Frage und ändern Sie diese so ab, dass die Servos während 6.5 Sekunden drehen. Vergessen Sie nicht, Ihre Berechnungen aufzuschreiben.

Projekte

1. Zeichnen Sie ein Schaltschema, in dem die LED Schaltungen an P14 und P15, und die zwei Servos an P12 und P13 angeschlossen sind.
2. Schreiben Sie ein Programm, das den Servo an P12 während 2 Sekunden im Gegenuhrzeigersinn drehen lässt. Während dieser Zeit soll die LED an P14 leuchten. Das Programm soll dann während 2 Sekunden nichts tun. Dann soll das Programm den Servo an P13 im Uhrzeigersinn drehen. Dabei soll die LED an P15 aufleuchten.
3. Schreiben Sie ein Programm, das die LED an P14 schwach leuchten lässt (Ein/Aus mit jedem Puls), solange der Servo an P12 dreht.
4. Schreiben Sie ein Programm, das die Servos durch je 3 Sekunden von jeder der vier verschiedenen Rotations-Kombinationen führt. Hinweis: Sie werden vier verschiedene **FOR**…**NEXT** Schleifen benötigen. Zuerst sollen beide Servos im Gegenuhrzeigersinn rotieren, dann beide im Uhrzeigersinn. Dann soll der P12 Servo im Uhrzeiger-, der P13 Servo im Gegenuhrzeigersinn drehen. Schliesslich soll der P12 Servo im Gegen-, der P13 Servo im Uhrzeigersinn rotieren.

Kapitel 3: Zusammenbau und Test Ihres Boe-Bot

Dieses Kapitel enthält Instruktion für den Zusammenbau und Test ihres Boe-Bot. Es ist besonders wichtig, dass Sie die jeweiligen Tests ausführen, bevor sie sich dem nächsten Kapitel zuwenden. Auf diese Weise können Sie Fehler vermeiden, die sehr häufig sind und dazu führen, dass ihr Boe-Bot in späteren Kapiteln Unerwartetes tut. Hier eine Zusammenfassung der unterschiedlichen Tätigkeiten in diesem Kapitel:

Tätigkeit	Zusammenfassung
1	Bauen Sie ihren Boe-Bot zusammen
2	Überprüfung ob die Servos richtig verbunden sind durch nochmaliges Testen
3	Verbindung und Testen eines Lautsprechers, der Sie informiert, wenn die Batterien des Boe-Bot beinahe leer sind
4	Optionales Vertiefungsthema – Zeichnen von Transferkurven für die Servos.

AKTIVITÄT 1: ZUSAMMENBAU IHRES BOE-BOT

Diese Tätigkeit führt Sie Schritt für Schritt durch den Zusammenbau ihres Boe-Bot. In jedem Schritt sammeln Sie wenige Teile und setzen Sie so zusammen, dass sie mit dem Bild übereinstimmen. Jedes Bild hat dazugehörende Erklärungen. Folgen Sie diesen sorgfältig.

Servo Werkzeuge und Bauteile

Alle der in Figur 3-1 gezeigten Werkzeuge sind häufig und sind in den meisten Haushalten und Baumärkten zu finden. Sie sind ausserdem in der nächsten Eisenwarenhandlung erhältlich.

Tools

(1) Kreuzschraubenzieher Nr. 1 $^{1}/_{8}$" (3.18 mm) Schaft (notwendig)
(1) $^{1}/_{4}$" Kombischlüssel (Optional)
(1) Spitzzange (Optional)

Figur 3-1
Boe-Bot
Zusammenbau
Werkzeuge

Montieren der Hardware der Oberseite

√ Legen Sie sich zunächst die nachfolgend aufgelisteten Teile bereit.
√ Befolgen Sie dann die Begleitinstruktionen

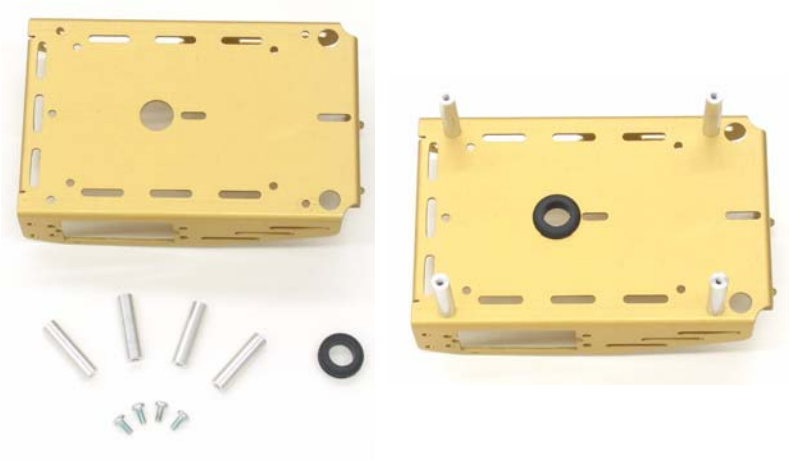
Bauteileliste:

Siehe Figur 3-2.

(1) Boe-Bot Chassis
(4) 1" Abstandsbolzen
(4) Zylinderkopfschrauben, 1/4" 4-40
(1) Gummidichtung, 13/32"

Instruktionen:

√ Fügen Sie die 13/32" Gummidichtung in das Loch in der Mitte des Boe-Bot Chassis..
√ Versichern Sie sich, dass die Rille am äusseren Rand der Gummidichtung auf der Kante des Lochs im Chassis sitzt.
√ Befestigen Sie die vier Abstandbolzen mit den vier Zylinderkopfschrauben an der Chassis wie im Bild gezeigt.



Figur 3-2
Chassis und Oberseiten-Hardware

Bauteile (links); zusammengebaut (rechts)

Boe-Bot Teile – Die Teile für den Boe-Bot sind entweder im Boe-Bot Full Kit, oder in einer Kombination aus Board of Education Full Kit und Robotics Parts Kit enthalten. Vgl. Anhang E: Boe-Bot Parts List für nähere Informationen.

Entfernen der Servo Hörner

- √ Schalten Sie den Strom ihrer BASIC Stamp und ihres Servos aus.
- √ Entfernen Sie alle AA-Batterien aus ihrem Batteriepack.
- √ Entfernen Sie die Servos von ihrem Board.

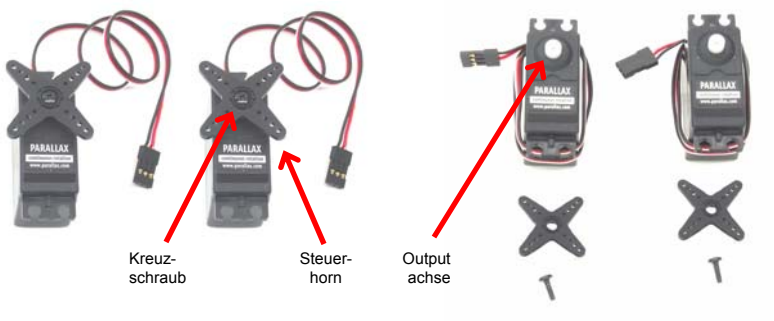
Bauteileliste:

Siehe Figur 3-3.

(2) bereits zentrierte Parallax Freilaufservos

Instruktionen:

- √ Entfernen Sie die Schrauben, welche die Servo Kontrollhörner am Output-Schaft befestigen, mit einem Kreuzschraubenzieher.
- √ Ziehen Sie jedes Horn nach oben, und vom Servo Output-Schaft nach oben.
- √ Bewahren Sie die Schrauben auf, Sie werden Sie später wieder brauchen.



Figur 3-3
Servo Steuer-Horn Demontage

Teile (links); Nach der Demontage (rechts).

Stop!

√ Bevor Sie mit dem nächsten Schritt weitermachen, müssen Sie unbedingt die folgenden Tätigkeiten des Kapitels 2: erledigt haben

- Aktivität 3: Anschluss der Servomotoren
- Aktivität 4: Einmitten der Servos

Montage der Servos am Chassis

Bauteileliste:

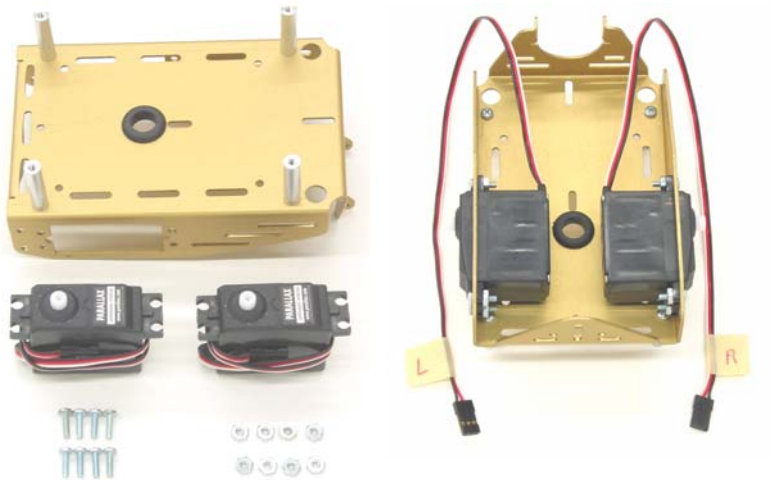
Siehe Figur 3-4.

(1) Boe-Bot Chassis (teilweise zusammengesetzt)
(2) Parallax Freilaufservos
(8) Zylinderkopfschrauben, 3/8” 4-40
(8) Muttern, 4-40

Instruktionen:

√ Bringen Sie die Servos mit den Schrauben und den Muttern am Chassis an. Um die beste Leistung zu erzielen, müssen Sie die Vorderseite jedes Servos durch das viereckige Fenster platzieren, und zwar von der Innenseite der Chassis, anstatt Sie von aussen hineinfallen zu lassen

√ Schreiben Sie die Servos mit einem Stück Klebeband mit links (L) und rechts (R) an.



Figur 3-4
Montage der Servos am Chassis

Teile (links); Zusammengebaut (rechts).

Montage des Batteriepacks

Figur 3-5 zeigt zwei verschiedene Bauteilsätze. Benützen Sie den linken Satz, wenn Sie das Board of Education haben, und den rechten, wenn Sie das Home Work Board haben..

Bauteileliste für den Boe-Bot mit einem Board of Education Rev C:

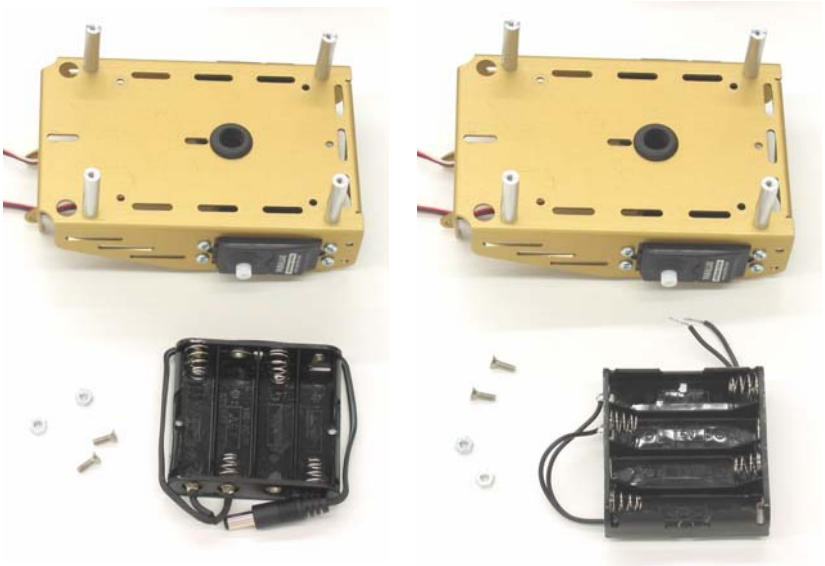
Figur 3-5 (linke Seite).

(1) Boe-Bot Chassis (teilweise zusammengesetzt)
(2) Flachkopfkreuzschrauben, 3/8” 4-40
(2) Muttern, 4-40
(1) Batteriepack mit Stecker (Mittelkontakt positiv)

Bauteileliste für den Boe-Bot mit einem HomeWork Board:

Figur 3-5 (rechte Seite).

(1) Boe-Bot Chassis (teilweise zusammengesetzt)
(2) Flachkopfkreuzschrauben, 3/8” 4-40
(2) Muttern, 4-40
(1) Batteriepack mit verzinnten Anschlussleitungen

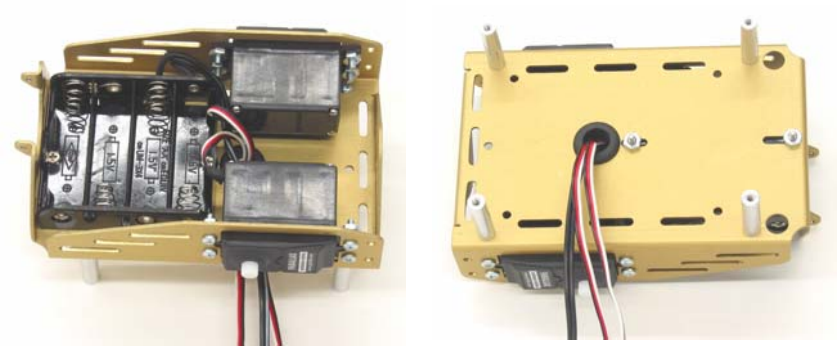


Mit dem Board of Education Mit dem HomeWork Board

Figur 3-5
Batteriepack Montage Hardware

Anweisungen:

- √ Befestigen Sie das Batteriepack mit den zylinderköpfigen Schrauben und den Muttern an der Unterseite der Boe-Bot Chassis, wie auf der linken Seite der Figur 3-6.
- √ Versichern Sie sich, dass die Schrauben durch das Batteriepack, führen, und die Muttern an der Oberseite der Chassis gut angezogen sind.
- √ Ziehen Sie die Stromleitung des Batteriepacks durch dass Loch in der Gummidichtung in der Mitte der Chassis, wie auf der rechten Seite der Figur 3-6 zu sehen
- √ Ziehen Sie die Servokabel durch das gleiche Loch
- √ Ordnen Sie die Servo- und Stromkabel wie auf dem Bild an..



Figur 3-6
Batteriepack installiert

Unterseite (links); Oberseite (rechts).

Montage der Räder

Bauteileliste:

(1) Teilweise zusammengesetzter Boe-Bot (nicht abgebildet)
(1) 1/16” Splint (Keilnadel)
(1) 1” Kunststoffball
(2) Gummibänder als Pneus
(2) Plastikräder
(2) Schrauben, die Sie beim Entfernen der Servohörner beiseite legten

Figur 3-7
Rad Hardware

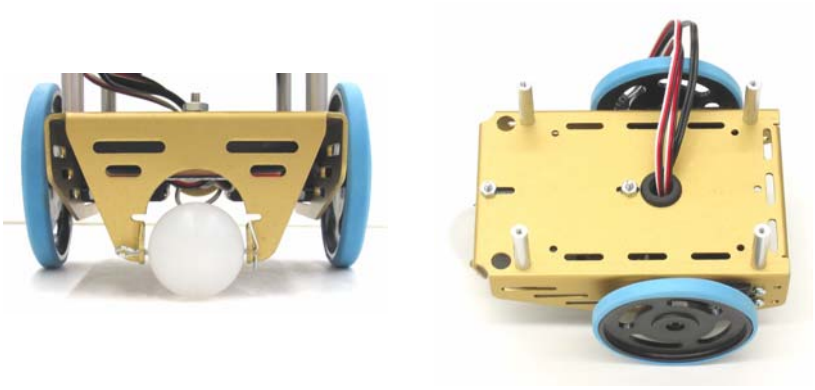
Instruktionen:

Die linke Seite der Figur 3-8 zeigt das am Chassis befestigte Heckrad des Boe-Bot. Das Heckrad ist lediglich ein Plastikball mit einem Loch durch die Mitte. Ein Splint befestigt ihn am Chassis und dient als Achse für das Rad.

√ Bringen Sie das Loch im Heckrad auf eine Linie mit dem Heck des Chassis.
√ Führen Sie den Splint durch alle drei Löcher (Chassis links, Heckrad, Chassis rechts).
√ Biegen Sie die Enden des Splints auseinander, damit er nicht wieder aus dem Loch rutschen kann.

Die rechte Seite der Figur 3-8 zeigt die an den Servos festgemachter Antriebsräder des Boe-Bot.

- √ Strecken Sie beide Gummiband-Pneus und platzieren Sie diese am äussersten Rand jedes Rades.
- √ Beide Plastikräder haben eine Einbuchtung die auf die Servoachse passt. Drücken Sie jedes Plastikrad auf eine Servoachse. Versichern Sie sich dabei, dass die Achse mit der Einbuchtung auf einer geraden Linie liegt und darin einsinkt.
- √ Befestigen Sie die Räder an den Servoachsen mit den Schrauben, die Sie beiseite legten, als Sie die Servohörner entfernten



Figur 3-8
Montage der Räder

Heckrad (links); Antriebsräder (rechts).

Befestigung des Boards am Chassis

Bauteileliste für den Boe-Bot mit einem Board of Education Rev. C

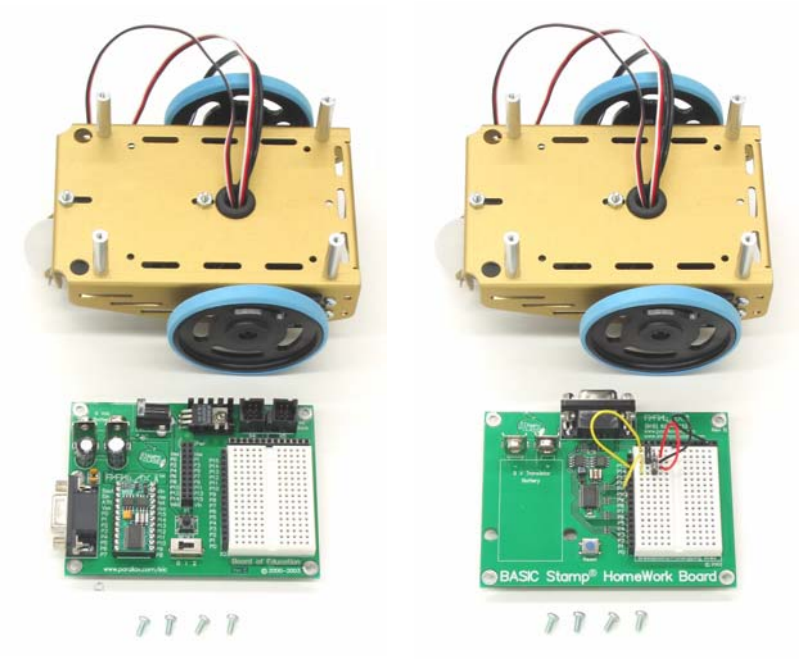
Siehe linke Seite (Figur 3-9).

(1) Boe-Bot Chassis (teilweise zusammengesetzt)
(4) Zylinderkopfschrauben, 1/4” 4-40
(1) Board of Education mit BASIC Stamp 2

Bauteileliste für den Boe-Bot mit einem HomeWork Board:

Siehe rechte Seite (Figur 3-9).

(1) Boe-Bot Chassis (teilweise zusammengesetzt)
(4) Zylinderkopfschrauben, 1/4” 4-40
(1) BASIC Stamp HomeWork Board



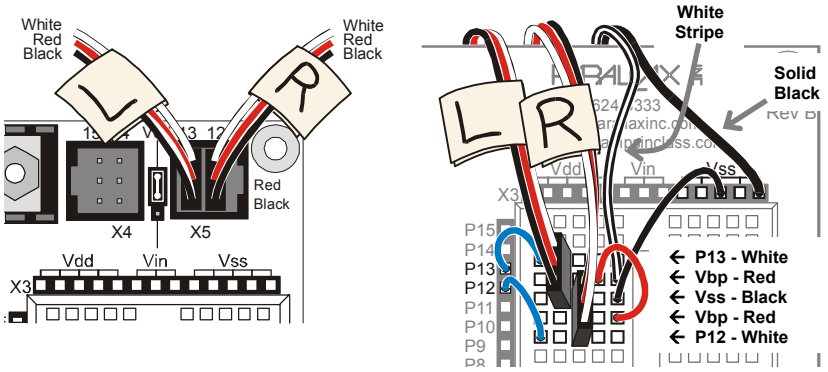
Mit dem Board of Education Rev C

Mit dem HomeWork Board

Figur 3-9
Boe-Bot Chassis und Boards

In Figur 3-10 sehen Sie die Servoports wieder angeschlossen, Board of Education Rev C (linke Seite), HomeWork Board (rechte Seite).

- √ Verbinden Sie die Servos wieder mit den Servo-Steckplätzen.
- √ Versichern Sie sich, dass Sie den mit 'L' markierten Stecker an den Port P13 und den mit 'R' markierten an den Port P12 anschliessen.



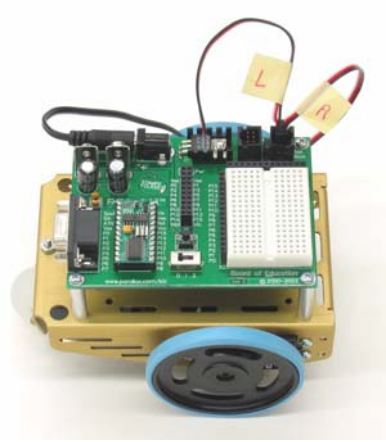
Auf dem Board of Education Rev C Auf dem HomeWork Board

Figur 3-10
Servo Ports Anschlüsse

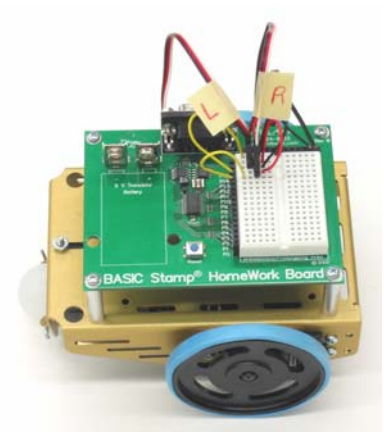
Board of Education Rev C (links) HomeWork Board (rechts).

In Figur 3-11 sehen Sie das Boe-Bot-Chassis mit den entsprechenden Boards.

- √ Setzen Sie das Board so auf seine vier Abstandsbolzen, dass sie in einer Reihe mit den vier Löchern an den äusseren Ecken des Boards sind.
- √ Versichern Sie sich, dass das weisse Breadboard näher bei den Antriebsrädern, nicht näher beim Heckrad ist.



Mit dem Board of Education Rev C

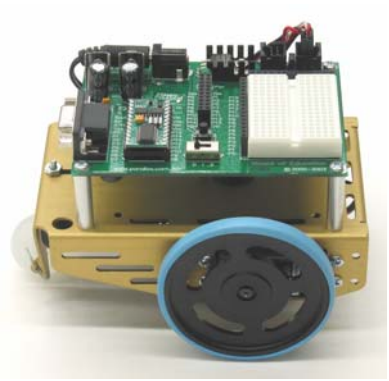


Mit dem HomeWork Board

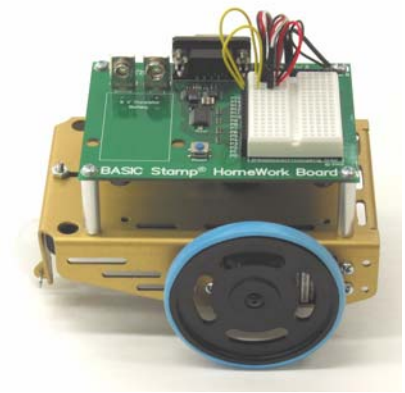
Figur 3-11
Boards befestigt am Boe-Bot Chassis

In Figur 3-12 sehen Sie die zusammengesetzten Boe-Bot, der linke mit dem Board of Education Rev C und der rechte mit einem HomeWork Board..

- √ Ziehen Sie alle überschüssigen Servo- und Batteriekabel von der Unterseite der Chassis durch das Loch mit der Gummidichtung.
- √ Verstauen Sie das überschüssige Kabel zwischen den Servos und der Chassis..



Mit dem Board of Education Rev C

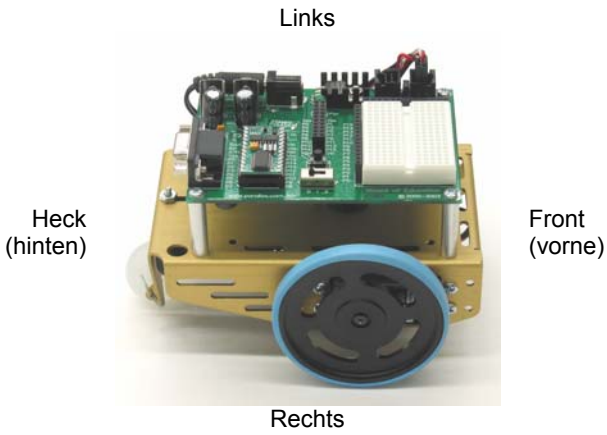


Mit dem HomeWork Board

Figur 3-12
Montierte Boe-Bots

AKTIVITÄT 2: ÜBERPRÜFEN DER SERVOS

In dieser Tätigkeit werden Sie testen ob die elektrischen Anschlüsse zwischen ihrem Board und den Servos korrekt ist. In Figur 3-13 sehen Sie die Vorder-, Hinter-, linke und rechte Seite ihres Boe-Bot. Sie versichern sich nun, dass das rechte Rad sich dreht, wenn es Impulse von P12 erhält, und das linke, wenn es Impulse von P13 erhält.



Figur 3-13
Vorne, Hinten, Rechts und Links am Boe-Bot

Test des rechten Rades

Das nächste Beispiel-Programm wird testen ob der Servo mit dem rechten Rad verbunden ist, Siehe Figur 3-14. Dieses Programm bewirkt, dass sich das Rad drei Sekunden lang im Uhrzeigersinn dreht, dann für eine Sekunde anhält, und sich dann drei Sekunden gegen den Uhrzeigersinn dreht.

Figur 3-14
Test des rechten Rades

Beispielprogramm: RightServoTest.bs2

√ Schliessen Sie ihre BASIC Stamp und die Servos an den Strom an.

Hinweise zum Anschliessen des Stroms an ihre BASIC Stamp und Servos..

- √ Wenn Sie ein Board of Education Rev C haben, schalten Sie den Dreifachschalter auf Position-2.
- √ Wenn Sie ein BASIC Stamp HomeWork Board haben, schliessen Sie die 9 V Batterie an den Batterieclip an und ersetzen Sie die Batterie, die vom Batteriepack entfernt wurde.

- √ Erfassen, speichern und starten Sie RightServoTest.bs2.
- √ Prüfen Sie nach, ob sich das rechte Rad drei Sekunden im Uhrzeigersinn dreht, für eine Sekunde anhält und dann drei Sekunden gegen den Uhrzeigersinn dreht.
- √ Wenn sich das rechte Rad/der rechte Servo nicht so verhält wie vorausgesagt, sehen Sie unter Servo Section nach. Es kommt gleich nach RightServoTest.bs2
- √ Wenn sich das rechte Rad/der rechte Servo richtig verhält, dann gehen Sie zum Abschnitt „Sie sind dran“, wo Sie das linke Rad testen.

```
' Robotics with the Boe-Bot - RightServoTest.bs2
' Right servo turns clockwise three seconds, stops 1 second, then
' counterclockwise three seconds.

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

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                        ' Signal start of program.

FOR counter = 1 TO 122                       ' Clockwise just under 3 seconds.
  PULSOUT 12, 650
  PAUSE 20
NEXT

FOR counter = 1 TO 40                        ' Stop one second.
  PULSOUT 12, 750
  PAUSE 20
NEXT

FOR counter = 1 TO 122                       ' Counterclockwise three seconds.
  PULSOUT 12, 850
  PAUSE 20
NEXT

END
```

Servo Fehlersuche: Hier ist eine Liste häufiger Probleme, und wie man sie behebt.

Der Servo dreht sich überhaupt nicht.

- √ Wenn Sie ein Board of Education Rev C benützen, versichern Sie sich, dass der 3-Positionen Schalter auf Position-2 ist. Danach starten Sie das Programm neu, indem Sie den Reset-Knopf drücken und wieder loslassen..
- √ Wenn Sie ein BASIC Stamp HomeWork Board benützen, versichern Sie sich, dass Batterien im Batteriepack sind.
- √ Überprüfen Sie ihre Servo-Verbindungen mit Figur 3-10 auf Seite 102 als Referenz. Wenn Sie ein HomeWork Board benützen, werfen Sie noch einen zweiten Blick auf Figur 2-18 auf Seite 67.
- √ Überprüfen Sie, dass Sie das Programm korrekt eingegeben haben.

Der rechte Servo dreht nicht, aber der linke.

Das bedeutet, dass die Servos vertauscht sind. Der Servo, der mit P12 verbunden ist, sollte an P13 angeschlossen sein, und umgekehrt.

- √ Schalten Sie den Strom aus..
- √ Ziehen Sie beide Servostecker aus.
- √ Schliessen Sie den Servo der an P12 angeschlossen war, an P13 an.
- √ Schliessen Sie den anderen Servo (der an P13 angeschlossen war) an P12 an
- √ Schalten Sie den Strom wieder ein.
- √ Starten Sie RightServoTest.bs2 nochmals

Das Rad stoppt nicht ganz, es dreht langsam.

Das bedeutet, dass der Servo nicht genau in der Mitte ist. Sie können meistens das Programm anpassen, damit der Servo still steht. Dies tun Sie indem Sie den `PULSOUT 12, 750` Befehl verändern.

- √ Wenn sich das Rad langsam im Gegenuhrzeigersinn dreht, benützen Sie ein value kleiner als 750.
- √ Wenn es sich im Uhrzeigersinn dreht, benützen Sie ein value grösser als 750.
- √ Wenn Sie ein Wert zwischen 740 und 760 finden, bei dem der Servo stoppt, dann versichern Sie sich, dass Sie ihn immer dann einsetzen wo sie den Befehl `PULSOUT 12, 750` sehen..

Das Rad pausiert keine Sekunde zwischen der Drehung im Uhrzeigersinn und der im Gegenuhrzeigersinn..

Es ist möglich, dass sich das Rad schnell für drei Sekunde in eine Richtung dreht und dann schnell für vier Sekunden in die andere. Es ist auch möglich, dass es schnell drei Sekunden dreht, dann langsamer für eine Sekunde und dann wieder schnell für drei Sekunden. Oder es dreht sich schnell für sieben Sekunden in die gleiche Richtung. Welches auch immer, es bedeutet, dass das Potentiometer nicht mehr angepasst ist.

- √ Entfernen Sie die Räder und die Servos und wiederholen Sie die Übungen in Kapitel 2 Aktivität 4: .

Sie sind dran: Test des linken Rades

Nun ist es an ihnen den gleichen Test am linken Rad zu vollziehen (siehe Figur 3-15). Dies beinhaltet Änderungen im RightServoTest.bs2, so dass die **PULSOUT** Befehl an den Servo, die an P13 angeschlossen sind geschickt werden, anstatt an den an P12 angeschlossenen.

Alles was Sie tun müssen, ist die drei **PULSOUT** Befehle so zu ändern, dass anstatt **PULSOUT 12** **PULSOUT 13** steht.

Figur 3-15
Test des linken Rades

- √ Speichern Sie RightServoTest.bs2 als LeftServoTest.bs2.
- √ Ändern Sie die drei **PULSOUT** Befehle, so dass **PULSOUT 13** statt **PULSOUT 12** steht..
- √ Speichern und Starten Sie das Programm.
- √ Versichern Sie sich, dass das Programm den linken Servo drei Sekunden lang im Uhrzeigersinn drehen, eine Sekunde stoppen und dann drei Sekunden im Gegenuhrzeigersinn drehen lässt.
- √ Wenn sich das linke Rad/der linke Servo nicht so verhält wie vorhergesagt, sehen Sie in der Servo section auf Seite 107 nach.
- √ Wenn sich das linke Rad richtig verhält, dann funktioniert ihr Boe-Bot richtig, und Sie können zur nächsten Aktivität weitergehen.

AKTIVITÄT 3: START/RESET-INDIKATOR SCHALTUNG UND PROGRAMM

Wenn die Spannungsversorgung unter den Pegel fällt, den ein Gerät braucht um richtig zu funktionieren, dann nennt man das einen "brownout". Die BASIC Stamp schützt sich selbst vor brownout, indem sie den Prozessor und die Programmspeicherchips einschlafen lässt, bis die Versorgungsspannung wieder auf normalem Level ist. Ein Spannungseinbruch unter 5.2 V bei Vin resultiert in einer Betriebsspannung unter 4.3 V

am internen Spannungsreglerausgang der BASIC Stamp. Eine Brownout Detektor Schaltung auf der BASIC Stamp überwacht diesen Fall. Wenn ein Brownout passiert, schaltet der Brownout Detektor den Prozessor und den Programmspeicher der BASIC Stamp ab.

Wenn die Spannungsversorgung wieder über 5.2 V ist, läuft die BASIC Stamp wieder, aber nicht an der gleichen Stelle im Programm. Stattdessen beginnt sie wieder am Anfang des Programms. Dies passiert auch, wenn Sie den Strom ausziehen und wieder einstecken und wenn Sie den Resetknopf drücken.

Wenn die Batterien des Boe-Bot leer werden, können Brownouts dazu führen, dass das Programm neu startet, wenn Sie es nicht erwarten. Das kann zu unerklärlichem Verhalten des Boe-Bot führen. In manchen Fällen läuft der Boe-Bot den programmierten Kurs und ganz plötzlich wirkt es, als hätte er sich verirrt und läuft in eine unerwartete Richtung. Wenn leer werdende Batterien der Grund sind, könnte es sein, dass das Programm wieder an den Anfang zurückging und von vorne startete. In anderen Fällen kann es passieren, dass der Boe-Bot einen konfusen Tanz aufführt, denn jedes Mal, wenn der Servo sich zu drehen beginnt, überlastet er die schon fast leeren Batterien. Das Programm versucht den Servo für den Bruchteil einer Sekunde zu drehen und startet dann neu, wieder und wieder..

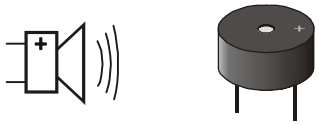
In dieser Situation ist ein Programmstart/reset Melder ein sehr nützliches Diagnosemittel und auch ein sehr nützliches Roboterwerkzeug. Eine Möglichkeit einen Neustart anzuzeigen ist, ein unmissverständliches Signal am Anfang des Boe-Bot-Programms einzufügen. Dieses Signal ertönt immer, wenn der Strom ausgesteckt wird, aber auch immer wenn ein Neustart nötig ist, weil die Umstände eines Brownout eingetroffen sind. Ein sehr effektives Signal für einen Neustart ist ein Lautsprecher, der einen Ton von sich gibt, jedes mal wenn das Programm von vorne beginnt, oder neu startet.

BASIC Stamp HomeWork Board Spezial Instruktionen

Obwohl der reset indicator ihnen mitteilt, wenn die 9 V Batterie, die den BASIC Stamp versorgt leer wird, teilt er ihnen nicht mit, wenn die Servoversorgung (das Batterienpack) leer wird.

Sie merken dass das Batteriepack leer wird, weil dann die Servos bei normalem Ablauf sich allmählich langsamer und langsamer bewegen werden Wenn Sie dieses Anzeichen bemerken, ersetzen Sie die leeren Batterien mit neuen 1.5 V Alkaline Batterien.

In dieser Aufgabe verwenden Sie ein Gerät namens Piezolautsprecher (piezospeaker), um Töne zu erzeugen. Dieser Lautsprecher kann verschiedene Töne produzieren, je nach der Frequenz der Hoch/Tief-Signale, die er vom BASIC Stamp empfängt. Das Schema-Symbol und die Bauteilzeichnung des Piezolautsprechers sehen Sie in Figur 3-16. Dieser Lautsprecher wird benutzt, um Töne auszugeben, wenn der BASIC Stamp in dieser Aktivität neu gestartet wird, aber auch in den anderen Aktivitäten in diesem Text.



Figur 3-16
Piezolautsprecher

Was ist Frequenz? Es ist die Masseinheit dafür, wie oft etwas in einer bestimmten Zeitspanne passiert.

Was ist ein Piezoelektrisches Element und wie kann es Töne erzeugen? Es ist ein Kristall, der sich ein wenig verbiegt, wenn man eine Spannung anlegt. Wenn der piezoelektrischen Kristall sehr schnell unter hohe und tiefe Spannung gesetzt wird, führt dies dazu, dass der piezoelektrische Kristall sich sehr schnell verbiegt oder vibriert. Vibrierende Objekte lassen die Luft um sie herum ebenfalls vibrieren. Und dies nimmt unser Ohr als Töne auf. Jede Vibrationsgeschwindigkeit hat einen anderen Ton. Zum Beispiel, wenn Sie an einer Gitarren-Saite zupfen, wird sie in einer bestimmten Frequenz vibrieren und Sie werden einen bestimmten Ton hören. Wenn Sie an einer anderen Gitarrensaite zupfen, wird sie in einer anderen Frequenz vibrieren und einen anderen Ton erzeugen.

Hinweis: Piezoelektrische Elemente können vielseitig eingesetzt werden. Wenn zum Beispiel auf ein piezoelektrisches Element Druck ausgeübt wird, kann es Spannung erzeugen. Manche piezoelektrischen Elemente haben eine Frequenz mit welcher sie natürlicherweise vibrieren. Diese werden verwendet um gewisse Frequenzen zu erzeugen, die dann als Uhrenoszillatoren und Taktgeber in vielen Computern und Mikrocontrollern vorkommen..

Benötigte Bauteile

(1) Zusammengebauter und getesteter Boe-Bot
(1) Piezolautsprecher
(div.) Verbindungsdrähte

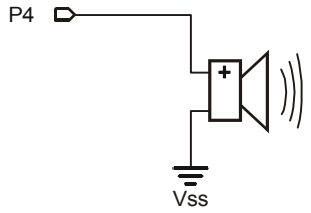
Aufbau der Start/Reset-Indikator Schaltung

In Figur 3-17 sehen Sie einen piezospeaker alarm circuit Schema für sowohl das Board of Education und das BASIC Stamp HomeWork Board. In Figur 3-18 sehen Sie ein Draht-Diagramm für jedes Board..

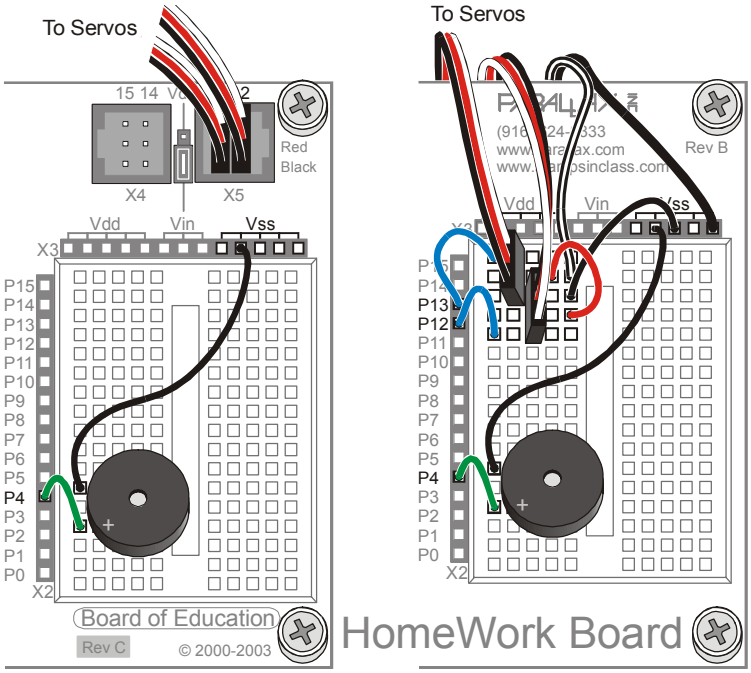
Schalten Sie immer den Strom aus, bevor Sie elektrische Schaltkreise aufbauen oder ändern!

- √ Wenn Sie ein Board of Education Rev C haben, schalten sie den Dreifachschalter auf Position 0.
- √ Wenn Sie ein BASIC Stamp HomeWork Board haben, entfernen Sie die 9 V Batterie vom Batterie-Clip und entfernen Sie eine Batterie aus dem Batterie-Pack.

√ Bauen Sie den Schaltkreis wie in Figur 3-17 und Figur 3-18 auf



Figur 3-17
Programm Start/ Reset Indikator Schaltung



Figur 3-18
Verdrahtungsdiagramm für die Programm-Start/Reset Indikator Schaltung

Board of Education (links) und HomeWork Board (rechts).

Der Piezolautsprecher und die Servo-Schaltkreise bleiben für alle weiteren Aktivitäten in diesem Text mit ihrem Board verbunden.

Alle weiteren Schaltschemata werden Schaltkreise zeigen, die zu den schon vorhandenen Servo- und Piezolautsprecher-Schaltkreisen hinzugefügt werden.

Alle Verdrahtungsdiagramme zeigen den Schaltkreis des Schemas, das gerade davor steht, zusammen mit den Servo- und Piezospeaker-Schaltkreisverbindungen.

Programmierung des Start/Reset-Indikators

Das nächste Beispiel-Programm testet den Piezolautsprecher. Dabei wird der **`FREQOUT`** Befehl verwendet um genau getimte hoch/tief-Signale an den Lautsprecher zu senden. Nachfolgend die Syntax des **`FREQOUT`** Befehls:

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

Hier ein Beispiel für einen **FREQOUT** Befehl, welches Sie im nächsten Beispielprogramm verwenden.

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

Das ***Pin*** Argument ist 4, das bedeutet, dass die High/Low-Signale an den I/O Pin P4 gesendet werden. Das ***Duration*** Argument, welches angibt wie lange die High/Low-Signale dauern, ist 2000, das bedeutet 2000 ms, oder 2 Sekunden. Das ***Freq1*** Argument ist die Frequenz der High/Low-Signale. In diesem Beispiel erzeugen die High/Low - Signale einen Ton von 3000 Hertz, oder 3 kHz.

Frequenz wird in Hertz gemessen (Hz). Das Hertz ist ein Frequenzmass, das angibt, wie oft pro Sekunde etwas passiert. Ein Hertz entspricht einmal pro Sekunde und wird mit 1 Hz abgekürzt. Ein Kilohertz ist tausendmal pro Sekunde und wird mit 1kHz abgekürzt.

FREQOUT erzeugt digitale Töne. Der **FREQOUT** Befehl erzeugt High/Low-Impulse verschiedener Länge, welche die Vibrationen des Piezolautsprechers den natürlichen Vibrationen einer Musik-Saite ähnlicher machen.

Beispielprogramm: StartResetIndicator.bs2

Dieses Beispiel-Programm erzeugt einen Pieps am Anfang des Programms, dann startet es ein Programm, das jede halbe Sekunde eine **DEBUG** Mitteilung sendet. Diese Nachricht wird unendlich weitergesendet, denn sie befindet sich zwischen einem **DO** und einem **LOOP**. Wenn die Stromzufuhr zum BASIC Stamp unterbrochen wird, während es in der Mitte eines **DO**…**LOOP** ist, beginnt das Programm wieder am Anfang. Wenn es dies tut, wird es wieder piepsen. Sie können die Umstände eines brownout simulieren, indem Sie entweder den Resetknopf drücken oder indem Sie die Batterieversorgung des Boards unterbrechen und wieder anschliessen.

- √ Schliessen Sie das Board wieder an den Strom an.
- √ Erfassen, speichern und Starten Sie StartResetIndicator.bs2.
- √ Überprüfen Sie, dass der Piezolautsprecher für zwei Sekunden einen gut hörbaren Ton von sich gibt, bevor die "Waiting for reset…" Nachrichten anfangen in ihrem Debug Terminal zu erscheinen.
- √ Wenn Sie gar keinen Ton hörten, überprüfen Sie die Verdrahtung und den Code auf Fehler. Wiederholen Sie dies, bis Sie einen gut hörbaren Ton von ihrem Lautsprecher bekommen..

√ Wenn Sie einen gut hörbaren Ton hörten, versuchen Sie die Umstände eines Brownout (langsamer Spannungsabfall) zu simulieren, indem Sie den Resetknopf auf ihrem Board drücken. Versichern Sie sich, dass der Piezolautsprecher einen gut hörbaren Ton nach jedem Neustart erzeugt.

√ Versuchen Sie ausserdem ihre Batterieversorgung zu unterbrechen und wieder anzuschliessen, und überprüfen Sie, dass auch dann ein Neustart-Warnungston ertönt.

```
' Robotics with the Boe-Bot - StartResetIndicator.bs2
' Test the piezospeaker circuit.

' {$STAMP BS2}                                  ' Stamp directive.
' {$PBASIC 2.5}                                 ' PBASIC directive.

DEBUG CLS, "Beep!!!"                           ' Display while speaker beeps.
FREQOUT 4, 2000, 3000                          ' Signal program start/reset.

DO                                             ' DO...LOOP
  DEBUG CR, "Waiting for reset…"               ' Display message
  PAUSE 500                                    ' every 0.5 seconds
LOOP                                           ' until hardware reset.
```

Wie das Programm funktioniert

StartResetIndicator.bs2 beginnt damit die Nachricht “Beep!!!” anzuzeigen. Sofort nach dem Ausdruck der Nachricht, spielt der **FREQOUT** Befehl einen 3 kHz-Ton auf dem Piezolautsprecher für 2 Sekunden. Weil die Instruktionen so schnell vom BASIC Stamp ausgeführt werden, sollte es scheinen, als ob die Nachricht im gleichen Moment angezeigt wird, wie der Piezolautsprecher den Ton zu spielen beginnt.

Wenn der Ton vorüber ist, beginnt das Programm die **DO**…**LOOP**Schleife, und zeigt die gleiche “Waiting for reset…” Nachricht immer und immer wieder. Jedes Mal, wenn der Resetknopf des Board of Education gedrückt wird oder die Stromzufuhr unterbrochen wird, beginnt das Programm wieder von vorne.

Sie sind dran: Fügen Sie StartResetIndicator.bs2 in ein anderes Programm ein

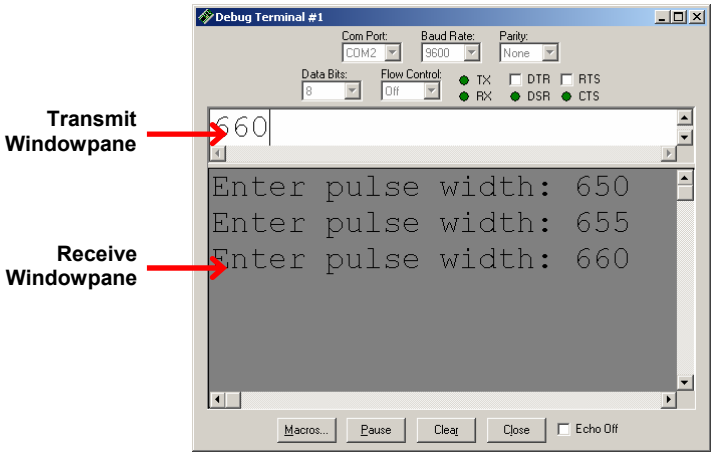
Die Codezeilen des Batterienmeldeprogramms werden von nun an am Anfang jedes Beispielprogramms verwendet. Sie können sie als Teil der "initialization routine" oder der "boot routine" jedes Boe-Bot-Programms sehen.

√ Kopieren Sie den **FREQOUT** Befehl des StartResetIndicator.bs2 und fügen Sie ihn in HelloOnceEverySecond.bs2 gerade oberhalb des **DO**...**LOOP** Abschnitts ein.

√ Lassen Sie das so veränderte Programm laufen und überprüfen Sie, dass es mit einem Warnton reagiert, jedes mal, wenn der BASIC Stamp neu startet (entweder weil der Resetknopf gedrückt wurde, oder weil die Batterienversorgung unterbrochen und wieder angeschlossen wurde).

AKTIVITÄT 4: VERTIEFUNGSTHEMA – SERVO TRANSFERKURVEN

In dieser Aktivität werden Sie eine Graphik erstellen, welche die Geschwindigkeit des Servos mit der Länge der Impulse vergleicht. Mit dem Transmit Windowpane (siehe Figur 3-19) des Debug Terminal können Sie den Prozess beschleunigen. Sie können das Transmit Windowpane gebrauchen, um dem BASIC Stamp Nachrichten zu senden. Sie können die Servo-Geschwindigkeit bei verschiedenen Impuls-Längen testen, indem Sie der BASIC Stamp Nachrichten schicken, in denen Sie ihr mitteilen, welche Impuls-Länge (**PULSOUT** ***Duration*** argument) er dem Servo übermitteln soll.



Figur 3-19
Debug Terminal Windowpanes

Verwendung des DEBUGIN Befehls

Inzwischen sind Sie mit dem **DEBUG** Befehl vertraut und wissen, wie Sie ihn verwenden um Meldungen vom BASIC Stamp ans Debug Terminal senden. Das Fenster in dem Sie die Meldungen sehen heisst Receive (Empfangs) Windowpane. Das Debug Terminal hat ebenfalls ein Transmit- (Übermittlungs) Windowpane, welches ihnen erlaubt dem BASIC

Stamp Meldungen zu senden, während ein Programm läuft. Um die Nachrichten die Sie eingeben zu empfangen, muss ihr Programm den **DEBUGIN** Befehl enthalten.

Der **DEBUGIN** Befehl platziert die Grösse, die sie im Transmit Windowpane eingeben in einer Variablen. Im nächsten Beispiel-Programm speichert eine Wort-Variable namens **pulseWidth** die Grössen, die der **DEBUGIN** Befehl enthält. Deshalb muss das Programm zuerst eine Deklaration dieser Variablen enthalten.

```
pulseWidth    VAR    Word
```

Nun kann der **DEBUGIN** Befehl benutzt werden, um eine Dezimalzahl, die Sie im Transmit Windowpane des Debug Terminals eingeben, entgegenzunehmen und sie in **pulseWidth** abzuspeichern. :

```
DEBUGIN DEC pulseWidth
```

Sie können die BASIC Stamp dann programmieren, diesen Wert zu benutzen. In diesem Beispiel wird er im ***Duration*** Argument des **PULSOUT** Befehls verwendet t:

```
PULSOUT 12, pulseWidth
```

Beispielprogramm: TestServoSpeed.bs2

Dieses Programm erlaubt es ihnen, das ***Duration*** Argument des **PULSOUT** Befehls zu bestimmen, indem Sie es im Transmit Windowpane des Debug Terminals eingeben.

- √ Erfassen, speichern und starten Sie TestServoSpeed.bs2.
- √ Zeigen Sie mit ihrer Maus auf das Transmit Windowpane des Debug Terminals, und klicken um in diesem Fenster zu schreiben.
- √ Geben Sie 650 ein und drücken Sie Enter..
- √ Versichern Sie sich, dass der Servo sich für sechs Sekunden mit voller Geschwindigkeit im Uhrzeigersinn dreht.

Wenn der Servo mit Drehen fertig ist, werden Sie aufgefordert eine andere Grösse einzugeben.

- √ Geben Sie 850 ein und drücken Sie Enter.

√ Versichern Sie sich, dass sich der Servo mit voller Geschwindigkeit im Gegenuhrzeigersinn dreht..

```
' Robotics with the Boe-Bot - TestServoSpeed.bs2
' Enter pulse width, then count revolutions of the wheel.
' The wheel will run for 6 seconds
' Multiply by 10 to get revolutions per minute (RPM).

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


counter        VAR     Word
pulseWidth     VAR     Word
pulseWidthComp VAR     Word

FREQOUT 4, 2000, 3000                        ' Signal program start/reset.

DO

  DEBUG "Enter pulse width: "

  DEBUGIN DEC pulseWidth

  pulseWidthComp = 1500 - pulseWidth

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

LOOP
```

Wie das Programm TestServoSpeed.bs2 funktioniert

Drei Variablen werden deklariert: **`counter`** für die **`FOR`**…**`NEXT`** Schleife, **`pulseWidth`** für die **`DEBUGIN`** und **`PULSOUT`** Befehle, und **`pulseWidthComp`** welche eine Dummy-Grösse aufbewahrt, welche in einem zweiten **`PULSOUT`** Befehl gebraucht wird.

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

Der **`FREQOUT`** Befehl wird verwendet, um den Start des Programms zu signalisieren..

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

Der Rest des Programms ist in einer **DO**…**LOOP** Schleife eingebettet, so dass es immer und immer wieder ausgeführt wird. Der Operator des Debug Terminals wird nach einer Puls-Länge gefragt. Diese ist in der **pulseWidth** Variablen gespeichert..


```
DEBUG "Enter pulse width: "

DEBUGIN DEC pulseWidth
```

Um die Messungen exakter zu machen, müssen zwei **PULSOUT** Befehle gesendet werden. Indem der eine **PULSOUT** Befehl gleich viel unter 750 ist, wie der andere darüber wird die Summe der zwei **PULSOUT** ***Duration*** Argumente immer 1500 sein. Dies sorgt dafür, dass die beiden **PULSOUT** Befehle zusammen gleich lange dauern. Das Resultat ist, dass unabhängig der ***Duration*** ihres **PULSOUT** Befehls, die **FOR**…**NEXT** Schleife immer gleich lange braucht um ausgeführt zu werden. Dies sorgt dafür, dass die **RPM** Messungen, die Sie in der "Ihre Aufgabe"-Sektion machen, genauer sein werden.

Dieser nächste Befehl wird die Impuls-Länge, die Sie eingeben nehmen, und eine Puls-Länge berechnen, die zu ihrer dazugerechnet 1500 ergibt. Wenn Sie eine Impuls-Länge von 650 eingeben, wird die **pulseWidthComp** 850 sein. Wenn Sie eine Impuls-Länge von 850 eingeben, wird die **pulseWidthComp** 650 sein. Wenn Sie eine Impuls-Länge von 700 eingeben, wird die **pulseWidthComp** 800 sein. Versuchen Sie einige weitere Beispiele. Sie werden zusammen immer 1500 ergeben..

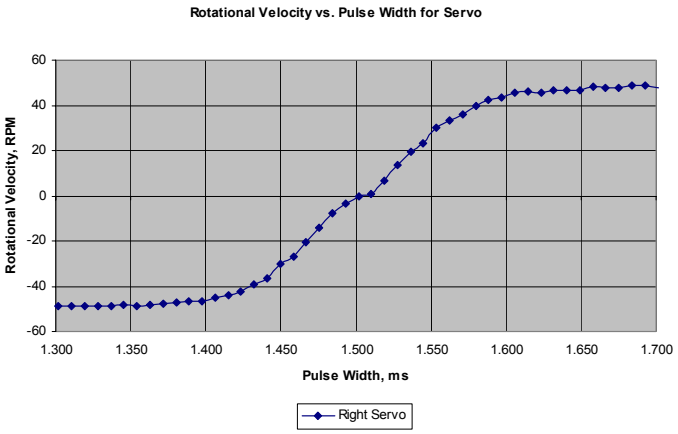
```
pulseWidthComp = 1500 - pulseWidth
```

Eine **FOR**…**NEXT** Schleife, die während 6 Sekunden läuft, sendet Impulse an den rechten Servo (P12). Die **pulseWidthComp** Grösse wird an den Piezolautsprecher gesendet und sorgt dafür, dass er einen schnellen, klickenden Ton von sich gibt. Es ist möglich, den Boe-Bot auf die Seite (auf sein sich nicht bewegendes Rad) zu legen wenn sie das sich bewegende Rad testen.

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

Your Turn – Graphing Pulse Width vs. Rotational Speed
Ihre Aufgabe – Kurve mit Impulslänge vs. Rotationsgeschwindigkeit

In Figur 3-20 sehen Sie ein Beispiel einer Transferkurve für einen Freilaufservo. Die horizontale Achse zeigt die Impuls-Länge in ms, die vertikale die Rotationsgeschwindigkeit in RPM. In dieser Graphik sind Drehungen im Uhrzeigersinn negativ, solche im Gegenuhrzeigersinn positiv. Die Übertragungskurve dieses bestimmten Servos reicht von etwa -48 RPM bis 48 RPM über einen Bereich von Test-Impuls-Längen von 1.3 ms bis 1.7 ms.



Figur 3-20
Transferkurve (Beispiel) für einen Parallax Servo

Sie können Table 3-1 benutzten, um die Daten ihrer eigenen Übertragungskurve aufzuschreiben.

Table 3-1: Pulse Width and RPM for Parallax Servo

Pulse Width (ms)	Rotational Velocity (RPM)	Pulse Width (ms)	Rotational Velocity (RPM)	Pulse Width (ms)	Rotational Velocity (RPM)	Pulse Width (ms)	Rotational Velocity (RPM)
1.300		1.400		1.500		1.600	
1.310		1.410		1.510		1.610	
1.320		1.420		1.520		1.620	
1.330		1.430		1.530		1.630	
1.340		1.440		1.540		1.640	
1.350		1.450		1.550		1.650	
1.360		1.460		1.560		1.660	
1.370		1.470		1.570		1.670	
1.380		1.480		1.580		1.680	
1.390		1.490		1.590		1.690	
						1.700	

Erinnern Sie sich daran, dass das ***Duration*** Argument des **PULSOUT** Befehls in 2 µs Einheiten ist. **PULSOUT 12, 650** schickt Impulse die 1.3 ms dauern an P12. **PULSOUT 12, 655** schickt Impulse die 1.31 ms dauern, **PULSOUT 12, 660** schickt Impulse die 1.32 Sekunden dauern, und so weiter.

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

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

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

- √ Markieren Sie ihr Rad, so dass Sie einen Referenzpunkt haben um die Drehungen zu zählen
- √ Starten Sie TestServoSpeed.bs2.
- √ Klicken Sie in das Transmit Windowpane des Debug Terminals.
- √ Geben Sie 650 ein.
- √ Zählen Sie, wie viele Drehungen ihr Rad machte.

Da sich der Servo für 6 Sekunden dreht, können Sie diese Zahl mit 10 multiplizieren und erhalten die Drehungen pro Minute (RPM=Revolutions per Minute).

- √ Multiplizieren Sie ihre Zahl mit 10 und schreiben Sie das Resultat in der Table 3-1 neben den Eintrag 1.3 ms.
- √ Geben Sie 655 ein.
- √ Zählen Sie wie viele Drehungen ihr Rad machte.
- √ Multiplizieren Sie mit 10 und tragen Sie es in der Table 3-1 neben 1.31 ms ein
- √ Erhöhen Sie weiterhin die Dauer um 5 (0.01 ms) bis Sie bei 850 (1.7 ms) sind
- √ Erstellen Sie nun mit einem Kalkulationsblatt, einem Taschenrechner oder einem Millimeterpapier eine Graphik ihrer Daten
- √ Wiederholen Sie diesen Prozess mit ihrem anderen Servo.

ZUSAMMENFASSUNG

In diesem Kapitel behandelten wir Montage und Test des Boe-Bot. Dies umfasste zunächst den mechanischen Aufbau, wie das Anschrauben der verschiedenen beweglichen Teile am Chassis des Boe-Bot. Dazu gehörte aber auch der Schaltungsbau und der Aufbau und Test der Piezo-Lautsprecher. Die Testphase umfasste auch die Nachkontrolle der Servos, nachdem diese für den Bau des Boe-Bot nochmals aus der Schaltung entfernt werden mussten. Ein Abschnitt für Fortgeschrittene führte in den **DEBUGIN** Befehl ein, welcher die Erzeugung von grafischen Datenpunkten für die Servo Transferkurven vereinfachte.

Das Konzept des Brownout (Spannungseinbruch) wurde eingeführt, zusammen mit der Betrachtung, was ein solcher für das auf der BASIC Stamp laufende Programm bedeutet. Ein Brownout veranlasst die BASIC Stamp sich abzuschalten und darauf das Programm neu zu starten. Darauf haben wir in die Schaltung einen Piezo-Lautsprecher eingebaut, der den Programmstart anzeigt. Wenn der Piezo-Lautsprecher mitten in einer Programmausführung ertönt, wo es nicht vorgesehen ist, kann dies auf einen Brownout hindeuten. Ein Brownout wiederum kann auf eine leere Batterie hindeuten. Damit der Piezo-Lautsprecher einen Ton erzeugt, wurde der **FREQOUT** Befehl eingeführt. Dieser ist Teil einer Initialisierungsroutine, welche wir in allen Boe-Bot Programmen verwenden werden.

Bis zu diesem Kapitel haben wir das Debug Terminal verwendet, um die von der BASIC Stamp zum Computer gesandten Meldungen anzuzeigen. Diese Meldungen wurden im Empfangs-Fenster (Receive windowpane) angezeigt. Das Debug Terminal hat auch ein Sende-Fenster (Transmit windowpane), das man verwenden kann, um Daten zur BASIC Stamp zu übermitteln. Die BASIC Stamp kann diese Daten mit der Ausführung des **DEBUGIN** Befehls übernehmen, welcher die vom Sende-Fenster übermittelten Werte in einer Variablen speichert. Die Werte können dann vom PBASIC Programm weiterverwendet werden. Diese Technik haben wir verwendet als Hilfe beim Sammeln der Daten, um die Transferkurve eines Freilaufservos aufzuzeichnen.

Fragen

1. Was ist ein Brownout?
2. Was sind Symptome eines Brownout beim Boe-Bot?

3. Wie kann ein Piezo-Lautsprecher eingesetzt werden, um einen Brownout zu erkennen?
4. Was ist ein Reset?
5. Was ist eine Initialisierungsroutine?
6. Wie erzeugt ein Piezo-Lautsprecher Töne?
7. Welches sind die Argumente im **FREQOUT** Befehl? Was bewirkt jedes der Argumente?
8. Nennen Sie drei (oder mehr) mögliche Fehler, die beim ausstecken und wieder anschliessen der Servos passieren können.
9. Welchen Befehl müssen Sie im Programm RightServoTest.bs2 ändern, um das linke statt das rechte Rad zu testen?
10. Was wird wohl passieren, wenn Sie versuchen, der BASIC Stamp mit dem Sende-Fenster des Debug Terminal Informationen zu senden, wenn auf der BASIC Stamp kein **DEBUGIN** Befehl auf einen Input wartet?
11. Was wird wohl passieren, wenn die BASIC Stamp einen **DEBUGIN** Befehl ausführt, aber keine Daten vom Sende-Fenster des Debug Terminal gesandt werden?

Übungen

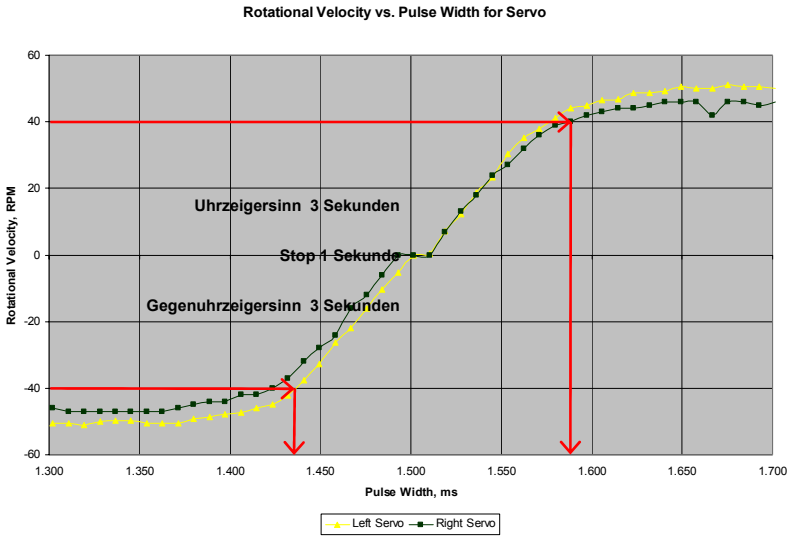
1. Schreiben Sie einen **FREQOUT** Befehl, der anders tönt, als der Reset Indikatorton, der das Ende eines Programms anzeigt.
2. Schreiben Sie einen **FREQOUT** Befehl, der einen Ton erzeugt (verschieden von den Anfangs- und Endtönen), der bedeutet, dass ein Zwischenschritt in einem Programm abgeschlossen wurde. Versuchen Sie es mit 100 ms Dauer und 4 kHz Frequenz.
3. Notieren Sie eine Variablendeklaration und einen **DEBUGIN** Befehl, der einen Wert in Ihre Variable lädt.

Projekte

1. Modifizieren Sie RightServoTest.bs2 so, dass es einen Ton erzeugt, der bedeutet, dass der Test abgeschlossen ist.
2. Modifizieren Sie RightServoTest.bs2 , dass es einen Ton erzeugt, der Sie jedes Mal warnt, wenn eine **FOR**…**NEXT** Schleife abgeschlossen ist.
3. Modifizieren Sie RightServoTest.bs2 so, dass Sie mit **DEBUGIN** sowohl die Pulsbreite für den linken und rechten Servo, als auch die Anzahl in der **FOR**…**NEXT** Schleife zu erzeugenden Pulse erfassen können. Benutzen Sie dieses

Programm, um die Bewegung Ihres Boe-Bot über das Sende-Fenster des Debug Terminal zu steuern.

4. *Für Fortgeschrittene* – Benutzen Sie die Transferkurven für den linken und rechten Servo um die **`PULSOUT Duration`** Argumente zu prognostizieren, welche den linken Servo mit einer bestimmten Geschwindigkeit in die eine, und den rechten Servo mit derselben Geschwindigkeit in die Gegenrichtung drehen lässt. Figur 3-21 zeigt ein Beispiel, wie das funktionieren kann, um den linken Servo mit 40 U/Min im Uhrzeigersinn und den rechten Servo mit 40 U/Min im Gegenuhrzeigersinn rotieren zu lassen. Beachten Sie, dass der linke Servo ungefähr 1.440 ms Pulse benötigt, während der rechte Servo 1.590 ms Pulse braucht. Sie können jeden dieser Werte mit 500 multiplizieren, um die **`PULSOUT Duration`** Argumente zu bestimmen. Versuchen Sie dies mit Ihrer eigenen Transferkurve und Ihren Servos.



Figur 3-21
Transfer Kurven

Zur Prognose der linken und rechten Pulsdauer

Die Servos in diesem Beispiel sind absichtlich unausgeglichen. Das hilft zu zeigen, dass physische Probleme, wie z.B. verschieden schnell drehende Servos, leicht durch eine Modifikation des steuernden PBASIC Programms überwunden werden können. Die Pulsbreite für jeden Servo kann ganz einfach mit dem ***Duration*** Wert in jedem **PULSOUT** Befehl gesteuert werden. Im Fall von Figur 3-21, lauten die PBASIC Befehle zum Abgleich der Servogeschwindigkeiten:

```
PULSOUT 13, 720     ' 1.44 X 500 = 720
PULSOUT 12, 795     ' 1.59 X 500 = 795
```

Kapitel 4: Boe-Bot Navigation

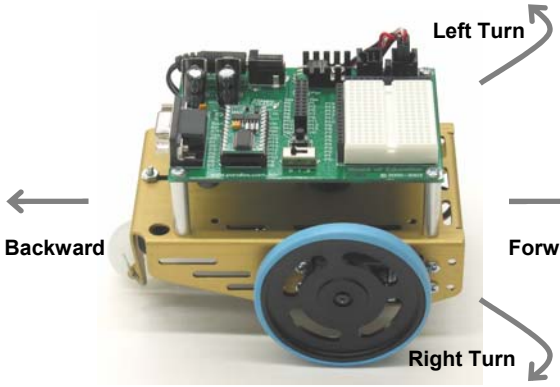
Der Boe-Bot kann für eine breite Auswahl verschiedener Bewegungen programmiert werden. Die Fahrmanöver und die Programmiertechniken, die wir hier zeigen, werden in späteren Kapiteln wieder verwendet. Der einzige Unterschied ist, dass der Boe-Bot in diesem Kapitel die Manöver blindlings ausführt. In späteren Kapiteln wird der Boe-Bot vergleichbare Manöver vornehmen als Reaktion auf äussere Bedingungen, die er mit seinen Sensoren entdeckt.

Dieses Kapitel zeigt auch in verschiedene Wege, wie die Navigation des Boe-Bot abgestimmt und kalibrieren kann. Ebenfalls eingeschlossen sind Techniken um Boe-Bot's gerade Linien zu strecken, präzisere Drehungen zu erreichen und Distanzen zu messen.

Aktivität	Überblick
1	Programmieren Sie den Boe-Bot so, dass er die Bewegungen Vorwärts, Rückwärts, Rechtsdrehung, Linksdrehung und Ortsdrehung (Pirouette) ausführen kann.
2	Stimmen Sie die Bewegungen von Aktivität 1 so ab, dass sie noch präziser sind.
3	Berechnen Sie die benötigte Anzahl Pulse, die der Boe-Bot benötigt, um eine vorbestimmte Distanz zurückzulegen.
4	Statt den Boe-Bot zu programmieren, dass er abrupte Starts und Stops macht, schreiben Sie Programme, die den Boe-Bot schrittweise in ein Manöver hinein beschleunigen und hinterher wieder langsam abbremsen.
5	Schreiben Sie Subroutinen für die elementaren Bewegungen, so dass Sie jede dieser Subroutinen immer wieder in einem Programm verwenden können.

ACTIVITY #1: ELEMENTARE BOE-BOT BEWEGUNGEN

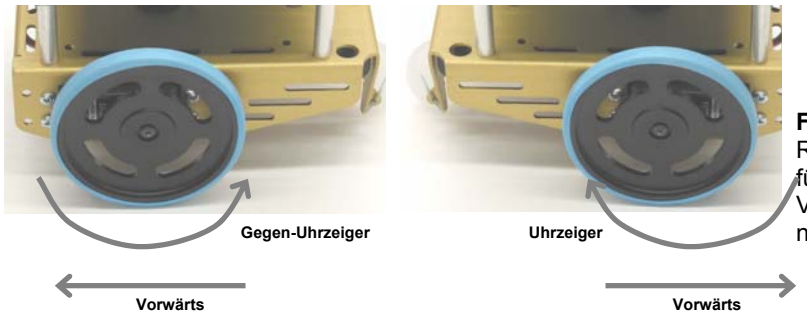
Figur 4-1 zeigt, welche Seite an Ihrem Boe-Bot vorne, hinten, links und rechts ist. Wenn sich der Boe-Bot vorwärts bewegt, müsste er zum rechten Rand der Seite rollen. Rückwärts wäre zum linken Rand der Seite. Eine Linksdrehung würde ihn beinahe über den oberen Rand hinaus fahren lassen und nach einer Rechtsdrehung würde er zum Fuss der Seite schauen.



Figur 4-1
Ihr Boe-Bot und die Fahrrichtungen

Vorwärts

Es ist schon drollig: Damit der Boe-Bot vorwärts rollt, muss das linke Rad des Boe-Bot im Gegenuhrzeigersinn drehen, und sein rechtes Rad im Uhrzeigersinn. Wenn Ihnen das noch etwas seltsam erscheint, werfen Sie einen Blick auf Figur 4-2 und versuchen Sie sich zu überzeugen, dass es wahr ist. Von Links gesehen muss das Rad im Gegenuhrzeigersinn drehen, damit der Boe-Bot sich vorwärts bewegt. Von Rechts gesehen muss das andere Rad sich im Uhrzeigersinn drehen, damit der Boe-Bot vorwärts kommt.



Figur 4-2
Radumdrehung für die Vorwärtsbewegung

Aus Kapitel 2 wissen Sie, dass das ***Duration*** Argument des **PULSOUT** Befehls die Geschwindigkeit und Richtung der Servodrehung steuert. Die ***StartValue*** und

EndValue Argumente einer **FOR…NEXT** Schleife steuern die Anzahl Pulse, die übermittelt werden. Da jeder Puls gleich lang ist, steuert das ***EndValue*** Argument such die Zeit, für die der Servo läuft. Hier ist ein Beispielprogramm, das den Boe-Bot für etwa drei Sekunden vorwärts fahren lässt.

Beispielprogramm: BoeBotForwardThreeSeconds.bs2

- √ Stellen Sie sicher, dass der Strom an der BASIC Stamp und den Servos angeschlossen ist.
- √ Erfassen, speichern und starten Sie BoeBotForwardThreeSeconds.bs2.

```
' Robotics with the Boe-Bot - BoeBotForwardThreeSeconds.bs2
' Make the Boe-Bot roll forward for three seconds.

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

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                        ' Signal program start/reset.

FOR counter = 1 TO 122

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

NEXT

END
```

Wie das Programm funktioniert

Aus Kapitel 2 haben Sie schon viel Erfahrung mit den Elementen dieses Programms: eine Variablendeklaration, eine **FOR…NEXT** Schleife, **PULSOUT** Befehle mit ***Pin*** und ***Duration*** Argumenten, und **PAUSE** Befehle. Hier ist eine Rekapitulation, was jeder Befehl bewirkt und dies für die Bewegung der Servos bedeutet.

Zuerst wird eine Variable deklariert, die in der **FOR..NEXT** Schleife benötigt wird.

```
counter VAR Word
```

Sie sollten diesen Befehl wiedererkennen, der einen Ton produziert, um den Programmstart anzuzeigen. Er wird in allen Programmen benutzt werden, welche die Servos laufen lassen.

```
FREQOUT 4, 2000, 3000            ' Signal program start/reset.
```

Diese **FOR**…**NEXT** Schleife sendet 122 Pulsserien an die Servos, je einen an P13 und P12, mit einer Pause von 20 ms nach jedem Durchlauf. Anschliessend kehrt das Programm zum Anfang der Schleife zurück.

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

PULSOUT 13, 850 lässt den linken Servo im Gegenuhrzeigersinn drehen, während **PULSOUT 12, 650** den rechten Servo im Uhrzeigersinn drehen lässt. Daher drehen sich beide Räder in Richtung des vorderen Endes des Boe-Bot, und damit fährt er vorwärts. Die **FOR**…**NEXT** Schleife braucht für die 122 Durchläufe etwa 3 Sekunden, also fährt der Boe-Bot etwa 3 Sekunden lang vorwärts.

Sie sind dran: Entfernung und Geschwindigkeit anpassen

- √ Durch Ändern des ***EndValue*** Arguments der **FOR**…**NEXT** Schleife von 122 auf 61 können Sie den Boe-Bot halb so lange vorwärts fahren lassen. Damit wird sich der Boe-Bot auch nur halb so weit bewegen.
- √ Speichern Sie BoeBotForwardThreeSeconds.bs2 unter einem neuen Namen.
- √ Ändern Sie das ***EndValue*** Arguments der **FOR**…**NEXT** Schleife von 122 auf 61.
- √ Starten Sie das Programm und überprüfen Sie, ob der Boe-Bot nur die halbe Zeit und die halbe Distanz fuhr.
- √ Versuchen Sie das ganze nochmals, aber diesmal ändern Sie das ***EndValue*** Argument der **FOR**…**NEXT** Schleife auf 244.

Die **PULSOUT** ***Duration*** Argumente von 650 und 850 liessen die Servos mit beinahe Maximalgeschwindigkeit rotieren. Wenn Sie die **PULSOUT** ***Duration*** Argumente näher zum Stillstandswert von 750 bringen, können Sie Ihren Boe-Bot verlangsamen.

- √ Modifizieren Sie Ihr Programm mit folgenden **PULSOUT** Befehlen:

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

√ Starten Sie das Programm und überzeugen Sie sich, dass sich Ihr Boe-Bot langsamer bewegt.

Rückwärtsfahren, Drehen und Schwenken

Das einzige was es braucht, um andere Bewegungsformen mit dem Boe-Bot machen zu können, sind verschiedene Kombinationen der **PULSOUT** ***Duration*** Argumente. Die folgenden zwei **PULSOUT** Befehle können zum Beispiel benutzt werden, um Ihren Boe-Bot rückwärts zu bewegen.:

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

Diese zwei Befehle lassen Ihren Boe-Bot nach links drehen (im Gegenuhrzeigersinn, wenn Sie von oben draufschauen):

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

Diese zwei Befehle lassen Ihren Boe-Bot nach rechts drehen (im Uhrzeigersinn, wenn Sie von oben draufschauen):

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

Sie können alle diese Befehle in einem einzigen Programm miteinander kombinieren, damit sich Ihr Boe-Bot vorwärts bewegt, nach links dreht, nach rechts dreht und schliesslich rückwärts fährt.

Beispielprogramm: ForwardLeftRightBackward.bs2

√ Erfassen, speichern und starten Sie ForwardLeftRightBackward.bs2.

TIP – Um dieses Programm schnell einzugeben, benützen Sie die Copy/Paste-Funktionen um vier **FOR**…**NEXT** Schleifen zu machen. Dann brauchen Sie nur noch die **PULSOUT** ***Duration*** Werte und die ***EndValues*** der **FOR**…**NEXT** Schleife anzupassen.

```
' Robotics with the Boe-Bot - ForwardLeftRightBackward.bs2
' Move forward, left, right, then backward for testing and tuning.

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

```
DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                         ' Signal program start/reset.


FOR counter = 1 TO 64                         ' Forward

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

NEXT

PAUSE 200

FOR counter = 1 TO 24                         ' Rotate left - about 1/4 turn

  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20

NEXT

PAUSE 200

FOR counter = 1 TO 24                         ' Rotate right - about 1/4 turn

  PULSOUT 13, 850
  PULSOUT 12, 850
  PAUSE 20

NEXT

PAUSE 200

FOR counter = 1 TO 64                         ' Backward

  PULSOUT 13, 650
  PULSOUT 12, 850
  PAUSE 20

NEXT

END
```

Sie sind dran: Schwenken

Sie können den Boe-Bot auch um eines seiner Räder schwenken lassen. Der Trick dabei ist, ein Rad stillzuhalten während das andere sich dreht. Wenn Sie z.B. das linke Rad anhalten und das rechte im Uhrzeigersinn (vorwärts) drehen, dann wird der Boe-Bot nach links schwenken.

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


Wenn Sie vorwärts nach rechts schwenken wollen, stoppen Sie einfach das rechte Rad und lassen Sie das linke Rad vorwärts (im Gegenuhrzeigersinn) drehen:

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

Sie sind die **PULSOUT** Befehle für Schwenken rückwärts und nach rechts:

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

Dies sind schliesslich die **PULSOUT** Befehle für Schwenken rückwärts und nach links:

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

- √ Speichern Sie ForwardLeftRightBackward.bs2 als PivotTests.bs2.
- √ Setzen Sie die oben diskutierten **PULSOUT** Befehle an Stelle der vorwärts, rechts, links und rückwärts Routinen.
- √ Passen Sie die Laufzeit für jedes Manöver an, indem Sie den ***EndValue*** aller **FOR**…**NEXT** Schleifen auf 30 setzen.
- √ Denken Sie auch daran, den Kommentar neben jeder **FOR**…**NEXT** Schleife anzupassen, damit er die neue Schwenkbewegung wiedergibt.
- √ Starten Sie das modifizierte Programm und überzeugen Sie sich, dass die verschiedenen Schwenkbewegungen funktionieren.

ACTIVITY 2: ABSTIMMEN DER GRUNDBEWEGUNGEN

Stellen Sie sich vor, Sie schreiben ein Programm, das den Boe-Bot 15 Sekunden lang mit Höchstgeschwindigkeit vorwärts fahren lässt. Was, wenn nun der Boe-Bot leicht nach rechts zieht, statt wie vorgesehen genau geradeaus? Es ist nicht notwendig, den Boe-Bot zu zerlegen und die Servos neu zu adjustieren um das zu korrigieren. Sie können einfach Ihr Programm ein bisschen anpassen, damit beide Räder des Boe-Bot genau gleich

schnell drehen. Während man den Schraubenzieher-Ansatz als “Hardware-Justierung” bezeichnen würde, nennt man den Ansatz über das Programm “Software-Justierung”.

Begradigen der Fahrstrecke

Der erste Schritt ist, den Kurs Ihres Boe-Bot genügend lange zu beobachten, um herauszufinden, ob er nach links oder nach rechts zieht, wenn er eigentlich geradeaus fahren sollte. Dazu sollten zehn Sekunden Fahrzeit ausreichen. Die kann mit einer einfachen Änderung an BoeBotForwardThreeSeconds.bs2 vom letzten Abschnitt erreicht werden.

Beispielprogramm: BoeBotForwardTenSeconds.bs2

- √ Öffnen Sie die Datei BoeBotForwardThreeSeconds.bs2.
- √ Ändern und Speichern Sie das Programm als BoeBotForwardTenSeconds.bs2.
- √ Andern Sie den ***EndValue*** des **FOR counter** von 122 auf 407, so dass es so ausschaut:

```
' Robotics with the Boe-Bot - BoeBotForwardTenSeconds.bs2
' Make the Boe-Bot roll forward for ten seconds.

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

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                        ' Signal program start/reset.

FOR counter = 1 TO 407                       ' Number of pulses - run time.

  PULSOUT 13, 850                            ' Left servo full speed ccw.
  PULSOUT 12, 650                            ' Right servo full speed cw.
  PAUSE 20

NEXT

END
```

- √ Starten Sie das Programm und beobachten Sie genau, ob der Boe-Bot in den zehn Sekunden nach links oder rechts driftet.

Sie sind dran – Anpassen der Servogeschwindigkeit zur Begradigung der Fahrstrecke des Boe-Bot

Wenn Ihr Boe-Bot perfekt geradeaus geht, probieren Sie dieses Beispiel trotzdem aus Wenn Sie sich an die Instruktionen halten, sollte Ihr Boe-Bot so adjustiert sein, dass er leicht nach rechts zieht.

Nehmen wir an, der Boe-Bot dreht leicht nach links. Das kann man auf zwei verschiedene Arten ansehen: entweder das linke Rad dreht zu langsam, oder das rechte Rad dreht zu schnell. Da der Boe-Bot mit Höchstgeschwindigkeit läuft, ist eine weitere Geschwindigkeitserhöhung des linken Rades nicht möglich, aber das rechte Rad zu bremsen sollte das Problem lösen.

Erinnern Sie sich, dass die Servogeschwindigkeit vom ***Duration*** Argument des **PULSOUT** Befehls bestimmt wird. Je näher die ***Duration*** an 750 ist, umso langsamer dreht der Servo. Das heisst, Sie sollten die 650 im Befehl **PULSOUT 12,650** in etwas ändern, das etwas näher an 750 ist. Wenn der Boe-Bot nur ein bisschen neben dem Kurs war, kann vielleicht **PULSOUT 12,663** bereits genügen. Wenn die Servos ernstlich unausgeglichen sind, braucht es vielleicht eher **PULSOUT 12,690**.

Möglicherweise brauchen Sie mehrere Anläufe, bis Sie den richtigen Wert finden. Nehmen wir an, Ihre erste Schätzung ist **PULSOUT 12,663**, aber es zeigt sich, dass das nicht genug ist, weil der Boe-Bot immer noch leicht links zieht. Also versuchen Sie **PULSOUT 12,670.** Eventuell haben Sie nun überkorrigiert, und es zeigt sich, dass **PULSOUT 12,665** genau richtig ist.

Wenn Ihr Boe-Bot nach rechts statt nach links zieht, so bedeutet das, dass Sie das linke Rad bremsen müssen, indem Sie die ***Duration*** von 850 im Befehl **PULSOUT 13,850** reduzieren. Auch hier, je näher der Wert an 750 liegt, desto langsamer dreht der Servo.

- √ Ändern Sie BoeBotForwardTenSeconds.bs2 so dass es Ihren Boe-Bot genau geradeaus fahren lässt.
- √ Benutzen Sie Klebeband oder eine Selbstklebeetikette um jeden Servo mit den besten **PULSOUT** Werten anzuschreiben.
- √ Wenn Ihr Boe-Bot bereits geradeaus fährt, versuchen Sie die gerade diskutierten Anpassungen um die Auswirkungen zu sehen. Es sollte den Boe-Bot in einer Kurve statt in einer geraden Linie fahren lassen.

Sie mögen feststellen, dass die Situation komplett verschieden ist, wenn Sie den Boe-Bot für die Rückwärtsfahrt programmieren.

- √ Ändern Sie BoeBotForwardTenSeconds.bs2 so dass es den Boe-Bot zehn Sekunden rückwärts rollen lässt.
- √ Wiederholen Sie den Test für gerade Linie.
- √ Wiederholen Sie die Schritte für die Korrektur des ***Duration*** Arguments des **PULSOUT** Befehls um die Rückwärtsfahrt des Boe-Bot zu begradigen.

Anpassen der Drehungen

Software Adjustierungen können auch gemacht werden, um den Boe-Bot genau in einem gewünschten Winkel drehen zu lassen, wie z.B. 90°. Die Zeitdauer, während welcher der Boe-Bot rotiert, bestimmt wie weit er dreht.
Weil die **FOR**…**NEXT** Schleife die Laufzeit steuert können Sie das ***EndValue*** Argument der **FOR**…**NEXT** Schleife so anpassen, dass Sie ganz schön nahe an den gewünschten Drehwinkel herankommen.

Hier ist die Routine für Rechtsdrehung von ForwardLeftRightBackward.bs2.

```
FOR counter = 1 TO 24         ' Rotate left - about 1/4 turn

  PULSOUT 13, 650
  PULSOUT 12, 650
  PAUSE 20

NEXT
```

Nehmen wir an, der Boe-Bot dreht grad ein bisschen mehr als 90° (1/4-Kreis). Versuchen Sie **FOR counter = 1 TO 23**, oder vielleicht sogar **FOR counter = 1 TO 22**. Wenn er nicht weit genug dreht, erhöhen Sie die Laufzeit der Rotation, indem Sie das ***EndValue*** Argument der **FOR**…**NEXT** Schleife auf den Wert erhöhen, der nötig ist, um die Vierteldrehung zu vollenden.

Wenn Sie feststellen, dass der eine Wert gerade etwas mehr als 90° , der andere etwas weniger als 90° dreht, versuchen Sie es mit dem Wert der etwas zu weit dreht, und verlangsamen Sie die Servos ein bisschen. Im Fall der Drehung links müssen dazu beide **PULSOUT** ***Duration*** Argumente von 650 ein bisschen gegen 750 geändert werden.. Wie schon bei der Übung mit der Geradeausfahrt wird dies ein iterativer Prozess sein.

Sie sind dran - 90° Drehungen

- √ Ändern Sie ForwardLeftRightBackward.bs2 so dass es 90° Drehungen bewirkt.
- √ Führen Sie ForwardLeftRightBackward.bs2 mit den **`PULSOUT`** Werten nach, die Sie für gerade Vorwärts- und Rückwärtsfahrt bestimmt haben.
- √ Ergänzen Sie die Etikette auf jedem Servo mit einem Hinweis auf den passenden **`EndValue`** für eine 90° Drehung.

Bitte erwarten Sie keine perfekten Ergebnisse, wenn Sie Ihren Boe-Bot auf einem Teppichboden laufen lassen! Ein Teppich ist ein bisschen wie ein Golfrasen – die Richtung, in der die Teppichfasern geneigt sind kann die Richtung beeinflussen, ich welche sich Ihr Boe-Bot bewegt, besonders über längere Distanzen. Für Präzisionsbewegungen ist eine glatte Oberfläche empfohlen.

AKTIVITÄT 3: DISTANZEN BERECHNEN

In vielen Robotik-Wettbewerben führt eine präzisere Navigation des Robots zu besseren Punktwerten. Ein beliebter Robotik-Wettbewerb für Einsteiger nennt sich “dead reckoning” (in Seemannssprache “Gissen”, deutsch: Koppelnavigation) . Das einzige Ziel dieses Wettbewerbs ist es, Ihren Robot zu einer oder mehreren Plätzen fahren zu lassen und dann auf genau den Ort zurückzukehren, wo er gestartet war.

Sie erinnern sich vielleicht, als Kind Ihre Eltern immer wieder diese Frage gestellt zu haben, wenn Sie unterwegs waren in die Ferien oder zu Verwandten:

> “Sind wir schon da?”

Als Sie etwas älter wurden und in der Schule dividieren lernten, haben Sie vielleicht die Strassentafeln beobachtet um zu sehen, wie weit es noch bis zum Zielort war. Dann haben Sie den Tachometer des Autos kontrolliert. Durch Division der Geschwindigkeit durch die Distanz erhielten Sie eine ziemlich gute Schätzung der verbleibenden Fahrzeit. Sie haben vielleicht nicht an genau diese Formel gedacht, aber da ist die Gleichung, die Sie benutzt haben.

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

Beispiel – Zeit für englische Distanz

Wenn Sie 140 Meilen von Ihrem Ziel entfernt sind und Sie 70 Meilen pro Stunde fahren, brauchen Sie 2 Stunden um anzukommen.

$$time = \frac{140\ miles}{70\ miles/hour}$$
$$= 140\ miles \times \frac{1\ hour}{70\ miles}$$
$$= 2\ hours$$

Beispiel – Zeit für metrische Distanz

Wenn Sie 200 Kilometer von Ihrem Ziel entfernt sind und Sie 100 Kilometer pro Stunde fahren, brauchen Sie 2 Stunden um anzukommen.

$$time = \frac{200\ kilometers}{100\ kilometers/hour}$$
$$= 200\ km \times \frac{1\ hour}{100\ km}$$
$$= 2\ hours$$

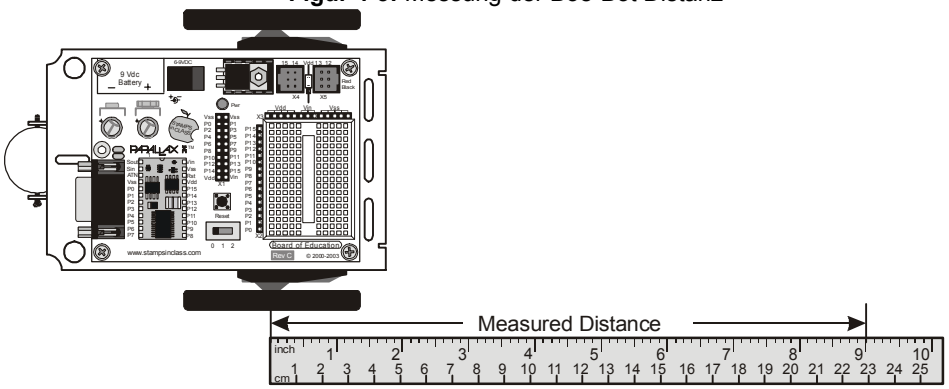
Sie können dieselbe Übung mit dem Boe-Bot machen, ausser dass Sie auch die Kontrolle darüber haben, wie weit das Ziel entfernt ist. Hier ist die Gleichung die Sie benutzen:

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

Sie müssen dazu die Geschwindigkeit des Boe-Bot ermitteln. Der einfachste Weg ist, den Boe-Bot neben einen Massstab zu setzen und ihn für eine Sekunde geradeaus fahren zu lassen. Wenn Sie nun messen, wie weit Ihr Boe-Bot gefahren ist, kennen Sie die Geschwindigkeit Ihres Boe-Bot. Wenn Ihr Massstab in Inches misst, haben Sie die Antwort in Inches pro Sekunde (in/s), wenn er Zentimeter zeigt, haben Sie Ihr Ergebnis in Zentimetern pro Sekunde (cm/s).

- √ Erfassen, speichern und starten Sie ForwardOneSecond.bs2.
- √ Setzen Sie Ihren Boe-Bot neben einen Massstab wie in Figur 4-3.
- √ Sorgen Sie dafür, dass der Punkt, wo das Rad den Boden berührt genau auf die 0 in/cm Markierung auf dem Massstab ausgerichtet ist.

Figur 4-3: Messung der Boe-Bot Distanz



√ Drücken Sie den Reset Knopf auf Ihrem Board um das Programm zu starten.
√ Messen Sie, wie weit Ihr Boe-Bot fuhr, indem Sie Messen, wo das Rad jetzt den Boden berührt und notieren das Ergebnis hier:_____ in / cm.

Beispielprogramm: ForwardOneSecond.bs2

```
' Robotics with the Boe-Bot - ForwardOneSecond.bs2
' Make the Boe-Bot roll forward for one second.

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

DEBUG "Program Running!"

counter        VAR     Word

FREQOUT 4, 2000, 3000                          ' Signal program start/reset.

FOR counter = 1 TO 41

  PULSOUT 13, 850
  PULSOUT 12, 650
  PAUSE 20

NEXT

END
```

Sie können die Distanz, die Sie eben gemessen haben, auch als Geschwindigkeit Ihres Boe-Bot in in/cm pro Sekunde auffassen. Nehmen wir an, Ihr Boe-Bot fuhr 9 in (23 cm). Da Ihr Boe-Bot für diese Distanz genau eine Sekunde benötigte, bedeutet das, dass Ihr Boe-Bot mit rund 9 in/s (23 cm/s) fährt. Daraus können Sie nun ausrechnen, wie viele Sekunden Ihr Boe-Bot fahren muss um eine bestimmte Distanz zurückzulegen.

Beispiel – Zeit für 20 in

Mit 9 in/s, muss Ihr Boe-Bot 2.22s unterwegs sein, um 20 in zurückzulegen.

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

Beispiel – Zeit für 51 cm

Mit 23 cm/s, muss Ihr Boe-Bot 2.22s unterwegs sein, um 51 cm zurückzulegen.

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

Um auszurechnen, wie viele Pulse Sie den Servos senden müssen, müssen Sie das Ergebnis mit 40.65 Pulse/Sekunde multiplizieren.

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

Die Berechnung in diesem Beispiel benötigt zwei Schritte. Zuerst finden wir heraus, wie lange die Servos laufen müssen, damit der Boe-Bot eine bestimmte Distanz zurücklegt. Dann berechnen wir, wie viele Pulse nötig sind, damit der Servo so lange läuft.. Da wir wissen, dass Sie mit 40.65 multiplizieren müssen, um von der Laufzeit zu der Anzahl Pulsen zu kommen, können wir das zu einem Schritt zusammenfassen.

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

Beispiel – Pulse für 20 in

Bei 9 in/s muss Ihr Boe-Bot 2.22 s lang fahren, um 20 in zurückzulegen.

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

Beispiel – Pulse für 51 cm

Bei 23 cm/s muss Ihr Boe-Bot 2.22 s lang fahren, um 51 cm zurückzulegen.

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

Sie sind dran – Distanz Ihres Boe-Bot

Nun wird es Zeit, dies mit Distanzen auszuprobieren, die Sie wählen.

- √ Wenn Sie das nicht schon gemacht haben, benutzen Sie einen Massstab und das Programm ForwardOneSecond.bs2 um die Geschwindigkeit Ihres Boe-Bot in in/s oder cm/s zu messen.
- √ Entscheiden Sie, wie weit Ihr Boe-Bot fahren soll.
- √ Benutzen Sie die Pulsgleichung um zu berechnen, wie viele Pulse den Servos übermittelt werden müssen:

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

- √ Modifizieren Sie BoeBotForwardOneSecond.bs2 so, dass es die Anzahl Pulse abgibt, die Sie für die gewählte Distanz berechnet haben.
- √ Lassen Sie das Programm laufen und überprüfen Sie, wie nahe sie gekommen sind.

Diese Technik hat Fehlerquellen. Die Aktivität, die Sie gerade abgeschlossen haben berücksichtigt nicht, dass es eine gewisse Anzahl Pulse braucht, bis Ihr Boe-Bot die Höchstgeschwindigkeit erreicht. Ebenso wenig hat es die Distanz berücksichtigt, die der Boe-Bot noch rollt, bevor er völlig zum Stillstand gekommen ist. Sie können das im Abschnitt Projekte in der Kapitelzusammenfassung ausprobieren..

AKTIVITÄT 4: BEWEGUNGEN - HOCHLAUFEN

Hochlaufen lassen (*ramping*) ist eine Möglichkeit, die Geschwindigkeit der Servos schrittweise zu erhöhen oder zu senken, statt abrupt zu starten und zu stoppen. Diese Technik kann die Lebensdauer sowohl der Batterien, wie auch der Servos verlängern.

Programmierung für das Hochlaufen

Der Schlüssel zum Hochlaufen ist, neben den Konstanten auch Variablen für das ***Duration*** Argument des **PULSOUT** Befehls zu benutzen. Figur 4-4 zeigt eine **FOR…NEXT** Schleife welche die Geschwindigkeit des Boe-Bot vom absoluten Stillstand bis zur Höchstgeschwindigkeit hochlaufen lässt. Jedes Mal wenn die **FOR…NEXT** Schleife sich wiederholt wird die **pulseCount** Variable um 1 erhöht. Beim ersten Durchlauf ist **pulseCount** 1, so dass es wie ein **PULSOUT 13, 751** und **PULSOUT 12, 749**.Befehl funktioniert. Beim zweiten Durchlauf ist der Wert von **pulseCount** 2, das entspricht den Befehlen **PULSOUT 13, 752** und **PULSOUT 12, 748**. In dem Masse wie die **pulseCount** Variable grösser wird, wird auch die Geschwindigkeit der Servos grösser. Beim Hundertsten Durchlauf ist die **pulseCount** Variable 100, entspricht also den Befehlen **PULSOUT 13, 850** and **PULSOUT 12, 650**, was der Höchstgeschwindigkeit für den Boe-Bot entspricht.

```
pulseCount    VAR     Word

FOR pulseCount = 1 TO 100

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

NEXT
```

1, 2, 3, …100

Figur 4-4
Beispiel für das Hochlaufen

Erinnern Sie sich aus Kapitel 2 Aktivität 5, dass eine **FOR…NEXT** Schleife auch Dekrementieren kann. Das heisst, von einer höheren zu einer tieferen Zahl herunter Zählen. Sie können dies verwenden, um die Geschwindigkeit wieder herunterzufahren, indem Sie analog **FOR pulseCount = 100 TO 1** eingeben. Hier ist ein Beispielprogramm das **FOR…NEXT** Schleifen verwendet um auf Höchstgeschwindigkeit hochzufahren und dann wieder herunterzufahren.

Beispielprogramm: StartAndStopWithRamping.bs2

√ Erfassen, Speichern und Starten Sie StartAndStopWithRamping.bs2.
√ Überzeugen Sie sich, dass der Boe-Bot schrittweise auf Höchstgeschwindigkeit beschleunigt, diese eine Weile beibehält und dann schrittweise bis zum Stillstand verlangsamt.


```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - StartAndStopWithRamping.bs2
' Ramp up, go forward, ramp down.

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

DEBUG "Program Running!"

pulseCount    VAR     Word                  ' FOR...NEXT loop counter.

' -----[ Initialization ]---------------------------------------------------

FREQOUT 4, 2000, 3000                       ' Signal program start/reset.

' -----[ Main Routine ]-----------------------------------------------------

' Ramp up forward.

FOR pulseCount = 1 TO 100                   ' Loop ramps up for 100 pulses.
  PULSOUT 13, 750 + pulseCount              ' Pulse = 1.5 ms + pulseCount.
  PULSOUT 12, 750 - pulseCount              ' Pulse = 1.5 ms – pulseCount.
  PAUSE 20                                  ' Pause for 20 ms.

NEXT

' Continue forward for 75 pulses.

FOR pulseCount = 1 TO 75                    ' Loop sends 75 forward pulses.
  PULSOUT 13, 850                           ' 1.7 ms pulse to left servo.
  PULSOUT 12, 650                           ' 1.3 ms pulse to right servo.
  PAUSE 20                                  ' Pause for 20 ms.
NEXT

' Ramp down from going forward to a full stop.

FOR pulseCount = 100 TO 1                   ' Loop ramps down for 100 pulses.
  PULSOUT 13, 750 + pulseCount              ' Pulse = 1.5 ms + pulseCount.
  PULSOUT 12, 750 - pulseCount              ' Pulse = 1.5 ms - pulseCount.
  PAUSE 20                                  ' Pause for 20 ms.
NEXT
```

```
END                                             ' Stop until reset.
```

Sie sind dran

Sie können auch Routinen aufbauen, die Hoch- und Runterfahren mit anderen Manövern kombinieren. Hier ist ein Beispiel für das Hochfahren bis Höchstgeschwindigkeit im Rückwärtsgang. Der einzige Unterschied zwischen dieser Routine und der Vorwärts-Hochfahren Routine ist, dass der Wert von **pulseCount** im **PULSOUT 13** Befehl von 750 subtrahiert wird, wo es vorher addiert wurde. Entsprechend wird **pulseCount** im **PULSOUT 12** Befehl zu 750 addiert, wo es vorher subtrahiert wurde.

```
' Ramp up to full speed going backwards

FOR pulseCount = 1 TO 100

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

NEXT
```

Sie können auch eine Routine für das Hochfahren in eine Drehung machen, indem Sie die Werte von **pulseCount** in beiden **PULSOUT** Befehlen zu 750 addieren. Indem Sie die Werte von **pulseCount** in beiden **PULSOUT** Befehlen von 750 subtrahieren, können Sie in eine Drehung in die andere Richtung hochfahren. Die Servos haben keine Chance auf Höchstgeschwindigkeit zu kommen, bevor sie wieder langsamer werden müssen.

```
' Ramp up right rotate.

FOR pulseCount = 0 TO 30

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

NEXT

' Ramp down right rotate

FOR pulseCount = 30 TO 0

  PULSOUT 13, 750 + pulseCount
  PULSOUT 12, 750 + pulseCount
```

```
    PAUSE 20

  NEXT
```

√ Öffnen Sie ForwardLeftRightBackward.bs2 aus Aktivität 1 und speichern Sie es als ForwardLeftRightBackwardRamping.bs2.

√ Modifizieren Sie das neue Programm so, dass Ihr Boe-Bot in jede Bewegung hoch- und herunterfährt. Hinweis: Sie könnten die Code Schnipsel oben verwenden, und ähnliche Schnipsel aus StartAndStopWithRamping.bs2.

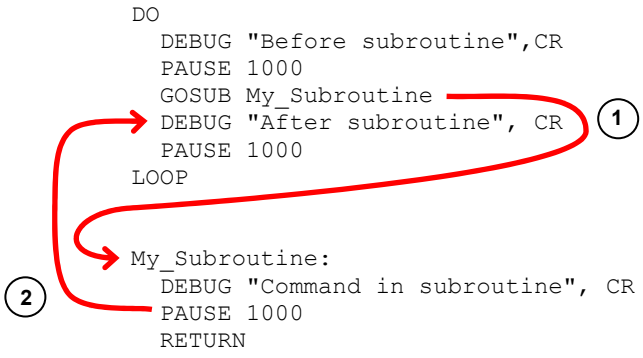
AKTIVITÄT 5: VEREINFACHUNG DER NAVIGATION MIT SUBROUTINEN

Im nächsten Kapitel muss Ihr Boe-Bot Ausweichmanöver zur Umgehung von Hindernissen machen müssen. Eines der Schlüsselelemente für die Umgehung von Hindernissen ist, vorprogrammierte Bewegungen auszuführen. Ein Weg, vorprogrammierte Bewegungen auszuführen sind Subroutinen. Diese Aktivität führt Subroutinen ein sowie zwei verschiedene Ansätze, um wiederverwendbare Bewegungen mit Subroutinen zu schaffen.

In der Subroutine

Eine PBASIC Subroutine hat zwei Teile. Ein Teil ist der Aufruf der Subroutine (subroutine call). Das ist der Befehl im Programm der es anweist, zu diesem wiederverwendbaren Codeabschnitt zu springen und wieder zurückzukommen, wenn das erledigt ist. Der andere Teil ist die Subroutine selbst. Sie beginnt mit einem Label (Bezeichner), das als Name dient und endet mit einem **`RETURN`** Befehl. Die Befehle zwischen dem Label und dem **`RETURN`** Befehl ist der Codeblock der die Arbeit ausführt, welche die Subroutine tun soll.

Figur 4-5 zeigt einen Teil eines PBASIC Programms das einen Subroutinenaufruf enthält und eine Subroutine. Der Aufruf der Subroutine ist der **`GOSUB My_Subroutine`** Befehl. Die Subroutine selbst ist alles ab dem **`My_Subroutine:`** Label bis zum **`RETURN`** Befehl. Und so funktioniert das. Wenn das Programm zum **`GOSUB My_Subroutine`** Befehl kommt, sucht es das **`My_Subroutine:`** Label. Wie mit Pfeil (1) gezeigt, springt das Programm zum **`My_Subroutine:`** Label und beginnt die Befehle auszuführen. Das Programm führt Zeile für Zeile vom Label abwärts aus, so dass Sie die Meldung "Command in subroutine" in Ihrem Debug Terminal sehen. **`PAUSE 1000`** bewirkt eine Pause von einer Sekunde. Dann, wenn das Programm den **`RETURN`** Befehl erreicht, zeigt Pfeil (2) wie es zu dem Befehl zurückspringt, der direkt nach dem **`GOSUB`** Befehl steht. In diesem Fall ist es ein **`DEBUG`** Befehl der die Meldung ausgibt "After subroutine".



Figur 4-5
Subroutinen Grundlagen

Beispielprogramm: OneSubroutine.bs2

√ Erfassen, speichern und starten Sie OneSubroutine.bs2

```
' Robotics with the Boe-Bot - OneSubroutine.bs2
' This program demonstrates a simple subroutine call.

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

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

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

√ Beobachten Sie Ihren Debug Terminal und drücken Sie einige male den Reset Knopf. Sie sollten jedes Mal die selbe Serie von drei Meldungen in der richtigen Reihenfolge erhalten.

Hier ist ein Beispiel, das zwei Subroutinen hat. Eine Subroutine erzeugt einen hohen Ton während das andere einen tiefen Ton erzeugt. Die Befehle zwischen **DO** und **LOOP** Rufen jede der Subroutinen abwechselnd auf. Probieren Sie das Programm aus und beachten Sie den Effekt.

Beispielprogramm: TwoSubroutines.bs2

√ Erfassen, speichern und starten Sie TwoSubroutines.bs2

```
' Robotics with the Boe-Bot - TwoSubroutines.bs2
' This program demonstrates that a subroutine is a reusable block of commands.

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

DO
  GOSUB High Pitch
  DEBUG "Back in main", CR
  PAUSE 1000
  GOSUB Low Pitch
  DEBUG "Back in main again", CR
  PAUSE 1000
  DEBUG "Repeat...",CR,CR
LOOP

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

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

Nun versuchen wir, die Vorwärts, Links, Rechts und Rückwärts-Navigation in Subroutinen zu verpacken. Hier ist ein Beispiel:

Beispielprogramm: MovementsWithSubroutines.bs2

√ Erfassen, speichern uns starten Sie MovementsWithSubroutines.bs2. Hinweis: Sie können das Edit Menu im BASIC Stamp Editor verwenden um Codeblocks von einem Programm in ein anderes zu übertragen (copy/paste).

```
' Robotics with the Boe-Bot - MovementsWithSubroutines.bs2
' Make forward, left, right, and backward movements in reusable subroutines.

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

DEBUG "Program Running!"

counter        VAR     Word
```

```
FREQOUT 4, 2000, 3000                          ' Signal program start/reset.


GOSUB Forward
GOSUB Left
GOSUB Right
GOSUB Backward

END

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

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

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

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

Sie sollten feststellen, dass das Bewegungsmuster des Boe-Bot dasselbe ist, wie bei ForwardLeftRightBackward.bs2. Klar gibt es viele verschiedene Arten ein Programm so

aufzubauen, dass es dieselben Bewegungen ergibt. Einen dritten Weg sehen Sie im folgenden Beispiel..

Beispielprogramm: MovementsWithVariablesAndOneSubroutine.bs2

Dies ist ein weiteres Beispielprogramm, das Ihren Boe-Bot dieselben Manöver ausführen lässt, aber es benutzt nur eine Subroutine sowie einige Variablen, um das zu erreichen.

Sie haben sicher schon gemerkt, dass bis hier jedes Boe-Bot Manöver mit ähnlichen Codeblocks erreicht wurde. Vergleichen Sie diese zwei Schnipsel:

```
' 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
```

Der Grund, dass diese zwei Codeblocks verschiedene Manöver ausführen sind Änderungen an den **FOR *StartValue*** und ***EndValue*** Argumenten und an den **PULSOUT *Duration*** Argumenten. Diese Argumente können Variablen sein, und diese Variablen können mehrfach während der Programmausführung geändert werden, um verschiedenen Bewegungen zu erzeugen. Statt separate Subroutinen mit spezifischen Argumenten zu verwenden, verwendet das Programm unten immer wieder dieselbe Subroutine. Allerdings speichert es einen neuen Wert in jede Variable, bevor es die Subroutine aufruft.

√ Erfassen, speichern und starten Sie MovementWithVariablesAndOneSubroutine.bs2.

```
' Robotics with the Boe-Bot - MovementWithVariablesAndOneSubroutine.bs2
' Make a navigation routine that accepts parameters.

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

DEBUG "Program Running!"

counter        VAR     Word
```

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

FREQOUT 4, 2000, 3000                        ' Signal program start/reset.

' Forward

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

' Left turn

pulseLeft = 650: pulseRight = 650: pulseCount = 24: GOSUB Navigate

' Right turn

pulseLeft = 850: pulseRight = 850: pulseCount = 24: GOSUB Navigate

' Backward

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

END

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

Hat Ihr Boe-Bot die bekannte Vorwärts-Links-Rechts-Rückwärts-Sequenz ausgeführt? Dieses Programm ist am Anfang vielleicht etwas schwierig zu verstehen, weil die Befehle in einer neuen Art aufgebaut sind. Statt jeden Variablenausdruck und jeden **GOSUB** Befehl auf einer neuen Zeile zu haben, sind diese auf einer Zeile gruppiert und durch Doppelpunkte getrennt. Hier funktionieren die Doppelpunkte wie die CR (Carriage Return = neue Zeile), die verschiedene PBASIC Befehle voneinander trennen. Die Verwendung von Doppelpunkten erlaubt es uns, alle neuen Variablenwerte für ein bestimmtes Manöver miteinander anzugeben, und auf derselben Zeile wie der Subroutinenaufruf.

Sie sind dran

√ Modifizieren Sie MovementWithVariablesAndOneSubroutine.bs2 so, dass Ihr Boe-Bot in einem Quadrat fährt, und zwar vorwärts auf den beiden ersten Seiten

und rückwärts auf den beiden letzten Seiten. Hinweis: Sie werden Ihr eigenes **`PULSOUT`** ***`EndValue`*** Argument benötigen, das Sie in Aktivität 2, Seite 135 ermittelt hatten.

AKTIVITÄT 6: AUFBAUTHEMA – KOMPLEXE MANÖVER IM EEPROM

Wenn Sie Ihr PBASIC Programm in Ihre BASIC Stamp laden, konvertiert der BASIC Stamp Editor Ihren Programmcode in numerische Werte, genannt "Token". Diese Tokens benutzt die BASIC Stamp als Befehle zur Programmausführung. Sie sind in einem der beiden kleineren schwarzen Chips oben auf Ihrer BASIC Stamp gespeichert, der mit "24LC16B" bezeichnet ist. Dieser Chip ist ein spezieller Typ von Computerspeicher (Memory), der EEPROM heisst. Das ist die Abkürzung für "electrically erasable programmable read only memory" (elektrisch löschbarer programmierbarer Nur-Lese-Speicher (EEPROM). Das EEPROM der BASIC Stamp kann 2048 Byte (2 kB) Informationen speichern. Was für den Programmspeicher nicht benutzt wird (dieser baut sich von der Adresse 2047 abwärts Richtung Adresse 0) kann für die Datenspeicherung benutzt werden (welche sich von Adresse 0 Richtung Adresse 2047 aufbaut).

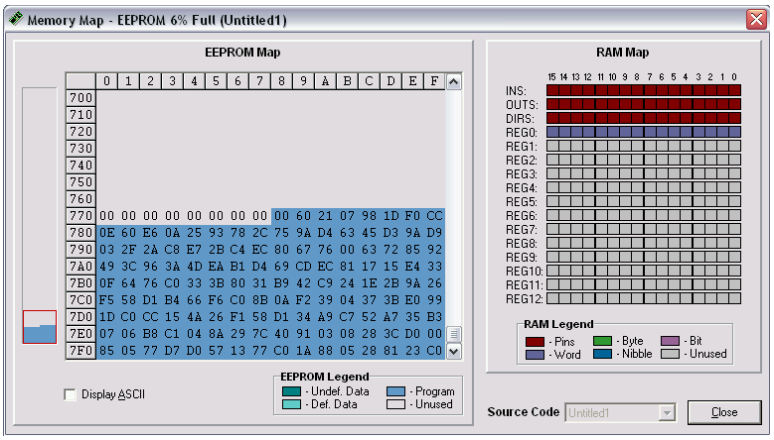
Wenn die Daten im EEPROM mit Ihrem Programm kollidieren wird das PBASIC Programm nicht richtig ausgeführt.

EEPROM Speicher unterscheiden sich vom Variablenspeicher RAM (random access memory = Frei adressierbarer Speicher) in mehrfacher Hinsicht:

- EEPROM benötigt mehr Zeit, um einen Wert zu speichern, manchmal bis zu mehreren Millisekunden.
- EEPROM erträgt nur eine begrenzte Anzahl Schreibvorgänge (etwa 10 Millionen). RAM kann unbeschränkt häufig wiederbeschrieben werden.
- Die Hauptaufgabe des EEPROM ist es, Programme zu speichern; Daten werden im übriggebliebenen Platz untergebracht.

Sie können Sich den Inhalt des EEPROM Ihrer BASIC Stamp im BASIC Stamp Editor anschauen, indem Sie Run klicken und Memory Map auswählen. Figur 4-6 zeigt die Memory Map (Speicherlandkarte) für MovementsWithSubroutines.bs2. Beachten Sie die verkleinerte EEPROM Map links im Bild. Der schattierte Bereich im kleinen Kasten zeigt die Menge von EEPROM die MovementsWithSubroutines.bs2 belegt.

Die hier gezeigten Memory Map Bilder stammen aus dem BASIC Stamp Editor v2.1. Wenn Sie eine frühere Version des BASIC Stamp Editor verwenden enthält Ihre Memory Map dieselben Informationen, aber sie ist anders aufbereitet.



Figur 4-6
BASIC Stamp Memory Map

Wenn wir schon dabei sind, beachten Sie auch, dass die **`counter`** Variable, die wir als Wort deklariert haben, im Register 0 der Memory Map zu sehen ist.

Das Programm mag beim Eintippen gross erschienen sein, aber es benötigt nur 136 der verfügbaren 2048 Byte des Programmspeichers. Vorläufig hat es also noch genug Platz für eine ziemlich lange Liste von Befehlen. Da ein Zeichen ein Byte im Speicher benötigt, hat es noch Platz für 1929 Ein-Buchstaben Richtungsbefehle.

EEPROM Navigation

Bisher haben wir drei verschiedene Programmieransätze verwendet, um Ihren Boe-Bot vorwärts, links, rechts und rückwärts fahren zu lassen. Jede dieser Techniken hat ihre Vorteile, aber alle würden ziemlich mühsam sein, wenn Sie mit Ihrem Boe-Bot eine länger, komplexere Serie von Bewegungen ausführen wollen. Die nächsten Programmbeispiele verwenden die inzwischen bekannten Codeblocks in Subroutinen für jede Elementarbewegung. Jede Bewegung erhält einen Ein-Buchstaben-Code als Referenz. Im EEPROM können lange Listen dieser Codebuchstaben gespeichert, und dann während der Programmausführung gelesen und decodiert werden. Das vermeidet die Mühe, lange Listen von Subroutinen wiederholen, oder die Variablen vor jedem **`GOSUB`** Befehl ändern zu müssen.

Diese Programmiertechnik benutzt ein paar neue PBASIC Befehle: Die **DATA** Directive, und die **READ** und **SELECT...CASE...ENDSELECT** Befehle. Wir wollen jedes davon näher betrachten, bevor wir sie an einem Beispiel ausprobieren.

Jede der Elementarbewegungen erhält einen Einbuchstaben-Code, der mit seiner Subroutine Korrespondiert: F für **Forward (vorwärts)**, B für **Backward (rückwärts)**, L für **Left_Turn (Linksdrehung)**, und R für **Right_Turn (Rechtsdrehung)**. Komplexe Boe-Bot Bewegungen können rasch choreographiert werden, indem man eine Kette dieser Codebuchstaben macht. Der letzte Buchstabe in der Kette ist ein Q, das heisst "quit" (verlassen), wenn die Bewegungen vorbei sind. Die Liste wird im EEPROM während des Programmdownloads gespeichert. Dazu verwenden wir die **DATA** Directive, die so aussieht:

```
DATA            "FLFFRBLBBQ"
```

Jeder Buchstabe ist in einem Byte des EEPROM gespeichert, beginnend mit Adresse 0 (es sei denn, wir befehlen ihm, anderswo zu starten). Der **READ** Befehl wird verwendet, um diese Liste während der Programmausführung wieder aus dem EEPROM zu holen. Die Werte können in einer **DO**…**LOOP** Schleife wie folgt gelesen werden:

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

Die **address** Variable ist der Speicherplatz jedes Byte im EEPROM der einen Codebuchstaben enthält. Die **instruction** Variable enthält den aktuellen Wert dieses Byte, also unseren Codebuchstaben. Beachten Sie, dass mit jedem Schleifendurchlauf der Wert der **address** Variable um eins erhöht wird. Damit kann man jeden Buchstaben von zusammenhängenden Speicherplätzen im EEPROM lesen, beginnend mit Adresse 0.

Ein Codeblock mit dem Befehl **SELECT...CASE...ENDSELECT** wird benutzt um eine Variable auszuwählen, sie schrittweise zu evaluieren und die passenden Codeblocks auszuführen. Hier ist der Codeblock, der jeden Buchstaben in der Variable **instruction** anschaut und dann die passende Subroutine für jedes Vorkommen eines bestimmten Buchstabens aufruft.

```
SELECT instruction
```

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

Der **DO…LOOP** Befehl hat optionale Instruktionen, die unter gewissen Umständen ganz nützlich sind. Der Befehl **DO...LOOP UNTIL (condition)** erlaubt einer Schleife zu drehen bis eine bestimmte Bedingung (condition) erreicht ist. **DO WHILE (condition) …LOOP** erlaubt einer Schleife nur solange zu laufen, als eine bestimmte Bedingung noch existiert. Unser Beispielprogramm benutzt **DO…LOOP UNTIL (condition)**. Dies lässt die **DO…LOOP** Schleife solange repetieren, bis der Buchstabe "Q" vom EEPROM gelesen wird.

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

Hier sind diese Konzepte alle zusammen in einem einzigen Programm.

Beispielprogramm: EepromNavigation.bs2

- √ Lesen Sie bitte sorgfältig die Instruktionen und Kommentare zum Code in EepromNavigation.bs2, damit Sie gut verstehen, was jeder Teil des Programms macht.
- √ Erfassen, speichern und starten Sie EepromNavigation.bs2.

```
' Robotics with the Boe-Bot - EepromNavigation.bs2
' Navigate using characters stored in EEPROM.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

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

pulseCount     VAR     Word                ' Stores number of pulses.
address        VAR     Byte                ' Stores EEPROM address.
instruction    VAR     Byte                ' Stores EEPROM instruction.

' -----[ EEPROM Data ]-----------------------------------------------------

'      Address: 0123456789                 ' These two commented lines show
```

```
'               ||||||||||                      ' EEPROM address of each datum.
DATA           "FLFFRBLBBQ"                     ' Navigation instructions.

' -----[ Initialization ]---------------------------------------------------

FREQOUT 4, 2000, 3000                           ' Signal program start/reset.

' -----[ Main Routine ]------------------------------------------------------

DO

  READ address, instruction                     ' Data at address in instruction.
  address = address + 1                         ' Add 1 to address for next read.

  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                                             ' Stop executing until reset.

' -----[ Subroutine - Forward ]----------------------------------------------

Forward:                                        ' Forward subroutine.
  FOR pulseCount = 1 TO 64                      ' Send 64 forward pulses.
    PULSOUT 13, 850                             ' 1.7 ms pulse to left servo.
    PULSOUT 12, 650                             ' 1.3 ms pulse to right servo.
    PAUSE 20                                    ' Pause for 20 ms.
  NEXT
  RETURN                                        ' Return to Main Routine loop.

' -----[ Subroutine - Backward ]---------------------------------------------

Backward:                                       ' Backward subroutine.
  FOR pulseCount = 1 TO 64                      ' Send 64 backward pulses.
    PULSOUT 13, 650                             ' 1.3 ms pulse to left servo.
    PULSOUT 12, 850                             ' 1.7 ms pulse to right servo.
    PAUSE 20                                    ' Pause for 20 ms.
  NEXT
  RETURN                                        ' Return to Main Routine loop.

' -----[ Subroutine - Left Turn ]--------------------------------------------

Left_Turn:                                      ' Left turn subroutine.
  FOR pulseCount = 1 TO 24                      ' Send 24 left rotate pulses.
    PULSOUT 13, 650                             ' 1.3 ms pulse to left servo.
    PULSOUT 12, 650                             ' 1.3 ms pulse to right servo.
```

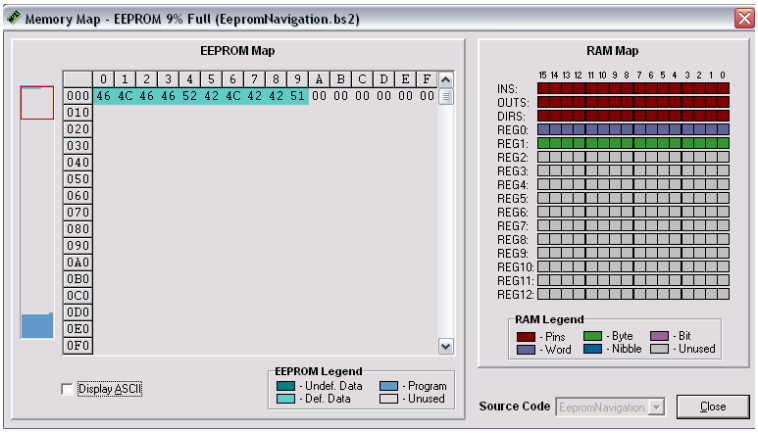
```
    PAUSE 20                                    ' Pause for 20 ms.
  NEXT
  RETURN                                        ' Return to Main Routine loop.

' -----[ Subroutine - Right Turn ]-------------------------------------------

Right_Turn:                                     ' right turn subroutine.
  FOR pulseCount = 1 TO 24                      ' Send 24 right rotate pulses.
    PULSOUT 13, 850                             ' 1.7 ms pulse to left servo.
    PULSOUT 12, 850                             ' 1.7 ms pulse to right servo.
    PAUSE 20                                    ' Pause for 20 ms.
  NEXT
  RETURN                                        ' Return to Main Routine section.
```

Hat Ihr Boe-Bot ein Rechteck abgefahren, vorwärts auf den ersten beiden Seiten und rückwärts auf den letzten beiden? Wenn es mehr wie ein Trapezoid aussah, sollten Sie die **`FOR pulseCount EndValue`** Argumente in den Drehungen-Subroutinen anpassen, um präzise 90-Grad Drehungen zu erhalten.

Sie sind dran

√ Klicken Sie mit dem EepromNavigation.bs2 aktiv im BASIC Stamp Editor Run and wählen Sie Memory Map.

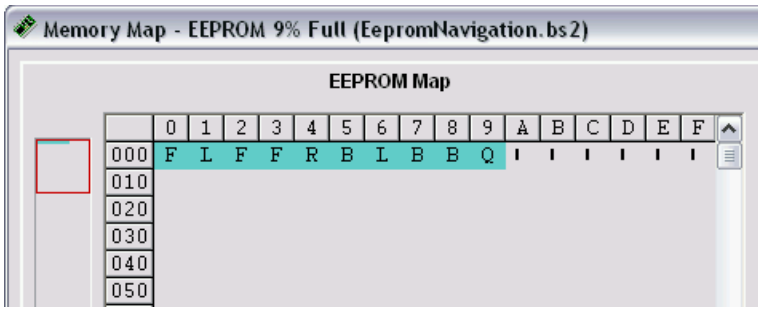
Die gespeicherten Befehle erscheinen blau hervorgehoben am Anfang der Detail EEPROM Map wie in Figur 4-7. Die Zahlen erscheinen in hexadezimalen ASCII (American Standard Code for Information Interchange) Codes, die den Buchstaben entsprechen, die Sie in Ihrem Data Statement eingegeben haben.



Figur 4-7
Memory Map mit gespeicherten Befehlen, angezeigt in der EEPROM Map

√ Klicken Sie auf die Display ASCII Checkbox bei der unteren linken Ecke des Memory Map Windows.

Nun erscheinen die Richtungsbefehle in einem vertrauteren Format. (Figur 4-8). Statt als ASCII Codes erscheinen Sie als die Buchstaben, die sie mit der **DATA** Direktive.effektiv erfasst haben.



Figur 4-8
Vergrösserung der detaillierten EEPROM Map nachdem die "Display ASCII" Box angekreuzt wurde.

Dieses Programm hat Total 10 Buchstaben im EEPROM gespeichert. Diese zehn Buchstaben wurden von der **address** Variable des **READ** Befehls adressiert. Die **address** Variable war als Byte deklariert, d.h. sie kann bis zu 256 Speicherplätze adressieren, viel mehr als die 10, die wir benötigen. Wenn die **address** Variable neu als Word Variable deklariert würde, könnten Sie theoretisch bis 65535 gehen, weit mehr als Speicherplätze verfügbar sind. Denken Sie daran, dass wenn Ihr Programm grösser wird, wird die Zahl der verfügbaren EEPROM Adressen für Datenspeicherung kleiner.

Sie können die bestehende Zeichenkette für eine neue Sequenz von Bewegungen abändern. Sie können auch zusätzliche **DATA** Statements ergänzen. Die Daten werden sequentiell gespeichert, so dass das erste Zeichen der zweiten Datenkette unmittelbar hinter dem letzten Zeichen der ersten Datenkette gespeichert wird..

√ Verändern Sie die Zeichenkette in der **DATA** Direktive durch ändern, löschen und hinzufügen von Zeichen, und starten Sie das Programm neu. Denken Sie daran, dass das letzte Zeichen in der **DATA** Direktive immer ein "Q" sein sollte.

√ Ändern Sie die **DATA** Direktive so, dass Ihr Boe-Bot die bekannte Serie von Vorwärts-, Links-, Rechts-, Rückwärts-Bewegungen ausführt.

√ Addieren Sie mal eine zweite **DATA** Direktive. Denken Sie daran, das "Q" vom Ende der ersten **DATA** Direktive zu entfernen und es am Ende der zweiten hinzuzufügen. Sonst wird das Programm nur die Befehle in der ersten **DATA** Direktive ausführen.

Beispielprogramm: EepromNavigationWithWordValues.bs2

Das nächste Beispielprogramm sieht am Anfang ziemlich kompliziert aus, aber es zeigt einen höchst effizienten Weg um Programme für massgeschneiderte Boe-Bot Choreographie aufzubauen. Dieses Beispielprogramm nutzt den EEPROM Datenspeicher, aber benutzt keine Subroutinen. Statt dessen wird ein einziger Codeblock benutzt, mit Variablen an Stelle der **FOR**…***EndValue*** und **PULSOUT** ***Duration*** Argumente. Diese Variablen werden mit den Werten aus dem EEPROM verändert.

Im Normalfall speichert die **DATA** Direktive die Informationen in Bytes im EEPROM. Um Word-grosse Datenelemente zu speichern, können Sie den **Word** Modifier zur **DATA** Direktive vor jedem Datenelement in Ihrer Zeichenkette hinzufügen. Jede Word-grosse Datenelement belegt zwei Byte des EEPROM Speichers. Damit wird auf die Daten über jede zweite Speicheradresse zugegriffen. Wenn man mehr als eine **DATA** Direktive benutzt, ist es äusserst praktisch, jeder ein eigenes Label zuzuweisen. So können Ihre **READ** Befehle die Labels referenzieren um Datenelemente auszulesen, ohne dass Sie herausfinden müssen, an welcher EEPROM-Adresse jeder Satz von Datenelementen beginnt. Schauen Sie sich mal den folgenden Codeausschnitt an:

```
' 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
```

Jedes der drei **DATA** Statements beginnt mit seinem eigenen Label. Der **Word** Modifier kommt vor jedem Datenelement, und die Elemente sind durch Kommata getrennt. Diese drei Ketten von Daten werden im EEPROM eine nach der anderen gespeichert. Wir können auf die Berechnungen der Adressnummer jedes Datenelementes verzichten, weil die Labels und die **addressOffset** Variable das automatisch tun. Der **READ** Befehl benutzt jedes Label um die EEPROM-Adresse zu bestimmen, an der diese Kette beginnt, und benutzt dann den Wert der **addressOffset** Variablen, um zu wissen, um wie viele Adresszahlen er verschieben muss, um das Element in der Kette zu finden. Die an dieser Adresse gefundenen Daten werden in die Wort Variable am Ende des **READ** Befehls übertragen. Nun schauen Sie sich diesen Codeausschnitt an:

```
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
```


Beim ersten Schleifendurchlauf ist **addressOffset** = 0. Der erste **READ** Befehl liest einen Wert von 64 von der ersten Adresse des **Pulses_Count** Label, und überträgt ihn in die **pulseCount** Variable. Der zweite **READ** Befehl liest einen Wert von 850 von der ersten Adresse, die durch das **Pulses_Left** Label bezeichnet ist, and überträgt ihn in die **pulseLeft** Variable. Der dritte **READ** Befehl liest einen Wert von 650 aus der ersten Adresse, die durch das Label **Pulses_Right** bezeichnet ist und überträgt ihn in die **pulseRight** Variable. Beachten Sie, dass das die drei Werte in der "0" Spalte des Codeausschnitts von Seite 157 sind. Wenn die Werte dieser Variablen in den folgenden Codeblock eingesetzt werden, erhalten wir folgendes:

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

wird

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

Erkennen Sie die Grundbewegung, die von diesem Codeblock erzeugt wird?

- √ Betrachten Sie die übrigen Spalten des Codeausschnitts auf Seite 157 und überlegen Sie, wie der **FOR**…**NEXT** Codeblock beim zweiten, dritten und vierten Schleifendurchlauf aussehen wird.
- √ Betrachten Sie die vorletzte Codezeile im Programm unten. Was passiert beim fünften Schleifendurchlauf?
- √ Erfassen, speichern und starten Sie EepromNavigationWithWordValues.bs2.

```
' Robotics with the Boe-Bot - EepromNavigationWithWordValues.bs2
' Store lists of word values that dictate.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.
```

```
DEBUG "Program Running!"

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

counter        VAR     Word
pulseCount     VAR     Word                  ' Stores number of pulses.
addressOffset  VAR     Byte                  ' Stores offset from label.
instruction    VAR     Byte                  ' Stores EEPROM instruction.
pulseRight     VAR     Word                  ' Stores servo pulse widths.
pulseLeft      VAR     Word

' -----[ EEPROM Data ]----------------------------------------------------

' 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

' -----[ Initialization ]-------------------------------------------------

FREQOUT 4, 2000, 3000                        ' Signal program start/reset.

' -----[ Main Routine ]---------------------------------------------------

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                                          ' Stop executing until reset.
```

Hat Ihr Boe-Bot die bekannten Vorwärts-, Links-, Rechts-, Rückwärtsbewegungen ausgeführt? Sind sie diese Bewegungen nun endgültig leid? Möchten Sie Ihren Boe-Bot endlich mal was anderes machen sehen, oder möchten Sie Ihre Eigene Routine choreographieren?

Sie sind dran – Machen Sie Ihre eigenen massgeschneiderten Navigationsroutinen

- √ Speichern Sie EepromNavigationWithWordValues.bs2. unter einem neuen Namen.
- √ Ersetzen Sie die **DATA** Direktiven mit denen unten.
- √ Starten Sie das modifizierte Programm und schauen Sie, was Ihr Boe-Bot macht.


```
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
```

- √ Erstellen Sie eine Tabelle mit drei Zeilen, eine für jede **DATA** Direktive, und je einer Spalte für jede Boe-Bot Bewegung, die sie machen wollen, plus eine für das **Word 0** Element in der **Pulses_Count** Zeile.
- √ Nutzen Sie die Tabelle um die Choreographie für Ihren Boe-Bot auszuarbeiten, indem Sie die **FOR**…***EndValue*** und **PULSOUT** ***Duration*** Argumente eintragen, die Sie für den Codeblock jeder Bewegung brauchen.
- √ Modifizieren Sie Ihr Programm mit den neu zusammengestellten **DATA** Direktiven.
- √ Erfassen, speichern und starten Sie Ihr massgeschneidertes Programm. Hat Ihr Boe-Bot das getan, was Sie wollten? Arbeiten Sie daran bis er es tut.

ZUSAMMENFASSUNG

Dieses Kapitel führte die grundlegenden Bewegungen des Boe-Bot ein: Vorwärts, Rückwärts, Rechtsdrehung, Linksdrehung und Drehung an Ort. Die Art der Bewegung wird durch die ***Duration*** Argumente des **PULSOUT** Befehls bestimmt. Wie weit die Bewegung geht wird durch die ***StartValue*** und ***EndValue*** Argumente der **FOR…NEXT** Schleife bestimmt.

Kapitel 2 zeigte die Hardware-Adjustierung, d.h. das physische zentrieren der Servos mit einem Schraubenzieher. Dieses Kapitel konzentrierte sich auf das Finetuning durch Anpassung der Software. Konkret wurde die Differenz der Rotationsgeschwindigkeiten der zwei Servos kompensiert, indem das ***Duration*** Argument des **PULSOUT** Befehls vom schnelleren der beiden Servos angepasst wurde. Dies kann die Spur des Boe-Bot von einer Kurve zu einer geraden Linie ändern, wenn die beiden Servos nicht perfekt abgeglichen sind. Um das Tuning so zu verfeinern, dass der Boe-Bot sich genau im gewünschten Winkel dreht, können die ***StartValue*** und ***EndValue*** Argumente der jeweiligen **FOR…NEXT** Schleife adjustiert werden.

Damit der Boe-Bot eine vorherbestimmte Strecke zurücklegt, kann mit einem Massstab die Strecke gemessen werden, die er in einer Sekunde zurücklegt. Ausgehend von dieser gemessenen Strecke und der Anzahl Pulse pro Sekunde Laufzeit kann man die Anzahl Pulse berechnen, die nötig sind, um eine bestimmte Distanz zurückzulegen.

Hochlaufen lassen (Ramping) wurde eingeführt als Möglichkeit, schrittweise zu beschleunigen und zu verlangsamen. Es schont die Servos, und wir haben empfohlen, dass Sie Ihre eigenen Ramping-Routinen anstelle der abrupten Start und Stop Routinen der Beispielprogramme einsetzen. Ramping wird erzeugt, indem man die Zählervariable einer **FOR…NEXT** Schleife zur Zahl 750 im ***Duration*** argument des **PULSOUT** Befehls addiert bzw. sie davon subtrahiert.

Es wurden Subroutinen eingeführt als eine Möglichkeit, um vorprogrammierte Bewegungen in einem PBASIC Programm mehrfach verwenden zu können. Statt eine ganze **FOR…NEXT** Schleife für jede neue Bewegung schreiben zu müssen, kann bei Bedarf mit dem **GOSUB** Befehl einfach die Subroutine ausgeführt werden, welche die **FOR…NEXT** Schleife enthält. Eine Subroutine beginnt mit einem Label, und endet mit dem **RETURN** Befehl. Die Subroutine wird mit dem **GOSUB** Befehl aus dem Hauptprogramm

aufgerufen. Wenn die Subroutine abgearbeitet ist und zum **RETURN** Befehl kommt, ist der nächste ausgeführte Befehl derjenige, der direkt auf den **GOSUB** Befehl folgt.

Die BASIC Stamp verwendet ihr EEPROM zur Speicherung des Programms, das sie ausführt, aber man kann den unbenutzten Teil verwenden, um Daten zu speichern. Das ist eine grossartige Form, um eigene Navigationsroutinen abzulegen. Die **DATA** Direktive speichert Werte im EEPROM. Im Normalfall werden Bytes gespeichert, aber wenn man den **Word** Modifier jedem Datenelement hinzufügt, kann man Werte bis zu 65535 in Doppelbyte Speicherplätzen des EEPROM unterbringen. Mit dem **READ** Befehl kann man die Werte aus dem EEPROM zurücklesen. Wenn Sie eine Wort grosse Variable abrufen, vergessen Sie nicht, einen **Word** Modifier vor die Variable zu schreiben, in welche der Wert eingelesen werden soll. **SELECT**…**CASE** wurde als eine Möglichkeit eingeführt, wie man eine Variable fallweise auswerten und für jeden einzelnen Fall einen anderen Programmcodeblock ausführen lassen kann. Optionale **DO**…**LOOP** Bedingungen sind unter gewissen Umständen hilfreich; Mit **DO**…**LOOP UNTIL (condition)** wurde gezeigt, wie man eine **DO**…**LOOP** Schleife drehen lassen kann, bis eine bestimmte Bedingung erfüllt ist.

Fragen

1. In welche Richtung muss sich das linke Rad drehen, damit sich der Boe-Bot vorwärts bewegt? In welche Richtung das rechte?
2. In welche Richtung muss sich das rechte Rad drehen, damit sich der Boe-Bot rückwärts bewegt? In welche Richtung das linke?
3. Was bestimmt die Geschwindigkeit und Drehrichtung des Servos? Welcher PBASIC Befehl und welches Argument lässt sie Geschwindigkeit und Drehrichtung des Servos mit dem Programm steuern?
4. Wenn der Boe-Bot nach links dreht, was machen die rechten und linken Räder? Welche PBASIC Befehle benötigen Sie, damit der Boe-Bot sich nach links dreht?
5. Wie können Sie es korrigieren, wenn Ihr Boe-Bot leicht nach links zieht, wenn er gemäss Programm geradeaus fahren sollte? Welcher Befehl muss angepasst werden und welche Anpassung können Sie machen?
6. Wenn Ihr Boe-Bot 11 in/s, zurücklegt, wie viel Pulse sind dann nötig, damit er 36 inches zurücklegt?
7. Wenn Ihr Boe-Bot 60 cm/s, zurücklegt, wie viel Pulse sind dann nötig, damit er 1 Meter zurücklegt?

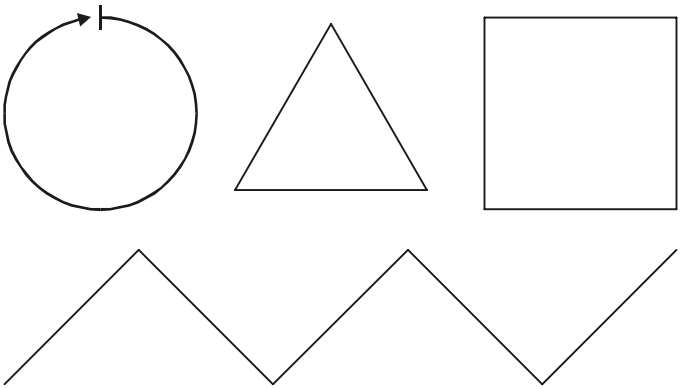
8. Was ist die Beziehung zwischen dem ***Counter*** Argument einer **FOR**…**NEXT** Schleife und dem ***Duration*** Argument des **PULSOUT** Befehls, das ein Ramping ermöglicht?
9. Was ist der Unterschied zwischen einer Subroutine und dem Aufruf einer Subroutine? Welche besonderen Eigenschaften haben sie?
10. Welchen Befehl können Sie verwenden, um Daten vorab im EEPROM der BASIC Stamp zu speichern, bevor Sie das Programm starten?
11. Mit welchem Befehl können Sie einen im EEPROM gespeicherten Wert auslesen und in eine Variable kopieren?
12. Mit welchem Codeblock können Sie eine ausgewählte Variable fallweise auswerten und für jeden Fall einen eigenen Codeblock ausführen lassen?
13. Was sind die möglichen Bedingungen, die Sie mit einem **DO**…**LOOP** verwenden können?
14. Woher weiss EepromNavigationWithWordValues.bs2 wann es zu Ende ist? Woher weiss EepromNavigation.bs2 wann es zu Ende ist?
15. Was ist der Unterschied zwischen dem **READ** Befehl in EepromNavigation.bs2 und dem in EepromNavigationWithWordValues.bs2?

Übungen

1. Schreiben Sie eine Routine, die den Boe-Bot für 350 Pulse rückwärts fahren lässt.
2. Schreiben Sie eine Routine, die den Boe-Bot für 50 Pulse nach links vorwärts schwenken lässt.
3. Nehmen wir an, Sie haben Ihre Servos getestet und festgestellt, dass sie 48 Impulse für eine 180° Drehung nach rechts benötigen. Schreiben Sie auf dieser Basis die Routinen, um den Boe-Bot 30, 45, und 60 Grad Drehungen ausführen zu lassen.
4. Mit Bezug auf Aktivität 3: Bestimmen Sie die Anzahl Impulse, die nötig sind, um den Boe-Bot 60cm zurücklegen zu lassen, unter der Annahme, dass er pro Sekunde 53 cm zurücklegt. Verwenden Sie den berechneten Wert um eine Routine zu schreiben, die Ihren Boe-Bot 60 cm vorwärts fahren lässt.
5. Schreiben Sie eine Routine, die den Boe-Bot geradeaus fahren lässt, dann langsam in eine Drehung übergeht, und daraus langsam wieder zurück in die Geradeausfahrt.
6. Es gibt vier mögliche Drehbewegungen. Schreiben Sie eine Subroutine für jede davon. Schreiben Sie eine Beispiel-Routine, die jede der vier Drehroutinen aufruft.

Projekte

1. Es ist Zeit die Spalte 3 von Table 2-1: PULSOUT Duration Kombinationen auf Seite 84 auszufüllen. Ändern Sie dazu die **`PULSOUT` *`Duration`*** Argumente im Programm BoeBotForwardThreeSeconds.bs2 ab, indem Sie jedes Wertepaar von Spalte 1 verwenden. Halten Sie das resultierende Verhalten ihres Boe-Bot in Spalte 3 Fest. Die vollständige Tabelle wird als Referenzliste dienen, wenn Sie Ihre eigenen Boe-Bot Bewegungen entwerfen.
2. Figur 4-9 zeigt vier einfache Hindernisparcours. Schreiben Sie ein Programm, das Ihren Boe-Bot jeder der vier Figuren nachfahren lässt. Nehmen Sie an, alle Abstände in gerader Linie (inklusive Kreisdurchmesser) seinen jeweils 1 m lang.



Figur 4-9
Einfache Hindernisparcours

3. Ändern Sie Ihre Programme von Projekt 2 so ab, dass der Boe-Bot in jede Bewegung langsam hineinfährt und am Ende wieder langsam abbremst (Ramping).
4. Erstellen Sie eine eigene Subroutine für jede Bewegung von Projekt 1.

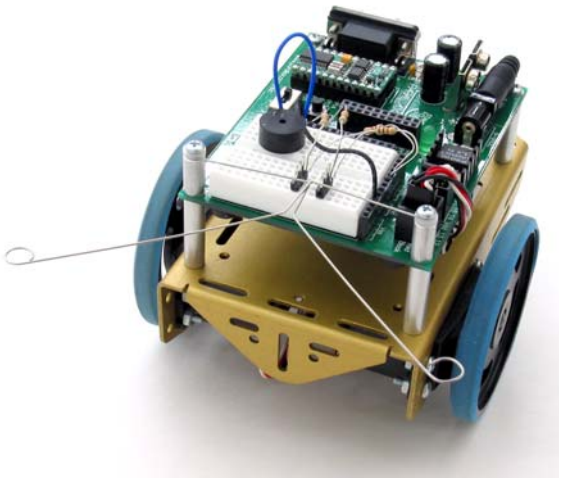
Kapitel 5: Berührungsnavigation mit Fühlern

Die verschiedensten robotischen Geräte verlassen sich auf eine Vielfalt von Berührungsschaltern. . So kann zum Beispiel ein Berührungsschalter erkennen, wenn ein Roboterarm auf ein Objekt trifft. Fabriken verwenden Berührungsschalter um Objekte einer Produktionsstrasse zu zählen und um Gegenstände während der industriellen Prozesse auszurichten. In allen diesen Fällen liefern die Schalter Inputs, die eine entsprechende Form von programmiertem Output bestimmen. Die Inputs werden elektronisch überwacht, sei es in einem Roboter, einem Taschenrechner oder einer Produktionsstrasse. Ausgehend vom Status der Schalter greift sich der Roboterarm einen Gegenstand, oder der Taschenrechner führt seine Anzeige nach, oder die Fertigungsanlage reagiert mit Motoren oder Servos um die Produktion zu steuern.

In diesem Kapitel werden Sie Berührungsschalter, genannt Fühler, an Ihren Boe-Bot anbauen und sie testen. Sie werden dann den Boe-Bot programmieren, den Status dieser Schalter zu überwachen und zu entscheiden, was zu tun ist, wenn er einem Hindernis begegnet.

BERÜHRUNGSNAVIGATION

Die Fühler (engl: *whiskers*, Schnurrbarthaare) heissen so, weil diese Stossstangen-Schalter so aussehen wie Fühler. Obwohl manche sagen, sie ähnelten eher Antennen. Wie auch immer, diese Fühler sehen Sie montiert an einem Boe-Bot in Figur 5-1. Fühler geben dem Boe-Bot die Möglichkeit, die Umwelt durch Berührung wahrzunehmen, ähnlich wie die Antennen einer Ameise oder die Schnurrbarthaare einer Katze. Die Aktivitäten in diesem Kapitel benutzen die Fühler allein, aber sie können ebenso gut mit anderen Sensoren kombiniert werden, die Sie in späteren Kapiteln noch kennen lernen werden, um die Funktionalität Ihres Boe-Bot zu erhöhen.



Figur 5-1
Boe-Bot mit Fühlern

AKTIVITÄT 1: AUFBAU UND TEST DER FÜHLER 1:

Bevor wir zu den Programmen kommen, die den Boe-Bot aufgrund seiner Berührungen navigieren, ist es wichtig, die Fühler aufzubauen und zu testen. Dieser Abschnitt führt Sie durch den Aufbau und den Test der Fühler.

Fühler - Schaltung und Zusammenbau

- √ Legen sie die Hardware für die Fühler wie in Figur 5-2 bereit.
- √ Schalten Sie die Stromversorgung von Ihrem Board und den Servos ab.

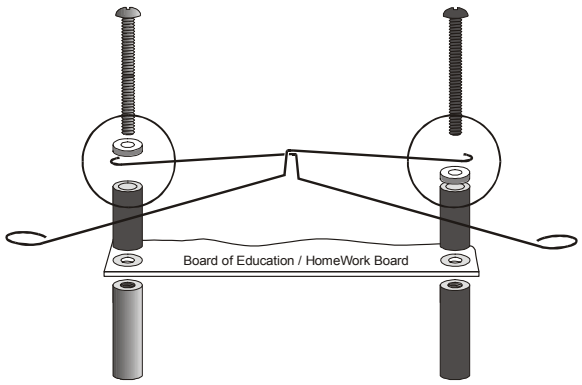
Bauteile:

(2) Fühler Drähte
(2) $^{7}/_{8}$" Flachkopf 4-40 Kreuzschrauben
(2) ½" Abstandsrohre
(2) Nylon Unterlagsscheiben Grösse 4
(2) 3-Pin Stecker/Stecker
(2) Widerstände, 220 Ω (rot-rot-braun)
(2) Widerstände, 10 kΩ (braun -schwarz-orange)

Figur 5-2 Fühler Hardware

Montage der Fühler

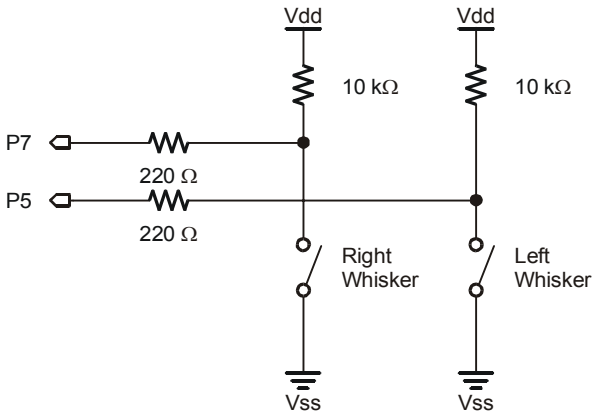
- √ Entfernen Sie die beiden vorderen Schrauben, die Ihr Board an den Abstandsrohren befestigen..
- √ Beachten Sie Figur 5-3 während Sie die folgenden Anweisungen befolgen.
- √ Schieben Sie erst eine Nylon-Unterlagsscheibe und dann einen ½" Abstandshalter auf jede der $^{7}/_{8}$" Schrauben.
- √ Stecken Sie die Schrauben durch die Löcher in Ihrem Board in die Abstandshalter darunter, aber schrauben Sie diese noch nicht völlig fest.
- √ Stülpen Sie die Haken-Enden der Tastdrähte um die Schrauben, einen unter, den anderen über die Unterlagsscheibe, so dass sich die beiden überkreuzen können, ohne sich zu berühren.
- √ Ziehen Sie nun die Schrauben in den Abstandsrohren fest.



Figur 5-3
Montage der Fühler

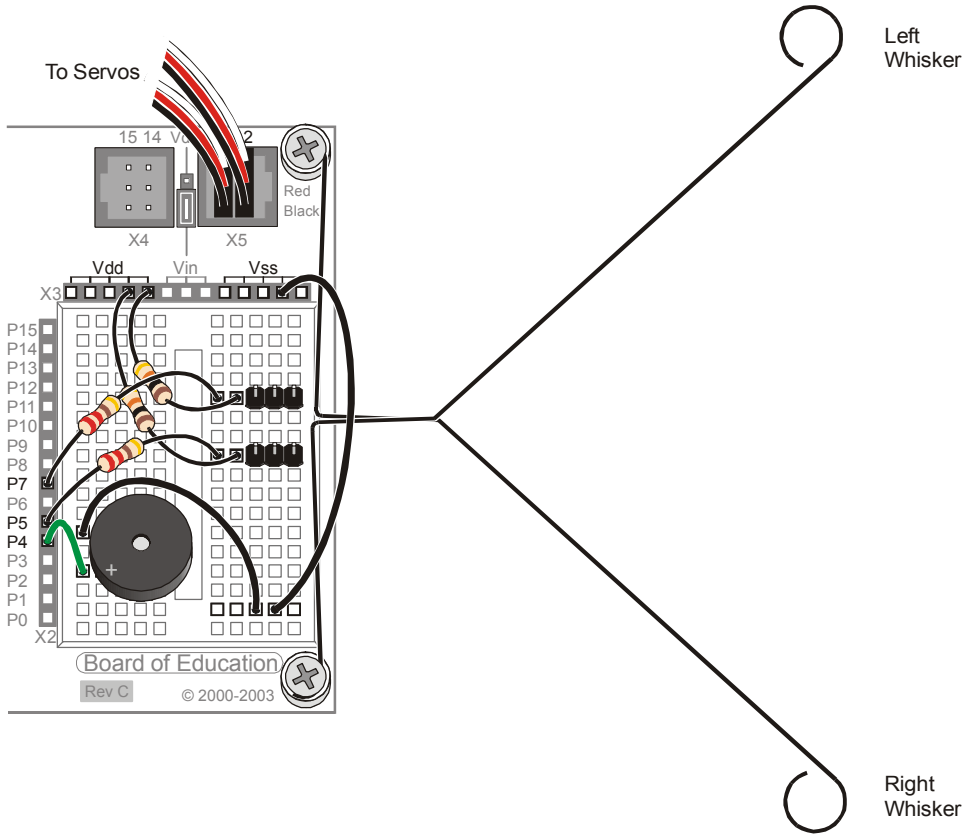
Der nächst Schritt ist, die Fühler-Schaltung von Figur 5-4 zum Piezo-Lautsprecher und den Servo-Schaltern, die Sie in Kapitel 2 und Kapitel 3 aufgebaut und getestet haben, hinzuzufügen.

- √ Wenn Sie ein Board of Education haben, bauen Sie die Fühler-Schaltung auf wie in Figur 5-4 gezeigt. Benutzen Sie dazu das Verdrahtungsdiagramm in Figur 5-5 auf Seite 169 als Referenz.
- √ Wenn Sie ein HomeWork Board haben, bauen Sie die Fühler-Schaltung auf wie in Figur 5-4 gezeigt. Benutzen Sie dazu das Verdrahtungsdiagramm in Figur 5-6 auf Seite 170 als Referenz.



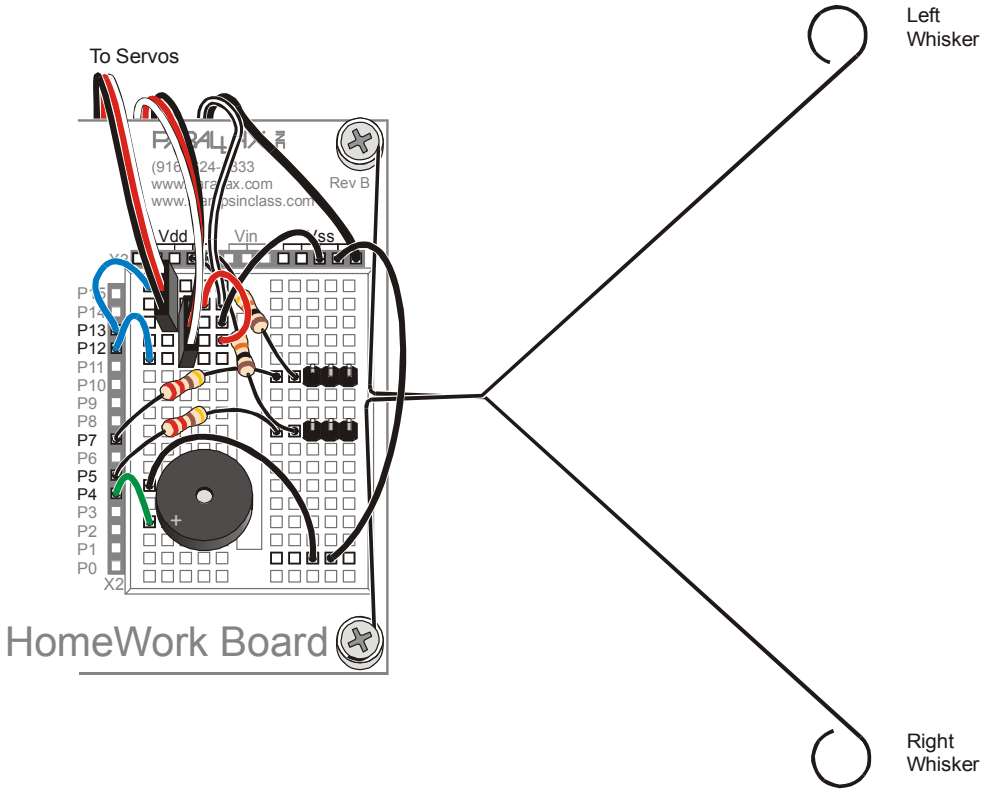
Figur 5-4
Fühler Schema

Figur 5-5: Fühler Verdrahtungsdiagramm für das Board of Education



Benutzen Sie die 220 Ω Widerstände (Farbcode rot-rot-braun) um P5 und P7 mit ihren jeweiligen 3-Pin-Steckern zu verbinden. Benutzen Sie die 10 kΩ Widerstände (braun-schwarz-orange) um jeden 3-Pin-Stecker mit Vdd zu verbinden.

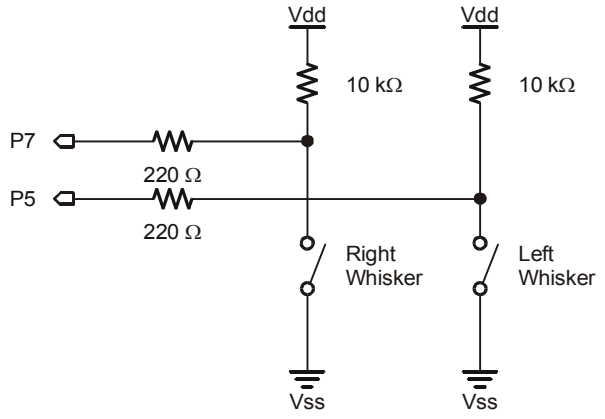
Figur 5-6: Fühler Verdrahtungsdiagramm für das HomeWork Board



Benutzen Sie die 220 Ω Widerstände (Farbcode rot-rot-braun) um P5 und P7 mit ihren jeweiligen 3-Pin-Steckern zu verbinden. Benutzen Sie die 10 kΩ Widerstände (braun-schwarz-orange) um jeden 3-Pin-Stecker mit Vdd zu verbinden.

Test der Fühler

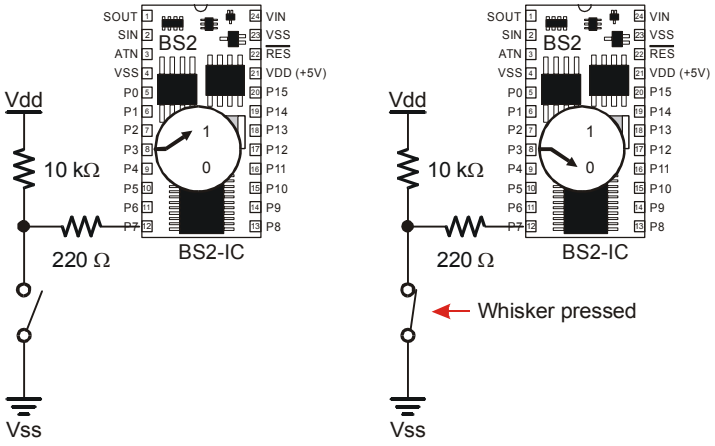
Betrachten Sie nochmals das Fühlerschema (Figur 5-7). Jeder Fühler ist sowohl der Schaltkontakt, als auch die Masseleitung eines einpoligen Tast-Einschalters. Die Tastdrähte sind mit der Masse (Vss) verbunden, weil die durchkontaktierten Löcher am Rand des Boards alle mit Vss verbunden sind. Die metallenen Abstandshalter und Schrauben sorgen für den elektrischen Kontakt mit den Tastdrähten.



Figur 5-7
Fühler Schema – Ein vertiefter Blick

Die BASIC Stamp kann programmiert werden zu erkennen, wenn ein Kontaktdraht gedrückt wird. Mit dem Fühler-Schaltkreis verbundene I/O Pins überwachen die Spannung am 10 kΩ Pull-up Widerstand. Figur 5-8 zeigt wie das funktioniert. Wenn ein gegebener Fühler nicht gedrückt ist, ist die Spannung an dessen I/O-Pin 5 V (logisch 1). Wenn der Fühler gedrückt ist, wird die I/O-Leitung mit Masse (Vss) kurzgeschlossen, so dass die I/O-Leitung 0V (logisch 0) sieht..

Wie können Sie die BASIC Stamp dazu bringen Ihnen zu sagen, ob sie eine 1 oder eine 0 liest? Da die Schaltung an P7 angeschlossen ist, erscheint dieser 1 oder 0 Wert in einer Variable mit der Bezeichnung **`IN7`**. **`IN7`** heisst ein Input Register. Input Register Variablen sind eingebaut und müssen zu Beginn des Programms nicht deklariert werden. Sie können den Wert, der in dieser Variablen gespeichert ist, sehen, indem Sie den **`DEBUG`** Befehl **`BIN1 IN7`** verwenden. **`BIN1`** ist eine Formatanweisung, die das Debug Terminal anweist, ein Binärzeichen (Bit) anzuzeigen (entweder 1 oder 0).



Figur 5-8
Feststellen von elektrischen Kontakten

Beispielprogramm: Testwhiskers.bs2

Das Beispielprogramm soll die Tastdrähte testen um zu überprüfen, dass sie richtig funktionieren. Dazu wird der Status der zu den I/O-Pins gehörenden Input Register angezeigt, an denen die Fühlerschaltungen angeschlossen sind (**`IN7`** and **`IN5`**).

Alle I/O Pins sind automatisch als Input geschaltet wenn ein PBASIC Programm startet. Das heisst, die I/O Pins an den Fühlern funktionieren automatisch als Inputs. Als Input bewirkt ein I/O Pin, dass das Input Register für diesen I/O Pin eine 1 Speichert, wenn die Spannung 5V ist (Fühler nicht gedrückt) und eine 0, wenn die Spannung 0V (Fühler gedrückt). Mit dem Debug Terminal können diese Werte angezeigt werden.

- √ Schalten Sie die Stromversorgung Ihres Boards und der Servos wieder ein.
- √ Erfassen, speichern und starten Sie TestWhiskers.bs2.
- √ Dieses Programm verwendet das Debug Terminal. Lassen Sie also das serielle Kabel während der Programmausführung an der BASIC Stamp angeschlossen.

```
' Robotics with the Boe-Bot - TestWhiskers.bs2
' Display what the I/O pins connected to the whiskers sense.

' {$STAMP BS2}                                 ' Stamp directive.
' {$PBASIC 2.5}                                ' PBASIC directive.

DEBUG "WHISKER STATES", CR,
```

```
      "Left     Right", CR,
      "------   ------"

DO
  DEBUG CRSRXY, 0, 3,
        "P5 = ", BIN1 IN5,
        "   P7 = ", BIN1 IN7
  PAUSE 50
LOOP
```


- √ Überprüfen Sie die im Debug Terminal angezeigten Werte. Es sollten sowohl P7 wie auch P5 als 1 angezeigt werden.
- √ Ziehen Sie Figur 5-5 auf Seite 169 (oder Figur 5-6 auf Seite 170) zu Rate, damit Sie wissen, welcher Fühler der "linke Fühler" und welcher der "rechte Fühler" ist.
- √ Drücken Sie den rechten Fühler an seinen 3-Pin Stecker und überprüfen Sie die im Debug Terminal angezeigten Werte. Dort sollte nun stehen:
  P5 = 1 P7 = 0
- √ Drücken Sie den linken Fühler an seinen 3-Pin Stecker und überprüfen Sie erneut die im Debug Terminal angezeigten Werte. Diesmal sollte dort stehen:
  P5 = 0 P7 = 1
- √ Drücken Sie beide Fühler gegen ihre 3-Pin Stecker. Jetzt sollte stehen
  P5 = 0 P7 = 0
- √ Falls die Fühler alle diese Tests bestanden haben, sind sie für die nächsten Schritte bereit. Andernfalls überprüfen Sie Ihr Programm und die Schaltungen auf Fehler..

Was ist `CRSRXY`?

Mit dieser Formatanweisung können Sie die Daten aus Ihrem Program auf dem Debug Terminal bequem arrangieren. Die Formatanweisung **`CRSRXY 0, 3,`** im Befehl

```
DEBUG CRSRXY, 0, 3,
        "P5 = ", BIN1 IN5,
        "   P7 = ", BIN1 IN7
```

Setzt den Cursor auf Spalte 0 Zeile 3 im Debug Terminal. Damit erhalten Sie die Anzeige sauber unter dem Tabellentitel "Whisker States". Bei jedem Schleifendurchlauf überschreiben die neuen Werte die alten, weil der Cursor immer wieder auf dieselbe Stelle zurückspringt.

AKTIVITÄT 2: FELDVERSUCH MIT DEN FÜHLERN

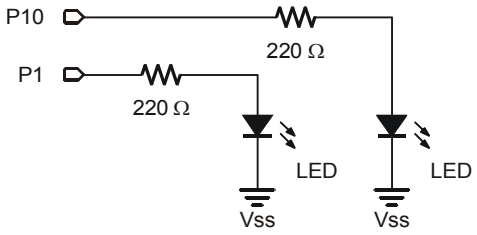
Nehmen Sie einmal an, dass Sie später einmal die Fühler ohne Ihren Computer testen sollten. Da dann das Debug Terminal nicht verfügbar sein wird, müssen Sie sich etwas anderes ausdenken. Eine Lösung wäre, die BASIC Stamp so zu programmieren, dass sie ein zum erhaltenen Input Signal passendes Output Signal produziert. Dies kann mit zwei LED-Schaltungen und einem Programm erreicht werden, das die LEDs abhängig vom Fühlerinput ein- und ausschaltet.

Bauteile Liste:

(2) Widerstände - 220 Ω (rot-rot-braun)
(2) LEDs – rot

Aufbau der LED Fühler Testschaltung

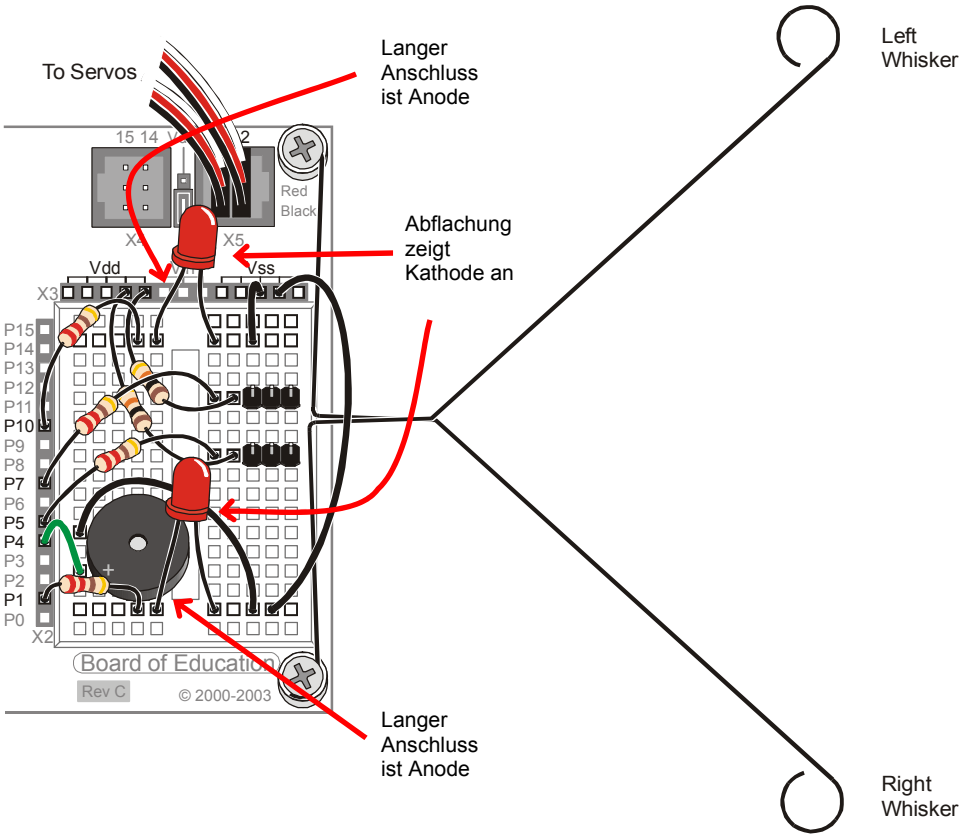
√ Schalten Sie den Strom an Ihrem Board und den Servos aus.
√ Wenn Sie ein Board of Education haben, ergänzen Sie die Schaltung von Figur 5-9 mit Hilfe des Verdrahtungsdiagramms von Figur 5-10 (Seite 175).
√ Wenn Sie ein HomeWork Board haben, ergänzen Sie die Schaltung von Figur 5-9 mit Hilfe des Verdrahtungsdiagramms von Figur 5-11 (Seite 176).



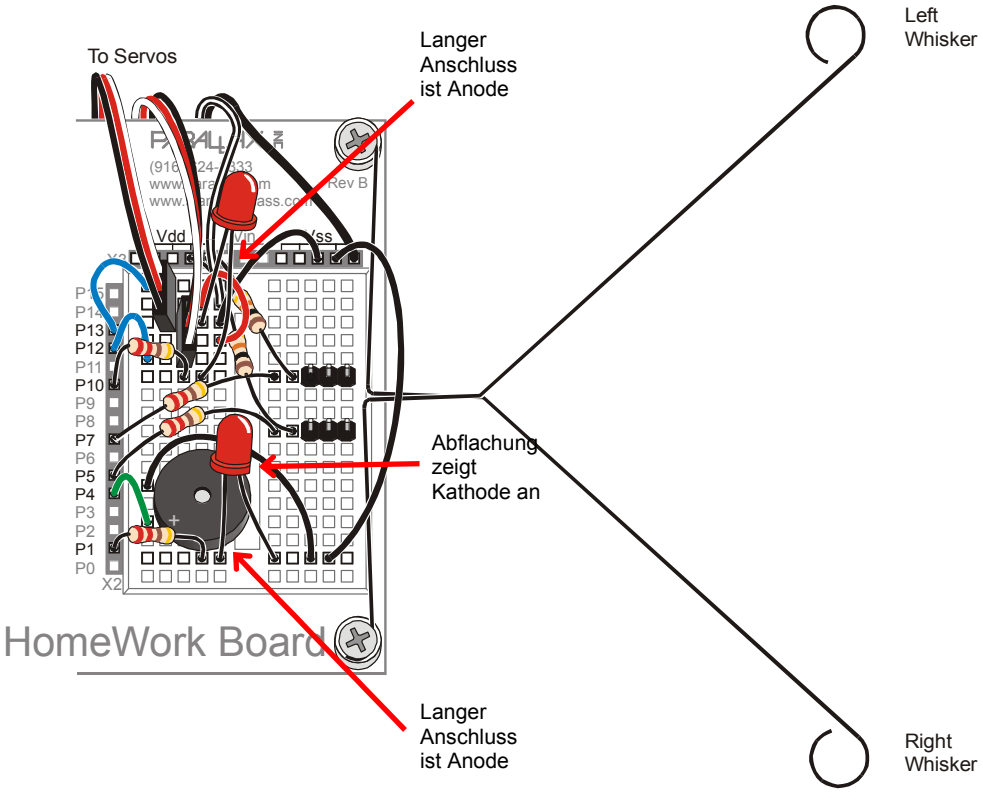
Figur 5-9
LED Fühler Test Schema

Montieren Sie diese LED Schaltung.

Figur 5-10: Fühler plus LED Verdrahtungsdiagramm für dasBoard of Education



Figur 5-11: Fühler plus LED Verdrahtungsdiagramm für das HomeWork Board



Programmierung der LED Fühlertestschaltung

√ Schalten Sie den Strom an Ihrem Board wieder ein.
√ Speichern Sie TestWhiskers.bs2 als TestWhiskersWithLeds.bs2.
√ Fügen Sie folgende zwei Befehle zwischen den zweiten **DEBUG** Befehl und den **PAUSE 50** Befehl.

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

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

Dies bezeichnet man als **IF…THEN…ELSE** (Wenn…Dann…Sonst) Befehle. Sie werden im nächsten Abschnitt detailliert vorgestellt. Diese Befehle werden in PBASIC verwendet um Entscheidungen zu treffen. Das erste der zwei **IF…THEN** Ausdrücke setzt P1 High, was die LED einschaltet, wenn der Fühler an P7 gedrückt wird. (**IN7 = 0**). Der **ELSE** Teil des Ausdrucks schaltet P1 auf Low, was die LED ausschaltet, wenn der Fühler nicht gedrückt ist. Der zweite **IF…THEN** Ausdruck macht dasselbe für den Fühler an P5 und die LED an P10.

√ Starten Sie TestWhiskersWithLeds.bs2.
√ Testen Sie das Programm, indem Sie vorsichtig die Fühler drücken. Die roten LEDs sollten aufleuchten, wenn der jeweilige Fühler Kontakt mit dem 3-Pin Stecker bekommt.

AKTIVITÄT 3: NAVIGATION MIT FÜHLERN

In Aktivität 1 wurde die BASIC Stamp darauf programmiert zu erkennen, ob ein bestimmter Fühler gedrückt wurde. In dieser Aktivität soll diese Information im Programm genutzt werden, um den Boe-Bot zu lenken. Wenn der Boe-Bot vor sich hin rollt, und ein Fühler gedrückt wird, bedeutet dies, dass der Boe-Bot mit etwas

zusammengestossen ist. Ein Navigationsprogramm muss diesen Input aufnehmen, entscheiden, was das bedeutet um dann eine Serie von Bewegungen aufzurufen, damit der Boe-Bot sich rückwärts von dem Hindernis befreit und in eine andere Richtung weiterfährt.

Programmierung der Boe-Bot-Navigation mit Fühler-Input

Das nächste Programm lässt den Boe-Bot vorwärts fahren, bis er auf ein Hindernis stösst. In diesem Fall weiss der Boe-Bot, dass er ein Hindernis getroffen hat, weil er mit einem oder beiden Fühlern daran angestossen ist. Sobald das Hindernis mit den Fühlern entdeckt wurde lassen Navigationsroutinen und –Subroutinen, die wir im Kapitel 4 entwickelt haben, den Boe-Bot zurücksetzen und drehen. Dann nimmt der Boe-Bot seine Vorwärtsfahrt wieder auf, bis er auf ein anderes Hindernis trifft.
Damit er das tun kann, muss der Boe-Bot programmiert werden, Entscheidungen zu treffen. PBASIC hat einen Befehl genannt **IF…THEN** Statement, der Entscheidungen trifft. Die Syntax für **IF…THEN** Statements ist:

IF *(condition)* THEN…{ELSEIF *(condition)*}…{ELSE}…ENDIF

Die "…" bedeuten, dass Sie einen Codeblock (ein oder mehrere Befehle) zwischen die Schlüsselwörter einfügen können. Das nächste Beispielprogramm trifft Entscheidungen auf Grundlage der Fühler Inputs und ruft dann eine Subroutine auf, damit der Boe-Bot entsprechende Massnahmen ergreift. Die Subroutinen sind ähnlich denjenigen aus Kapitel 4. Und so wird **IF…THEN** verwendet:

```
IF (IN5  = 0) AND (IN7 = 0) THEN
  GOSUB Back_Up                ' Both whiskers detect obstacle,
  GOSUB Turn_Left              ' back up & U-turn (left twice)
  GOSUB Turn_Left
ELSEIF (IN5  = 0) THEN         ' Left whisker contacts
  GOSUB Back_Up                ' Back up & turn right
  GOSUB Turn_Right
ELSEIF (IN7  = 0) THEN         ' Right whisker contacts
  GOSUB Back_Up                ' Back up & turn left
  GOSUB Turn_Left
ELSE                           ' Both whiskers 1, no contacts
  GOSUB Forward_Pulse          ' Apply a forward pulse &
ENDIF                          ' check again
```

Beispielprogramm: RoamingWithWhiskers.bs2

Dieses Programm zeigt eine Möglichkeit, die Fühler Inputs auszuwerten und mit **IF**…**THEN** zu entscheiden, welche Subroutine aufzurufen ist.

- √ Schalten Sie die Stromversorgung für Ihr Board und die Servos ein.
- √ Erfassen, speichern uns starten Sie RoamingWithWhiskers.bs2.


```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - RoamingWithWhiskers.bs2
' Boe-Bot uses whiskers to detect objects, and navigates around them.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

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

pulseCount     VAR     Byte                ' FOR...NEXT loop counter.

' -----[ Initialization ]--------------------------------------------------

FREQOUT 4, 2000, 3000                      ' Signal program start/reset.

' -----[ Main Routine ]----------------------------------------------------

DO
  IF (IN5  = 0) AND (IN7 = 0) THEN         ' Both whiskers detect obstacle
    GOSUB Back Up                          ' Back up & U-turn (left twice)
    GOSUB Turn Left
    GOSUB Turn_Left
  ELSEIF (IN5  = 0) THEN                   ' Left whisker contacts
    GOSUB Back Up                          ' Back up & turn right
    GOSUB Turn Right
  ELSEIF (IN7  = 0) THEN                   ' Right whisker contacts
    GOSUB Back Up                          ' Back up & turn left
    GOSUB Turn_Left
  ELSE                                     ' Both whiskers 1, no contacts
    GOSUB Forward Pulse                    ' Apply a forward pulse
  ENDIF                                    ' and check again
LOOP

' -----[ Subroutines ]-----------------------------------------------------

Forward Pulse:                             ' Send a single forward pulse.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
```

```
  RETURN

Turn Left:                                      ' Left turn, about 90-degrees.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn Right:
  FOR pulseCount = 0 TO 20                      ' Right turn, about 90-degrees.
    PULSOUT 13, 850
    PULSOUT 12, 850

    PAUSE 20
  NEXT
  RETURN

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

Wie es funktioniert

Die **IF...THEN** Ausdrücke im Main Routine Abschnitt prüfen zuerst die Fühler auf irgend einen Status, um den man sich kümmern müsste. Wenn beide Fühler gedrückt sind (**IN5 = 0** und **IN7 = 0**), wird eine Wende ausgeführt, indem die **Back_Up** Subroutine aufgerufen wird gefolgt von der **Turn_Left** Subroutine zweimal hintereinander. Wenn nur der linke Fühler gedrückt ist, (**IN5 = 0**), ruft das Programm die **Back_Up** Subroutine gefolgt von der **Turn_Right** Subroutine auf. Wenn der rechte Fühler gedrückt ist **(IN7 = 0**), wird die **Back_Up** Subroutine aufgerufen, gefolgt von der **Turn_Left** Subroutine. Die einzige damit noch nicht abgedeckte mögliche Kombination bleibt, wenn keiner der Fühler gedrückt ist. (**IN5 = 1** and **IN7 = 1**). Der **ELSE** Befehl ruft in diesem Falle die **Forward_Pulse** Subroutine auf.

```
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
```


Die **Turn_Left**, **Turn_Right**, und **Back_Up** Subroutinen sollten ziemlich vertraut erscheinen, aber die **Forward_Pulse** Subroutine hat eine Besonderheit. Sie sendet nur gerade einen einzigen Puls, und springt dann zurück (Return). Das ist wirklich wichtig, weil das dem Boe-Bot erlaubt, die Fühler zwischen jedem Vorwärts-Puls abzufragen.

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

Da jeder Vollgas-Puls den Boe-Bot etwa einen halben Zentimeter vorwärts rollen lässt, erscheint es wirklich sehr vernünftig, nur einen einzigen Puls zu senden, und dann wieder die Fühler zu kontrollieren. Da das **IF…THEN** Statement innerhalb einer **DO…LOOP**, Schleife steckt, kommt das Programm jedes mal wenn es von einem **Forward_Pulse** zurückkommt direkt zu **LOOP**, was das Programm wieder nach oben zu **DO** verzweigt. Was passiert dann? Die **IF…THEN** Statements prüfen die Taster immer wieder.

Sie sind dran

Die ***EndValue*** Argumente der **FOR...NEXT** Schleife in den **Back_Right** und **Back_Left** Routinen können für mehr oder weniger Drehung angepasst werden, und an der **Back_Up** Routine kann der ***EndValue*** so angepasst werden, dass er weniger weit zurücksetzt um die Navigation in engen Räumen zu erleichtern.

√ Experimentieren Sie mit den **FOR...NEXT** Schleife ***EndValue*** Argumenten in den Navigationsroutinen in RoamingWithWhiskers.bs2.

Sie können auch Ihre **IF…THEN** Statements anpassen, damit die LEDs von vorhin anzeigen, welches Manöver der Boe-Bot ausführt, indem Sie **HIGH** und **LOW** Befehle zur Steuerung der LED-Schaltungen hinzufügen. Hier ist ein Beispiel:

```
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
```

√ Ändern Sie die **IF…THEN** Statement in RoamingWithWhiskers.bs2 so, dass der Boe-Bot seine Bewegungen mit den LEDs anzeigt.

AKTIVITÄT 4: KÜNSTLICHE INTELLIGENZ UND ENTSCHEIDEN, WANN MAN FESTSITZT

Vielleicht haben Sie schon bemerkt, dass der Boe-Bot in Ecken stecken bleibt. Wenn der Boe-Bot in eine Ecke kommt, berühren seine Fühler die Wand links, also dreht er rechts. Wenn er wieder nach vorne fährt, kollidiert der rechte Fühler mit der Wand, also dreht er links. Dann dreht er und fährt wieder in die linke Wand, und dann wieder in die rechte Wand, und so weiter, bis ihn jemand aus seiner Zwickmühle befreit.

Programm um aus Ecken zu kommen

RoamingWithWhiskers.bs2 kann angepasst werden, damit es dieses Problem erkennt und etwas dagegen unternehmen kann. Der Trick ist zu zählen, wie oft die Fühler abwechselnd Kontakt haben. Bei diesem Trick ist es wichtig, dass das Programm sich erinnern muss, welchen Status die Fühler beim letzten Kontakt hatten. Diese müssen mit dem Status der Fühler beim aktuellen Kontakt verglichen werden. Wenn diese übers Kreuz sind, muss der Zähler eins hochgezählt werden. Wenn der Zähler eine von Ihnen (dem Programmierer) bestimmte Limite überschreitet, ist es an der Zeit, eine 180-Grad Wende zu machen und den Zähler wieder auf Null zu setzen.

Das folgende Programm basiert auf der Tatsache, dass Sie **IF…THEN** Statements "verschachteln" können. Mit anderen Worten, das Programm prüft eine Bedingung, und wenn diese erfüllt ist, prüft es eine weitere Bedingung innerhalb der ersten Bedingung. Hier ist ein Pseudo-Code Beispiel, wie man das verwendet.

```
IF condition1 THEN
  Commands for condition1
  IF condition2 THEN
    Commands for both condition2 and condition 1
  ELSE
    Commands for condition1 but not condition 2
  ENDIF
  ELSE
    Commands for not condition1
ENDIF
```


Es hat ein Beispiel eines verschachtelten **IF…THEN** Statements in der Routine im nächsten Programm, welche die fortlaufenden abwechselnden Fühlerkontakte erkennt.

Beispielprogramm: EscapingCorners.bs2

√ Erfassen, speichern und starten Sie EscapingCorners.bs2. Es wird Ihren Boe-Bot entweder beim vierten oder fünften abwechselnden Kontakt eine Wende ausführen lassen, abhängig davon, welcher Fühler zuerst Kontakt hatte.

```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - EscapingCorners.bs2
' Boe-Bot navigates out of corners by detecting alternating whisker presses.

' {$STAMP BS2}                             ' Stamp directive.
' {$PBASIC 2.5}                            ' PBASIC directive.

DEBUG "Program Running!"

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

pulseCount     VAR     Byte                ' For...next loop counter.
counter        VAR     Nib                 ' Counts alternate contacts.
old7           VAR     Bit                 ' Stores previous IN7.
old5           VAR     Bit                 ' Stores previous IN5.

' -----[ Initialization ]--------------------------------------------------

FREQOUT 4, 2000, 3000                      ' Signal program start/reset.
```

```
counter = 1                                 ' Start alternate corner count.
old7 = 0                                    ' Make up old values.
old5 = 1

' -----[ Main Routine ]------------------------------------------------------

DO

' ---  Detect Consecutive Alternate Corners ------------------------
' See the "How EscapingCorners.bs2 Works" section that follows this program.

  IF (IN7 <> IN5) THEN                      ' One or other is pressed.
    IF (old7 <> IN7) AND (old5 <> IN5) THEN ' Different from previous.
      counter = counter + 1                 ' Alternate whisker count + 1.
      old7 = IN7                            ' Record this whisker press
      old5 = IN5                            ' for next comparison.
      IF (counter > 4) THEN                 ' If alternate whisker count = 4,
        counter = 1                         ' reset whisker counter
        GOSUB Back_Up                       ' and execute a U-turn.
        GOSUB Turn Left
        GOSUB Turn Left
      ENDIF                                 ' ENDIF counter > 4.
    ELSE                                    ' ELSE (old7=IN7) or (old5=IN5),
      counter = 1                           ' not alternate, reset counter.
    ENDIF                                   ' ENDIF (old7<>IN7) and
                                            ' (old5<>IN5).
  ENDIF                                     ' ENDIF (IN7<>IN5).

' ---  Same navigation routine from RoamingWithWhiskers.bs2 ------------------

  IF (IN5  = 0) AND (IN7 = 0) THEN          ' Both whiskers detect obstacle
    GOSUB Back Up                           ' Back up & U-turn (left twice)
    GOSUB Turn Left
    GOSUB Turn Left
  ELSEIF (IN5  = 0) THEN                    ' Left whisker contacts
    GOSUB Back_Up                           ' Back up & turn right
    GOSUB Turn_Right
  ELSEIF (IN7  = 0) THEN                    ' Right whisker contacts
    GOSUB Back Up                           ' Back up & turn left
    GOSUB Turn Left
  ELSE                                      ' Both whiskers 1, no contacts
    GOSUB Forward_Pulse                     ' Apply a forward pulse
  ENDIF                                     ' and check again

LOOP

' -----[ Subroutines ]-------------------------------------------------------

Forward_Pulse:                              ' Send a single forward pulse.
  PULSOUT 13,850
  PULSOUT 12,650
```

```
  PAUSE 20
  RETURN

Turn Left:                                      ' Left turn, about 90-degrees.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn_Right:
  FOR pulseCount = 0 TO 20                      ' Right turn, about 90-degrees.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN

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


Wie das Programm funktioniert

Da dieses Programm eine abgeänderte Version von RoamingWithWhiskers.bs2 ist, werden hier nur die neuen Teile diskutiert, die mit dem Erkennen und Entkommen aus Ecken zu tun haben.

Drei zusätzliche Variablen werden für das Erkennen von Ecken definiert. Die nibble Variable **`counter`** kann einen Wert zwischen 0 and 15 aufnehmen. Da der Zielwert zur Erkennung einer Ecke 4 ist, scheint die Grösse dieser Variable vernünftig. Erinnern Sie sich, dass eine Bit Variable ein einzelnes Bit (0 oder 1) speichern kann. Die nächsten Beiden Variablen (**`old7`** and **`old5`**) sind beides Bit Variablen. Diese haben ebenfalls die richtige Grösse für Ihre Aufgaben, da diese die alten Werte von **`IN7`** und **`IN5`**, aufnehmen, welche ebenso Bit Variablen sind.

```
counter         VAR     Nib
old7            VAR     Bit
old5            VAR     Bit
```

Diese Variablen müssen initialisiert (mit einem Anfangswert versehen) werden. Um das Programm einfacher lesbar zu machen, wird **`counter`** auf 1 gesetzt und – wenn sie 4 erreicht weil der Boe-Bot in eine Ecke geraten ist - wieder auf 1 zurückgesetzt. Die **`old7`** und **`old5`** Variablen müssen so gesetzt werden, dass es aussieht, als sei einer der beiden Fühler irgendwann vor dem Programmstart gedrückt worden. Dies ist notwendig, weil die Routine zur Erkennung abwechselnder Berührungen nur abwechselnde Stellungen vergleicht, entweder (**`IN5 = 1`** und **`IN7 = 0`**) oder (**`IN5 = 0`** und **`IN7 = 1`**). Entsprechen müssen **`old5`** und **`old7`** von Anfang an verschieden sein.

```
counter = 1
old7 = 0
old5 = 1
```

Nun kommen wir zum Abschnitt Detect Consecutive Alternate Corners (erkenne aufeinanderfolgende abwechselnde Ecken). Das erste was wir prüfen wollen, ist, ob der eine oder andere Fühler gedrückt wurde. Eine einfache Lösung dafür ist zu fragen: "Ist **`IN7`** verschieden von **`IN5`**?" In PBASIC können wir dafür den Ungleich-Operator **`<>`** in einem **`IF`** Statement verwenden, das so aussieht "**`IF IN7 <> IN5`**".

```
IF (IN7 <> IN5) THEN
```

Wenn tatsächlich genau ein Fühler gedrückt ist, muss im nächsten Schritt geprüft werden, ob oder ob es nicht das genau umgekehrte Muster als beim letzten Mal ist. Mit anderen Worten: "Ist **`(old7 <> IN7)`** und ist **`(old5 <> IN5)`**?" Wenn das (beides) wahr ist, dann wird es Zeit den Zähler, der die fortlaufenden abwechselnden Fühlerkontakte zählt um eins zu erhöhen. Gleichzeitig müssen wir das aktuelle Fühlermuster festhalten, indem wir **`old7`** auf das aktuelle **`IN7`** und **`old5`** auf das aktuelle **`IN5`** setzen.

```
IF (old7 <> IN7) AND (old5 <> IN5) THEN
  counter = counter + 1
  old7 = IN7
  old5 = IN5
```

Wenn es sich dabei herausstellt, dass dies nun der vierte abwechselnde Fühlerkontakt ist, dann wird es Zeit den Zähler auf 1 zurückzusetzen und eine 180-Grad Wende zu machen.

```
IF (counter > 4) THEN
  counter = 1
  GOSUB Back_Up
  GOSUB Turn_Left
```

```
        GOSUB Turn_Left
```

Dieses **ENDIF** beendet den Codeblock, der bei **counter** > 4 ausgeführt wird.

```
      ENDIF
```

Das **ELSE** Statement schliesst an das **IF (old7 <> IN7) AND (old5 <> IN5) THEN** Statement an. Das **ELSE** Statement sorgt für die Fälle, wenn das **IF** Statement nicht wahr (true) ist. Mit anderen Worten, es ist kein abwechselnder Taster gedrückt, also müssen wir den Zähler zurücksetzen, da der Boe-Bot nicht in einer Ecke eingeklemmt ist.


```
    ELSE
      counter = 1
```

Dieses **ENDIF** Statement beendet den Entscheidungsprozess für das **IF** (**old7 <> IN7**) **AND (old5 <> IN5) THEN** Statement.

```
    ENDIF
  ENDIF
```

Der Rest des Programms ist gleich wie oben.

Sie sind dran

Eines der **IF...THEN** Statements in EscapingCorners.bs2 prüft, ob **counter** 4 erreicht hat.

- √ Setzen Sie den Wert versuchsweise auf 5 bzw. 6 hoch und schauen Sie, was passiert.
- √ Versuchen Sie mal, den Wert zu reduzieren und schauen Sie, welche Auswirkungen das auf das normale Fahrverhalten hat.

ZUSAMMENFASSUNG

In diesem Kapitel wurde der Boe-Bot programmiert, dass er sich abhängig vom Sensor-Input bewegte, statt nach einer vorprogrammierten Bewegungsliste. Wir verwendeten Fühler als Sensoren, die als Schliesskontakte dienten. Wenn richtig verdrahtet liefern diese Schalter an ihren Anschlüssen eine Spannung (5 V) wenn der Schalter offen ist, und eine andere Spannung (0 V) wenn der Schalter geschlossen ist. Die Input-Register der BASIC Stamp speichern eine "1", wenn sie Vdd (5 V) erkennen, und eine "0", wenn Sie auf Vss (0 V) liegen.

Wir haben die BASIC Stamp darauf programmiert die Fühlersensoren zu testen und die Ergebnisse in zwei verschiedenen Medien, dem Debug Terminal und den LEDs, anzuzeigen. PBASIC Programme wurden entwickelt, damit die BASIC Stamp die Fühler zwischen jeweils zwei Servo-Pulsen überprüft. Basierend auf dem Status der Fühler riefen **`IF`**…**`THEN`** Befehle in der Hauptroutine des Programms Subroutinen auf, ähnlich denen, die wir im vorhergegangenen Kapitel entwickelt hatten, um den Boe-Bot von Hindernissen fernzuhalten. Als ein Beispiel von künstlicher Intelligenz wurde eine weitere Subroutine entwickelt, die es dem Boe-Bot erlaubt zu erkennen, wenn er in einer Ecke eingeklemmt war. Diese Routine enthielt Schritte wie das Speichern der alten Fühlerzustände, dem Vergleich mit den aktuellen Fühlerzuständen, und dem Mitzählen der Anzahl abwechselnder Objekterkennungen.

Dieses Kapitel führte in die Input-basierte Boe-Bot Navigation ein. Die nächsten drei Kapitel werden sich auf verschiedene Sensortypen konzentrieren, mit denen man dem Boe-Bot Sehvermögen verleihen kann. Sowohl Sehvermögen wie auch Berührung eröffnen viele Möglichkeiten, um den Boe-Bot in zunehmend komplexen Umgebungen zu navigieren.

Fragen

1. Zu welcher Sorte von elektrischen Kontakten gehören die Fühler?
2. Was bewirkt die **`DEBUG`** Formatanweisung **`CRSRXY`**? Welches sind die zwei Werte, die man dieser Formatanweisung mitgeben muss?
3. Welche Spannung kann man an einem I/O-Pin messen, wenn der daran angeschlossene Fühler gedrückt wird? Welchen Binärwert erscheint im Input Register? Wenn I/O-Pin P8 benutzt wird um einen Input-Pin zu überwachen,

welchen Wert hat dann **IN8** wenn der Fühler gedrückt wird, und welchen, wenn er nicht gedrückt wird?

4. An welchen Fühler ist **IN5** angeschlossen? Was ist mit **IN7**? Wenn **IN7 = 1**, was bedeutet das? Was bedeutet **IN7 = 0**?
   Und was ist mit **IN5 = 1** und **IN5 = 0**?
5. Welcher Befehl wurde verwendet um – abhängig vom Wert einer Variablen – in verschiedene Subroutinen zu springen? Welcher Befehl wurde verwendet um zu entscheiden, in welche Subroutine verzweigt werden muss? Worauf basieren diese Entscheidungen?
6. Welche drei PBASIC Programmiertechniken wurden in diesem Kapitel benutzt um Ereignisse festzustellen und darauf basierende Entscheidungen zu treffen?
7. Was ist der Zweck von geschachtelten **IF**…**THEN** Ausdrücken?

Übungen

1. Schreiben Sie einen **DEBUG** Befehl für TestWhiskers.bs2, der jeden Fühlerstatus auf einer neuen Zeile anzeigt. Verlängern Sie den **PAUSE** Befehl von 50 auf 250.
2. Was ist die neue Abtastrate aus Übung 1? Hinweis: Die Abtastrate ist die Anzahl von Statusabtastungen pro Sekunde an den Fühlern. Die Antwort kann in Abtastungen pro Sekunde angegeben werden. Man berechnet sie, indem man den Reziprokwert der Dauer zwischen zwei Fühlerabfragen nimmt. Mit anderen Worten, die Abfragerate = 1/Zeit pro Abtastung.
3. Bestimmen Sie in RoamingWithWhiskers.bs2 die Abtastrate für die Vorwärtsfahrt. Bestimmen Sie auch die Abtastraten bei den Manövern. Hinweis: Die Frage nach den Manövern kann als Trick-Frage betrachtet werden.
4. Benutzen Sie RoamingWithWhiskers.bs2 als Referenz und schreiben Sie eine **Turn_Away** Subroutine, welche die **Back_Up** Subroutine einmal und die **Turn_Left** Subroutine zweimal aufruft. Schreiben Sie die Änderungen auf, die Sie in der Hauptroutine von RoamingWithWhiskers.bs2 vornehmen müssen.

Projekte

1. Ändern Sie RoamingWithWhiskers.bs2 so, dass der Boe-Bot für 100 ms einen 4 kHz Ton erzeugt, bevor er ein Ausweichmanöver durchführt. Lassen Sie ihn zweimal Piepsen, wenn beide Fühler während der gleichen Abtastung Kontakt anzeigen.
2. Ändern Sie RoamingWithWhiskers.bs2 so, dass die LEDs blinken, wenn der Boe-Bot ein Manöver durchführt.

3. Generelles Ziel ist es, den Boe-Bot mehr Raum in weniger Zeit abfahren zu lassen. Ändern Sie RoamingWithWhiskers.bs2 so, dass der Boe-Bot eine 45° Drehung ausführt, wenn er auf ein Hindernis aufläuft. Testen Sie seine Leistungsfähigkeit in einem Zimmer mit Wänden und einigen grossen Hindernissen. Wiederholen Sie das Experiment mit 30°, 60°, 90°, and 120° Drehungen. Experimentieren Sie auch damit, wie weit der Boe-Bot zurücksetzt.
4. *Projekt für Fortgeschrittene* – Ändern Sie RoamingWithWhiskers.bs2 so, dass sich der Boe-Bot in einem Kreis von 1 Meter Durchmesser bewegt. Wenn Sie einen Fühler berühren, soll der Boe-Bot in einem engeren Kreis (kleinerer Durchmesser) rollen. Wenn Sie den anderen Fühler berühren, soll der Boe-Bot einen grösseren Kreis fahren.

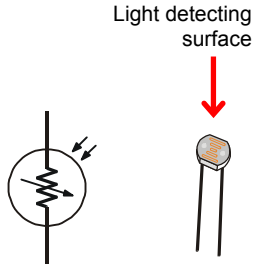
Kapitel 6: Lichtabhängige Navigation mit Fotowiderständen

Licht kennt viele Anwendungen in industriellen und Robotersteuerungen. Beispiele umfassen das Erkennen der Kanten einer Stoffrolle in der Textilindustrie, die Entscheidung, wann die Strassenbeleuchtung in den verschiedenen Jahreszeiten einzuschalten ist, wann eine Foto gemacht werden soll oder wann ein Pflanzenbeet Wasser erhalten soll.

Es gibt viele verschiedene Lichtsensoren die spezielle Funktionen erfüllen. Die Lichtsensoren in Ihrem Boe-Bot Bausatz sind für sichtbares Licht gemacht und sie können benutzt werden, damit Ihr Boe-Bot Veränderungen in der Helligkeit erkennen kann. Dank dieser Fähigkeit kann Ihr Boe-Bot programmiert werden, Orte mit hellen oder dunklen Bereichen zu erkennen, die Helligkeit oder Dunkelheit insgesamt zu melden und Lichtquellen anzusteuern, wie etwa den Strahl einer Taschenlampe oder den Lichtspalt, den eine Tür in ein dunkles Zimmer fallen lässt.

DER FOTOWIDERSTAND

Die Widerstände, die Sie in früheren Kapiteln verwendet haben, hatten feste Werte, wie etwa 220 Ω und 10 kΩ. Der Fotowiderstand (photoresistor) andererseits ist ein lichtabhängiger widerstand (“light dependent Resistor”, LDR) . Das bedeutet, dass sein Widerstandswert von der Helligkeit oder Lichtintensität des Lichts abhängt, das auf seine lichtempfindliche Seite fällt. Figur 6-1 zeigt das Schemasymbol und die Bauteilzeichnung des Fotowiderstandes, den Sie benutzen werden, damit der Boe-Bot Änderungen in der Lichtstärke erkennen kann.



Figur 6-1
Fotowiderstand
Schema und
Bauteilzeichnung

Ein Fotowiderstand ist ein lichtabhängiger Widerstand (LDR) der eine ähnliche spektrale Empfindlichkeit hat wie das menschliche Auge. Mit anderen Worten, die Sorte Licht, die Ihre Augen erkennen ist dieselbe Sorte Licht, die auch den Widerstandswert des Fotowiderstandes beeinflusst. Das aktive Element dieser Fotowiderstände bestehen aus Kadmium Sulfid. (CdS). Licht fällt in eine Halbleiterschicht die auf einem Keramikträger aufgebracht wird und erzeugt dort freie Ladungsträger. Ein definierter elektrischer Widerstand wird erzeugt, der umgekehrt proportional zur Lichtstärke ist. In anderen Worten, Dunkelheit erzeugt grösseren Widerstand, Helligkeit kleineren.

Lichtstärke ist eine wissenschaftliche Bezeichnung für die Messung von einstrahlendem Licht. Ein Weg um die Lichteinstrahlung zu verstehen, ist an eine Taschenlampe zu denken, die eine Wand beleuchtet. Der gebündelte Strahl, den Sie auf die Wand scheinen sehen, ist einfallendes Licht. Die Messgrösse für die Lichtstärke ist die "foot-candle" im englischen System oder das "Lux" im metrischen System. Wenn wir die Fotowiderstände benutzen, spielen für uns absoluten Lux-Stufen keine Rolle, nur ob die Helligkeit in gewissen Richtungen höher oder tiefer ist. Man kann den Boe-Bot dafür programmieren, die relative Lichtintensität als Information für seine Navigationsentscheide zu benutzen.

AKTIVITÄT 1: AUFBAU UND TEST DER FOTOWIDERSTANDSSCHALTUNG 1:

In diesem Abschnitt bauen und testen Sie die Lichtstärke-Sensorschaltung mit Fotowiderständen. Ihre Lichtstärke-Sensorschaltungen werden in der Lage sein, den Unterschied zwischen Schatten und kein Schatten zu erkennen. Die PBASIC-Befehle zur Unterscheidung ob ein Schatten auf den Fotowiderstand fällt ist ziemlich ähnlich zu denjenigen, die entscheiden ob oder ob nicht ein Taster ein Objekt berührt.

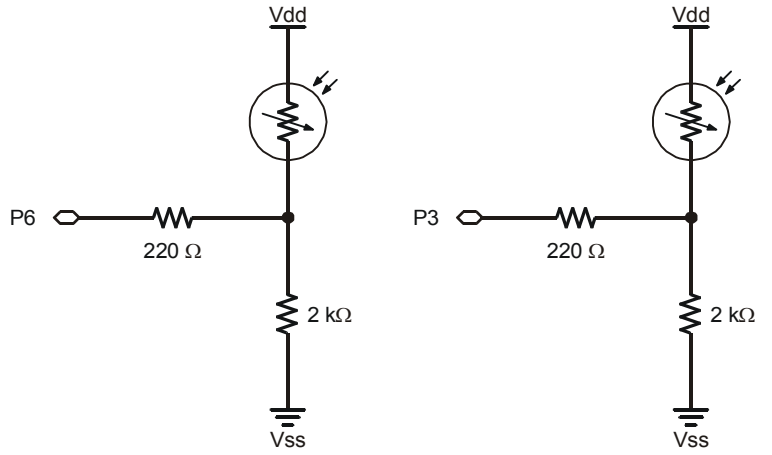
Bauteile:

(2) Fotowiderstände - CdS
(2) Widerstände – 2 kΩ (rot-schwarz- rot)
(2) Widerstände – 220 Ω (rot - rot -braun)
(4) Verbindungsleitungen
(2) Widerstände – 470 Ω (gelb-violett- braun)
(2) Widerstände – 1 kΩ (braun - schwarz - rot)
(2) Widerstände – 4.7 kΩ (gelb -violett- rot)
(2) Widerstände – 10 kΩ (braun - schwarz -orange)

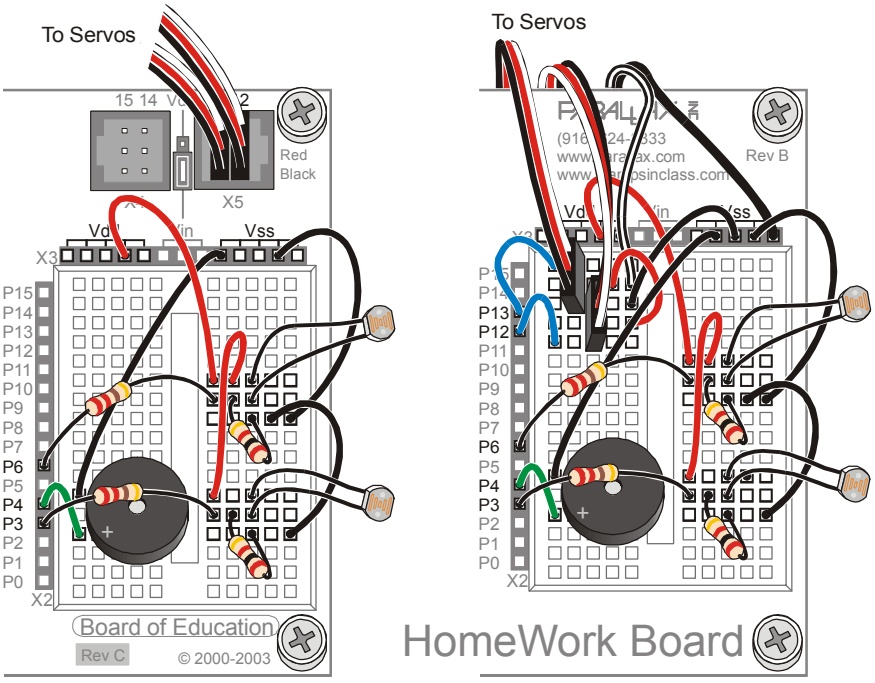
Aufbau der lichtempfindlichen Augen

Figur 6-2 zeigt das Schema und Figur 6-3 das Verdrahtungsdiagramm für die Fotowiderstandsschaltungen, die Sie in dieser und den folgenden beiden Aktivitäten verwenden.

√ Schalten Sie den Strom an Ihrem Board und den Servos aus.
√ Bauen Sie die Schaltung in Figur 6-2 auf, mit Figur 6-3 als Anleitung.



Figur 6-2
Schema – Erste lichtempfindliche Schaltung



Figur 6-3
Verdrahtungs-Diagramme für die erste lichtempfindliche Schaltung

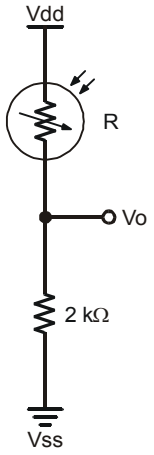
Board of Education (links) und HomeWork Board (rechts).

Wie der Fotowiderstands-Schaltkreis funktioniert

Ein BASIC Stamp I/O Pin funktioniert als Input oder Output. Als Output kann der I/O-Pin ein High (5V) oder eine Low (0V) Signal geben. Bisher haben wir die High und Low Signale verwendet um LED-Schaltungen ein und auszuschalten, Servos zu steuern und Töne zu einem Lautsprecher zu leiten. Ein BASIC-Stamp I/O-Pin kann auch als Input funktionieren. Als Input legt der I/O-Pin keine Spannung an die Schaltung, an der er angeschlossen ist. Statt dessen hört er nur still zu, ohne sichtbare Auswirkung auf die Schaltung. Wenn der I/O-Pin eine Spannung von über 1.4 V detektiert, speichert er eine 1 in seinem Input Register. Wenn er eine Spannung von weniger als 1.4 V detektiert, speichert er eine 0 in seinem Input Register. Im vorangehenden Kapitel speicherten diese Input Register Werte, die anzeigten, ob die Fühler gedrückt waren oder nicht. Zum Beispiel speicherte das `IN7` Input Register eine 1 wenn es 5V erkannte (Fühler nicht gedrückt), oder eine 0 wenn es 0V erkannte (Fühler gedrückt).

BASIC Stamp I/O Pins sind zunächst Inputs. Wenn ein BASIC Stamp Programm startet, sind alle I/O Pins als Inputs geschaltet. Wenn Sie Befehle wie **`HIGH`**, **`LOW`**, **`PULSOUT`** oder **`FREQOUT`** verwenden, wird der I/O Pin von Input auf Output umgeschaltet, damit die BASIC Stamp High oder Low Signale senden kann.

Wenn ein BASIC Stamp I/O Pin ein Input ist, verhält sich die Schaltung so, wie wenn weder der I/O Pin noch der 220 Ω Widerstand da wären. Figur 6-4 zeigt die entsprechende Schaltung. Der Widerstand des Fotowiderstands ist mit dem Buchstaben R bezeichnet. Das können wenige Ω sein, wenn das Licht sehr hell ist, oder es können etwas um die 50 kΩ in vollständiger Dunkelheit sein. In einem gut beleuchteten Raum mit Fluoreszenzröhren an der Decke kann der Widerstand zwischen 1 kΩ (voll im Licht) und 25 kΩ (Schattenwurf auf dem Objekt) liegen. So wie der Widerstandswert des Fotowiderstands mit der Lichtstärke ändert, so ändert die Spannung an Vo; wenn R grösser wird, wird Vo kleiner, und wenn R kleiner wird, wird Vo grösser. Vo ist das, was der Stamp I/O Pin erkennt, wenn er als Input geschaltet ist. Wenn diese Schaltung an **`IN6`** angeschlossen wird und die Spannung an Vo grösser ist als 1.4 V, wird **`IN6`** eine 1 speichern. Wenn die Spannung unter 1.4 V fällt, speichert **`IN6`** eine 0.



Figur 6-4
Schema – Spannungsteiler Schaltung

Wenn Widerstände wie in Figur 6-4 in einer Reihe hintereinander angeschlossen sind, sind sie **in Serie geschaltet**, und werden auch als Serienwiderstände (series resistors) bezeichnet..

Wenn zwei Widerstände in Serie geschaltet sind, um eine Spannung an Vo zu erzeugen, bezeichnet man diese Schaltung als einen **Spannungsteiler (Voltage divider)**. In so einer Schaltung kann der Wert von Vo irgendwo zwischen Vdd und Vss liegen. Der exakte Wert von Vo wird durch das Verhältnis von R zu 2 kΩ bestimmt. Wenn R grösser ist als 2 kΩ, liegt Vo näher bei Vss. Wenn R kleiner ist als 2 kΩ, liegt Vo näher an Vss. Wenn R gleich 2 kΩ ist, liegt Vo genau in der Mitte zwischen Vss und Vdd, also bei 2.5 V. Wenn man einen der beiden Werte (R oder Vo) misst, kann man den anderen Wert mit einer der folgenden Gleichungen berechnen.

$$Vo = 5V \times \frac{2000\Omega}{2000\Omega + R} \qquad R = \left(5V \times \frac{2000\Omega}{Vo}\right) - 2000\Omega$$

1.4 V nennt sich die **Schwellenspannung (threshold voltage)** der BASIC Stamp I/O Pins, alias **logischer Grenzwert (logic threshold)**. Wenn die von einem I/O-Pin gemessene Spannung über der Schwellspannung liegt, dann speichert das Input Register dieses I/O-Pins eine 1 (logisch high). Wenn sie unter dem Schwellwert liegt, speichert das Input Register dieses I/O-Pins ein 0 (logisch Low).

Schatten erkennen

Schattenwurf erhöht den Widerstandswert (R), was wiederum Vo senkt. 2 kΩ Widerstände wurden so gewählt, dass der Wert von Vo in einem gut beleuchteten Zimmer etwas oberhalb des 1.4V Schwellwertes der BASIC Stamp I/O Pins verharrt. Wenn man z.B. mit der Hand Schatten auf den Sensor fallen lässt, sollte das Vo unter den 1.4V Schwellwert absenken.

In einem gutbeleuchteten Zimmer speichern sowohl **`IN6`** als auch **`IN3`** den Wert1. Wen Sie einen Schatten auf den Fotowiderstands-Spannungsteiler an P6 fallen lassen, wird **`IN6`** eine 0 speichern. Entsprechend, wenn Sie den Schatten auf den Fotowiderstands-Spannungsteiler an P3 fallen lassen, wird dies eine 0 in **`IN3`** zur Folge haben.

Beispielprogramm: TestPhotoresistorsDividers.bs2

Dieses Beispielprogramm ist ein an die Fotowiderstands-Spannungsteiler angepasstes TestWhiskers.bs2. Statt P5 und P7 zu überwachen wie bei den Fühlern, überwachen wir nun P3 und P6, die an die Fotowiderstands-Spannungsteilerschaltungen angeschlossen sind. Dieses Programm sollte in einem gut beleuchteten Zimmer auf beiden Seiten einen Wert 1 anzeigen. Wenn Sie Schatten auf einen oder beide Fotowiderstände werfen sollten ihre entsprechenden Werte auf 0 wechseln.

- √ Schalten Sie die Stromversorgung an Ihrem Board wieder ein.
- √ Erfassen, speichern uns starten Sie TestPhotoresistorDividers.bs2.
- √ Überzeugen Sie sich, dass ohne Schatten sowohl **IN6** als auch **IN3** den Wert 1 speichern.
- √ Überzeugen Sie sich, dass Sie mit Ihrer Hand auf jedem der Fotowiderstände Schatten machen können und dadurch die Input Register von 1 auf 0 wechseln.
- √ Wenn Sie damit Probleme haben, entweder durch den Schatten das Input Register auf 0 zu ändern, oder wenn das Input Register eine 0 speichert, egal ob mit oder ohne Schatten, dann konsultieren Sie den Abschnitt "Fehlersuche im Fotowiderstands-Spannungsteiler" unten.. Arbeiten Sie daran bis der Schattenwurf Ihrer Hand auf dem Fotowiderstand den Status zuverlässig von 1 nach 0 ändert.


```
' Robotics with the Boe-Bot - TestPhotoresistorDividers.bs2
' Display what the I/O pins connected to the photoresistor
' voltage dividers sense.

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

DEBUG "PHOTORESISTOR STATES", CR,
      "Left       Right", CR,
      "-------    --------"

DO
  DEBUG CRSRXY, 0, 3,
        "P6 = ", BIN1 IN6,
        "     P3 = ", BIN1 IN3
  PAUSE 100
LOOP
```

Fehlersuche im Fotowiderstands-Spannungsteiler

Allgemeine Kontrollen::

- √ Überprüfen Sie Ihre Verdrahtung und das Programm auf Fehler.
- √ Versichern Sie sich, dass jedes Bauteil fest in seinen Sockel gesteckt ist.
- √ Überprüfen Sie den Farbcode an Ihren Widerständen. Die Widerstände zwischen VSS und den Fotowiderständen sollten 2 kΩ (rot-schwarz-rot) sein. Die Widerstände die P6 und P3 mit den Fotowiderständen verbinden sollten 220 Ω (rot-rot-braun) sein.

Wenn das IN3 oder das IN6 Register eine 0 anzeigt, egal ob Sie Schatten machen oder nicht:

- √ Wenn der Raum eher schwach beleuchtet ist, versuchen Sie zusätzliche Lampen hineinzustellen. Alternativ können Sie die 2 kΩ Widerstände durch 4.7 kΩ Widerstände (gelb-violett-rot) ersetzen. Das verbessert die Leistung des Spannungsteilers in schwachem Licht. Für echte Schwachlichtsituationen können Sie sogar die 10 kΩ Widerstände (braun-schwarz-orange) verwenden.

Wenn das IN3 oder das IN6 Register eine 1 anzeigt, egal ob Sie Schatten machen oder nicht:

- √ Wenn der Raum sehr hell beleuchtet ist, und sie Ihre Hand um die lichtempfindliche Seite des Fotowiderstandes legen müssen, um aus der 1 eine 0 zu machen, müssen Sie vielleicht einen kleineren Widerstandwert anstelle der 2 kΩ einsetzen. Versuchen Sie es mit 1 kΩ Widerstand (braun-schwarz-rot) oder sogar einem 470 Ω Widerstand (gelb-violett-braun) wenn Sie draussen sind.

Sie sind dran – Experimente mit Spannungsteilern

Abhängig von den Lichtverhältnissen in Ihrem Robotik-Labor können grössere oder kleinere Serienwiderstände anstelle der 2 kΩ Widerstände die Leistungsfähigkeit Ihrer Schattensensoren verbessern.

- √ Denken Sie daran die Stromversorgung bei jeder Änderung an den Schaltkreisen auszuschalten.
- √ Versuchen Sie die 2 kΩ (rot-schwarz-rot) Widerstände mit jedem der anderen Widerstände auszutauschen, die Sie inzwischen beisammen haben: 470 Ω, 1 kΩ, 4.7 kΩ, and 10 kΩ.
- √ Testen Sie jede Spannungsteilerkombination mit TestPhotoresistorDividers.bs2 und finden Sie heraus, welche Widerstände in Ihren Lichtverhältnissen am besten funktionieren. Die beste Kombination ist eine, die nicht übermässig empfindlich ist, die aber auch anspricht, ohne dass Sie Ihre Hand direkt um den Fotowiderstand schliessen..

√ Verwenden Sie jene Widerstandskombination für die nächsten zwei Aktivitäten, die bei Ihnen am besten funktionieren.

AKTIVITÄT 2: HERUMFAHREN UND SCHATTEN UND GEGENSTÄNDEN AUSWEICHEN

Da die Fotowiderstands-Spannungsteiler sich so ähnlich wie die Fühler verhalten wollen wir untersuchen was nötig ist, um RoamingWithWhiskers.bs2 so zu adaptieren, dass es mit dem Fotowiderstands-Spannungsteiler funktioniert.

RoamingWithWhiskers.bs2 für die Fotowiderstände adaptieren

Das einzige, was Sie tun müssen, ist die **IF**…**THEN** Statements abzuändern, dass diese **IN6** und **IN3** überwachen, statt **IN7** und **IN5**. Figur 6-5 zeigt, wie man diese Änderungen vornimmt.

Figur 6-5: Modifizierung RoamingWithWhiskers.bs2 für die Anwendung mit Fotowiderstands-Spannungsteiler

```
' 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
```

Beispielprogramm: RoamingWithPhotoresistorDividers.bs2

√ Öffnen Sie das Programm RoamingWithWhiskers.bs2 von Seite 179, und speichern Sie es als RoamingWithPhotoresistorDividers.bs2.

√ Modifizieren Sie es wie Figur 6-5 gezeigt.

√ Schalten Sie den Strom an Ihrem Board und den Servos wieder ein.
√ Starten und Testen Sie das Programm.

Gleichzeitig Schatten auf beiden Fotowiderständen zu machen kann schwierig sein. Wenn der Boe-Bot vorwärts rollt, überprüft er die Fotowiderstände etwa 40 mal pro Sekunde. Sie müssen sich also ziemlich schnell bewegen, um zwischen zwei Pulsen beide Fotowiderständen zu beschatten. Um beide Fotowiderstände gleichzeitig auszulösen ist es zweckmässig, wenn Sie Ihre Hand schnell zwischen kein Schatten und Schatten bewegen. Alternativ können Sie Ihre Hand auch auf beide Fotowiderstände Schatten werfen lassen, während der Boe-Bot ein Manöver ausführt. Wenn das Programm von diesem Manöver zurückkommt und die Fotowiderstände erneut überprüft, sollte es erkennen, dass beide Fotowiderstände im Schatten liegen.

√ Überprüfen Sie, dass der Boe-Bot Schatten vermeidet, indem Sie Ihre Hand über die Fotowiderstände halten, um diese abzuschatten. Versuchen Sie es ohne Schatten, Schatten nur über dem rechten Fotowiderstands-Spannungsteiler (Schaltung an P3), Schatten nur über dem linken Fotowiderstands-Spannungsteiler (Schaltung an P7), und Schatten über beiden Fotowiderstands-Spannungsteilern.
√ Führen Sie die Kommentare wie Titel und die Beschreibungen der Reaktionen auf Fühlerdrücken nach, damit diese das Verhalten der Fotowiderstandsschaltung widerspiegeln. Es sollte etwa so aussehen wie das Programm unten, wenn Sie damit fertig sind.

```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - RoamingWithPhotoresistorDividers.bs2
' Boe-Bot detects shadows photoresistors voltage divider circuit and turns
' away from them.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

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

pulseCount     VAR     Byte                ' FOR...NEXT loop counter.

' -----[ Initialization ]--------------------------------------------------

FREQOUT 4, 2000, 3000                      ' Start/restart signal.

' -----[ Main Routine ]----------------------------------------------------

DO
```

```
IF (IN6  = 0) AND (IN3 = 0) THEN            ' Both photoresistors detects
  GOSUB Back Up                             ' shadow, back up & U-turn
  GOSUB Turn Left                           ' (left twice).
  GOSUB Turn Left
ELSEIF (IN6  = 0) THEN                      ' Left photoresistor detects
  GOSUB Back_Up                             ' shadow, back up & turn right.
  GOSUB Turn_Right
ELSEIF (IN3  = 0) THEN                      ' Right photoresistor detects
  GOSUB Back Up                             ' shadow, back up & turn left.
  GOSUB Turn Left
ELSE                                        ' Neither photoresistor detects
  GOSUB Forward_Pulse                       ' shadow, apply a forward pulse.
ENDIF

LOOP

' -----[ Subroutines ]------------------------------------------------------

Forward_Pulse:                              ' Send a single forward pulse.
  PULSOUT 12,650
  PULSOUT 13,850
  PAUSE 20
  RETURN

Turn_Left:                                  ' Left turn, about 90-degrees.
  FOR pulseCount = 0 TO 20
    PULSOUT 12, 650
    PULSOUT 13, 650
    PAUSE 20
  NEXT
  RETURN

Turn Right:
  FOR pulseCount = 0 TO 20                  ' Right turn, about 90-degrees.
    PULSOUT 12, 850
    PULSOUT 13, 850
    PAUSE 20
  NEXT
  RETURN

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

Sie sind dran – Verbesserte Leistungsfähigkeit

Sie können die Leistung Ihres Boe-Bot's erhöhen indem Sie einige der Subroutinen-Aufrufe auskommentieren, die dafür da waren, damit der Boe-Bot rückwärts von Hindernissen wegfährt und sich dann umdreht, um sie zu umfahren. Auskommentieren bedeutet, ein Apostroph (Hochkomma) an den Anfang der Zeile zu setzen. Der Boe-Bot betrachtet diese Zeile dann als reinen Kommentar ohne Programmbefehle, und ignoriert sie. So bleiben die Zeilen zwar (für spätere Modifikationen) unverändert im Programm stehen, werden aber nicht mehr ausgeführt. Figur 6-6 zeigt ein Beispiel, in dem die zwei **`Turn_Left`** Subroutinenaufrufe im **`IF…THEN`** Statement, für den Fall wo beide Fotowiderstände Schatten haben, auskommentiert wurden.. Im Fall wo nur ein einzelner Fotowiderstand Schatten hat, sind die Aufrufe der **`Back_Up`** Subroutine auskommentiert, so dass der Boe-Bot als Reaktion auf Schatten nur eine Drehung ausführt.

Figur 6-6: Modifiziere RoamingWithPhotoresistorDividers.bs2

```
' 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
```

- √ Ändern Sie RoamingWithPhotoresistorDividers.bs2 wie auf der rechten Seite von Figur 6-6 gezeigt.
- √ Starten Sie das Programm und vergleichen Sie die Leistung.

AKTIVITÄT 3: EIN REAKTIONSFREUDIGERER SCHATTENGESTEUERTER BOE-BOT

Indem man die **FOR**…**NEXT** Schleifen in den Navigationssubroutinen eliminiert, kann man den Boe-Bot spürbar reaktionsfreudiger machen. Das war nicht möglich mit den Fühlern, da der Boe-Bot ja zuerst rückwärts vom Hindernis wegfahren musste, bevor er drehen konnte, da er ja bereits Körperkontakt mit dem Hindernis hatte. Wenn Sie Schatten zur Führung des Boe-Bot verwenden, kann er zwischen den Impulsen jedes Mal prüfen, ob es noch Schatten hat, unabhängig davon, ob er vorwärts fährt oder ein Manöver ausführt.

Ein einfacher schattengesteuerter Boe-Bot

Eine interessante Form von Fernsteuerung ist, den Boe-Bot zunächst in normalem Licht stillsitzen zu lassen, und dann einem Schatten zu folgen, den Sie über die Fotowiderstände fallen lassen. Es ist eine Art benutzerfreundliche Art, die Bewegung eines Boe-Bot zu lenken.

Beispielprogramm: ShadowGuidedBoeBot.bs2

Wenn Sie das nächste Programm laufen lassen, sollte der Boe-Bot stillstehen, wenn die Fotowiderstände nicht im Schatten liegen. Wenn Sie beide Fotowiderstände abschatten sollte der Boe-Bot vorwärts fahren. Wenn Sie nur den einen Fotowiderstand abschatten, sollte der Boe-Bot sich in die Richtung desjenigen Fotowiderstandes drehen, der den Schatten erkennt.

- √ Erfassen, speichern und starten Sie ShadowGuidedBoeBot.bs2.
- √ Schatten Sie die Fotowiderstands-Spannungsteiler mit der Hand ab.
- √ Lesen Sie dieses Programm sorgfältig und sorgen Sie dafür, dass Sie verstanden haben, wie es funktioniert. Es ist sehr kurz, aber wirkungsvoll.

```
' Robotics with the Boe-Bot - ShadowGuidedBoeBot.bs2
' Boe-Bot detects shadows cast by your hand and tries to follow them.

' {$STAMP BS2}                                ' Stamp directive.
' {$PBASIC 2.5}                               ' PBASIC directive.

DEBUG "Program Running!"
FREQOUT 4, 2000, 3000                         ' Start/restart signal.

DO

  IF (IN6  = 0) AND (IN3 = 0) THEN            ' Both detect shadows, forward.
    PULSOUT 13, 850
```

```
    PULSOUT 12, 650
  ELSEIF (IN6  = 0) THEN                    ' Left detects shadow,
    PULSOUT 13, 750                         ' pivot left.
    PULSOUT 12, 650
  ELSEIF (IN3  = 0) THEN                    ' Right detects shadow,
    PULSOUT 13, 850                         ' pivot right.
    PULSOUT 12, 750
  ELSE
    PULSOUT 13, 750                         ' No shadow, sit still
    PULSOUT 12, 750
  ENDIF

  PAUSE 20                                  ' Pause between pulses.

LOOP
```

Wie das Programm funktioniert

Das **IF…THEN** Statement im **DO…LOOP** wartet auf eine von vier möglichen Schattenbedingungen: beide, links, rechts, keine. Abhängig von der Bedingung übermitteln **PULSOUT** Befehle Pulse für eines der folgenden Manöver: Vorwärts, Drehung rechts, Drehung links, oder Stillstehen. Unabhängig von der Bedingung, eine der vier Puls-Sequenzen wird bei jedem Durchlauf des **DO…LOOP** übermittelt. Denken Sie daran, nach dem **IF…THEN** Statement eine **PAUSE 20** einzufügen, um die Low-Zeit zwischen jedem Paar von Servo-Pulsen sicherzustellen.

Sie sind dran – Verdichten des Programms

Dieses Programm benötigt weder die **ELSE** Bedingung, noch die zwei folgenden **PULSOUT** Befehle. Wenn Sie keine Pulse senden, steht der Boe-Bot genauso still, wie wenn Sie Pulse senden mit 750 als **PULSOUT** ***Duration*** Argument.

- √ Löschen Sie den folgenden Codeblock (oder kommentieren Sie ihn aus).

```
ELSE
  PULSOUT 13, 750
  PULSOUT 12, 750
```

- √ Starten Sie das geänderte Programm.
- √ Können Sie einen Unterschied im Verhalten des Boe-Bot erkennen?

AKTIVITÄT 4: HOLEN SIE MEHR INFORMATION AUS IHREN FOTOWIDERSTÄNDEN HERAUS

Die einzige Information, welche die BASIC Stamp aus den Fotowiderstands-Spannungsteilerschaltungen bekommen konnte war, ob die Lichtstärke über oder unter einer bestimmten Schwelle lag. Diese Aktivität zeigt eine andere Schaltung, welche die BASIC Stamp überwachen kann, und mit der sie genügend Information sammeln kann, um relative Lichtstärken zu unterscheiden. Die Werte, welche die BASIC Stamp aus dem Schaltkreis erhält schwanken zwischen kleinen Zahlen, die helles Licht bedeuten und grossen Zahlen, die wenig Licht anzeigen. Das bedeutet, dass man nicht mehr manuell die Serienwiderstände austauschen muss, um den Boe-Bot an unterschiedliche Lichtverhältnisse anzupassen. Statt dessen können Sie Ihr Programm anpassen, um verschiedene Wertebereiche zu beachten.

Einführung des Kondensators

Ein Kondensator (capacitor) ist ein Bauteil, das elektrische Ladung speichert und ein grundlegendes Element in vielen Schaltkreisen. Die Ladungsmenge, die ein Kondensator speichern kann (Kapazität), wird in Farad (F) gemessen. Ein Farad ist ein sehr grosser Wert und kommt im Zusammenhang mit dem Boe-Bot nicht vor. Die Kondensatoren, die in dieser Aktivität verwendet werden, speichern Bruchteile eines Millionstels eines Farads. Ein Millionstel eines Farad heisst Mikrofarad (microfarad), abgekürzt µF. Der Kondensator, den Sie in dieser Übung verwenden, speichert einen Hundertstel eines Millionstel Farad. Das sind 0.01 µF.

Normale Messgrössen für elektronische Kapazität sind:

- Mikrofarad:
  (Millionstel eines Farad), abgekürzt µF — 1 µF = 1×10^{-6} F
- Nanofarad:
  (Milliardstel eines Farad), abgekürzt nF — 1 nF = 1×10^{-9} F
- Picofarads:
  (Billionstel eines Farad), abgekürzt pF — 1 pF = 1×10^{-12} F

Die 103 auf dem 0.01 µF Kondensatorgehäuse ist eine Kapazitätsangabe in Picofarad oder pF. 103 bedeutet 10, mit drei Nullen dahinter, das entspricht 10,000. So passen die 103 zu den 0.01 µF.

10,000 ist 10×10^3.

$(10 \times 10^3) \times (1 \times 10^{-12})$ F = 10×10^{-9} F

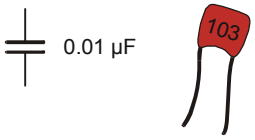
das ist auch 0.01×10^{-6} F

das ist 0.01 µF.

Figur 6-7 zeigt das Schemasymbol für einen 0.01 µF Kondensator zusammen mit einer Bauteilzeichnung des Bauteils in Ihrem Boe-Bot Kit. Die Bezeichnung 103 auf dem Kondensator gibt seinen Wert an.

Bauteile:

(2) Fotowiderstände - CDS
(2) Kondensatoren – 0.01 µF (103)
(2) Widerstände - 220 Ω (rot-rot-braun)
(2) Verbindungsdrähte



Figur 6-7 Kondensator Schema Symbol und Bauteilzeichnung

Es kann auch noch 0.1 µF Kondensatoren mit der Bezeichnung 104 in Ihrem Kit haben. Diese sollten Sie hier nicht verwenden.

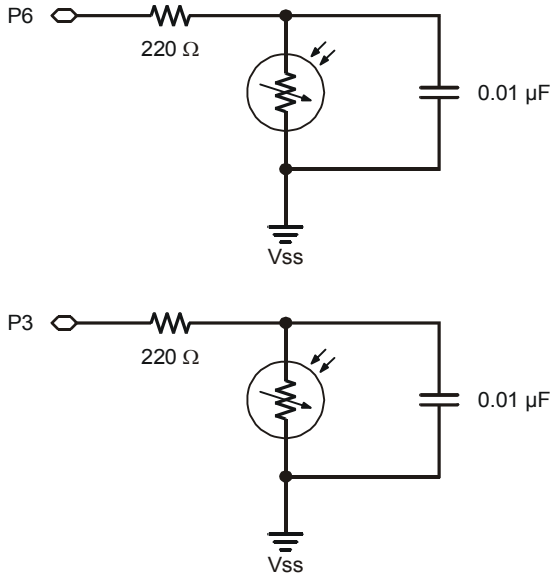
√ Kontrollieren Sie, dass Sie die 0.01 µF Kondensatoren (markiert 103) für diese Aktivität genommen haben.

Die 0.1 µF Kondensatoren können in hell beleuchteten Bereichen verwenden werden, aber unter normalen und schwachen Lichtverhältnissen beeinträchtigen sie die Leistungsfähigkeit des Boe-Bot.

Neubau der lichtempfindlichen Augen

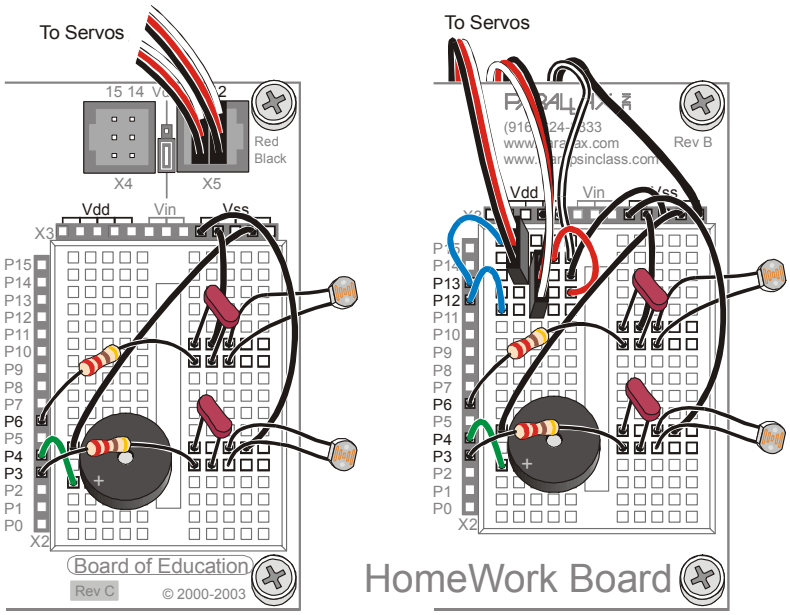
Der Schaltkreis, den die BASIC Stamp zur Erkennung von Lichtstärken verwenden kann, nennt man eine Widerstands/Kondensator- Schaltung, kurz RC (resistor/capacitor) – Schaltung. (In der Elektronikliteratur ist es üblich, in Teilelisten, Schemazeichnungen etc. die Bauteile mit einem oder zwei Buchstaben abzukürzen. Dabei haben sich auch die Abkürzungen **R** (*resistor*) für Widerstände, **C** (*capacitor*) für Kondensatoren, **D** (*diode*) für Dioden oder **IC** (*integrated circuit*) für Integrierte Schaltungen eingebürgert. Es ist daher üblich, häufig vorkommende Schaltungen oder Schaltungsteile mit solchen Abkürzungen zu bezeichnen: RC-Netzwerk, R-2R-Leiter etc. [Anm.d.Übers.]) Figur 6-8 zeigt die Schemata der Boe-Bot RC Lichtdetektorschaltungen und Figur 6-9 zeigt Musterverdrahtungsdiagramme für das Board of Education und das HomeWork Board.

√ Schalten Sie den Strom am Board und den Servos ab.
√ Bauen Sie die RC Schaltung von Figur 6-8 ausgehend von Figur 6-9 auf.



Figur 6-8
Schema - Zwei RC Schaltungen mit Fotowiderstand

Zur Messung von lichtabhängigen Widerständen.

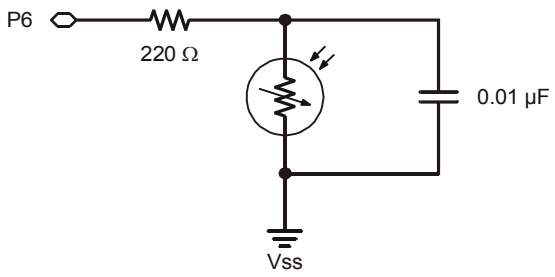


Figur 6-9
Verdrahtungs-Diagramme für Fotowider-standsschal-tungen

Board of Education (links) and HomeWork Board (rechts).

Über die RC-Abfallzeit und die Fotowiderstandsschaltung

Stellen Sie sich einen Kondensator in der Schaltung von Figur 6-10 als eine winzige aufladbare Batterie vor. Wenn P6 ein High-Signal sendet, lädt es im wesentlichen diese Kondensator-Batterie auf, indem es 5V an sie anlegt. Nach ein paar ms ist der Kondensator auf beinahe 5V aufgeladen. Wenn das BASIC Stamp Programm den I/O Pin umschaltet, so dass er still zuhört, gibt der Kondensator seine Ladung langsam durch den Fotowiderstand ab. Während der Kondensator seine Ladung abgibt, sinkt seine Spannung tiefer und tiefer. Die Zeit die es dauert, bis die Spannung, an **`IN6`** unter 1.4 V gefallen ist hängt davon ab, wie stark der Fotowiderstand dem elektrischen Strom aus dem Kondensator "Widerstand leistet". Wenn der Fotowiderstand wegen schummrigem Licht einen hohen Widerstandswert hat, benötigt der Kondensator für die Entladung länger. Wenn der Fotowiderstand einen geringen Widerstandswert hat, weil das Licht auf seiner Oberfläche sehr hell ist, wird er den Stromfluss nicht sehr stark behindern, und der Kondensator verliert seine Ladung sehr schnell.



Figur 6-10
RC Schaltung an I/O Pin

Parallel-Anschluss

Der Fotowiderstand und der Kondensator in Figur 6-10 sind parallel angeschlossen. Wenn zwei Komponenten parallel geschaltet sind, muss jeder ihrer Anschlüsse in ein gemeinsames Steckterminal (auch Knoten genannt) eingesteckt sein). Der Fotowiderstand und der Kondensator haben je einen Anschluss an Vss. Sie haben ebenfalls je einen Kontakt mit dem einen Anschluss des 220 Ω Widerstands.

Messung der RC-Abfallzeit mit der BASIC Stamp

Die BASIC Stamp kann darauf programmiert werden, den Kondensator aufzuladen und dann die Zeit zu messen, bis die Spannung des Kondensators auf 1.4V gefallen ist. Diese Abfallzeit-Messung kann als Indikator für die Lichtstärke verwendet werden die der Fotowiderstand tatsächlich detektiert. Diese Messung verlangt eine Kombination der **`HIGH`** und **`PAUSE`** Befehle zusammen mit dem neuen Befehl **`RCTIME`**. Der **`RCTIME`** Befehl ist dafür gemacht, die RC Abfallzeit (*decay time*) an einem Schaltkreis wie dem in Figur 6-10 zu messen. Hier ist die Syntax für den **`RCTIME`** Befehl:

RCTIME *Pin, State, Duration*

Das ***`Pin`*** Argument ist die Nummer des I/O Pins, den Sie messen wollen. Wenn Sie zum Beispiel P6 messen wollen, sollte das ***`Pin`*** Argument 6 lauten. Das ***`State`*** Argument kann entweder 1 oder 0 sein. Es wird auf 1 gesetzt, wenn die Spannung am Kondensator über 1.4 Volt startet und dann abnimmt. Wenn die Spannung am Kondensator unter 1.4V startet und dann ansteigt wird es auf 0 gesetzt. Da in der Schaltung von Figur 6-10, die Spannung über dem Kondensator bei nahezu 5V startet und dann auf 1.4V abfällt, muss das ***`State`*** Argument auf 1 gesetzt werden. Das ***`Duration`*** Argument muss eine Variable sein, welche die Zeitmessung – in 2 µs Schritten – speichert. Im nächsten Beispielprogramm messen wir die RC-Abfallzeit an der Fotowiderstandsschaltung an P6, das heisst am Fotowiderstand an der linken Seite des Boe-Bot. Wir verwenden eine Variable namens **`timeLeft`**.

Um den RC Spannungsabfall zu messen, müssen Sie zunächst mal sicherstellen, dass Sie die Variable deklariert haben, welche die Zeitmessung speichern soll:

```
timeLeft    VAR   Word
```

Die folgenden drei Codezeilen laden den Kondensator auf, messen die RC Abfallzeit und speichern das dann in der **`timeLeft`** Variable.

```
HIGH 6
PAUSE 3
RCTIME 6,1,timeLeft
```

Um den Messwert zu erhalten, implementiert der Code folgende drei Schritte:

1. Starte den Ladeprozess am Kondensator indem die Schaltung an 5 Volt gelegt wird (mit dem **`HIGH`** Befehl).
2. Verwende **`PAUSE`** um dem **`HIGH`** Befehl genug Zeit zu geben, den Kondensator in der RC-Schaltung zu laden.
3. Führe den **`RCTIME`** Befehl aus, welcher den I/O Pin auf Input schaltet, die Abfallzeit (von beinahe 5 V auf 1.4 V) misst, and die Messung in der **`timeLeft`** Variable speichert.

Beispielprogramm: TestP6Photoresistor.bs2

- √ Schalten Sie den Strom an Ihrem Board wieder ein.
- √ Erfassen, speichern und starten Sie TestP6Photoresistor.bs2.
- √ Beschatten Sie den Fotowiderstand an P6 und überzeugen Sie sich, dass die Zeitmessungen grösser werden, wenn die Umgebung dunkler wird.
- √ Richten Sie die lichtempfindliche Fläche des Fotowiderstandes direkt auf eine Deckenlampe oder leuchten Sie mit einer Taschenlampe direkt drauf. Die Zeitmessungen sollten sehr klein werden. Sie sollten darauf wieder grösser werden, wenn Sie mit dem Fotowiderstand graduell mehr von der Lichtquelle weg zeigen.

```
' Robotics with the Boe-Bot - TestP6Photoresistor.bs2
' Test Boe-Bot photoresistor circuit connected to P6 and display
' the decay time.

' {$STAMP BS2}                                ' Stamp directive.
' {$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
```


Sie sind dran

- √ Speichern Sie TestP6Photoresistor.bs2 als TestP3Photoresistor.bs2.
- √ Ändern Sie das Programm so, dass es die RC Abfallzeit am rechten Fotowiderstand, dem an P3, misst.
- √ Wiederholen Sie die Schatten- und Direktlichttests mit dem RC Schaltkreis an P3 und versichern Sie sich, dass er richtig funktioniert.

AKTIVITÄT 5: TASCHENLAMPENLICHTFOLGENDER BOE-BOT

In dieser Aktivität testen und kalibrieren Sie die Lichtsensoren Ihres Boe-Bot so, dass diese den Unterschied zwischen Umgebungshelligkeit und einem direkten Taschenlampenstrahl unterscheiden können. Darauf programmieren Sie den Boe-Bot so, dass er einem Taschenlampenstrahl folgt, der direkt auf den Boden vor dem Boe-Bot gerichtet wird.

Zusatzausrüstung

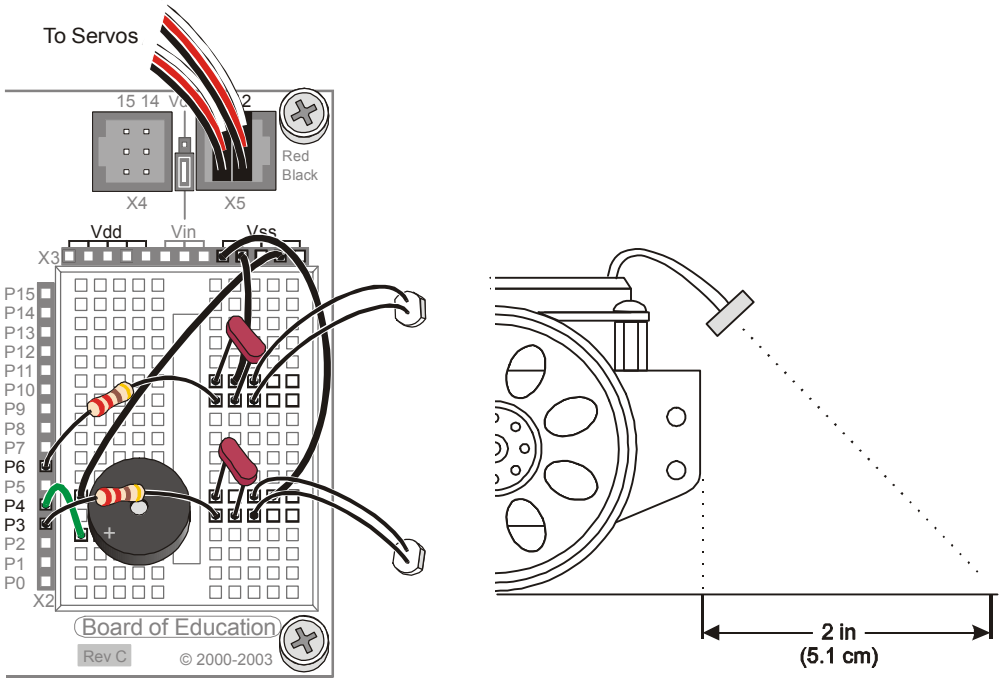
(1) Taschenlampe

Adjustierung der Sensoren für die Suche nach einem Taschenlampenstrahl

Diese Aktivität geht am besten, wenn die lichtempfindliche Fläche der Fotowiderstände geradeaus auf verschiedene Punkte auf dem Boden, etwa 2 in (5 cm) vor dem Boe-Bot, zeigen.

- √ Richten Sie die lichtempfindliche Fläche der Fotowiderstände so aus, dass sie zum Boden vor dem Boe-Bot zeigen wie in Figur 6-11.

Figur 6-11: Fotowiderstands-Ausrichtung



Test der Sensorreaktion auf einen Taschenlampenstrahl

Bevor Sie den Boe-Bot programmieren können, sich einem Taschenlampenstrahl zuzuwenden, müssen Sie den Unterschied zwischen den Lichtmessungen mit und ohne Taschenlampenlicht auf dem Weg vor dem Boe-Bot kennen.

Beispielprogramm: TestBothPhotoresistors.bs2

- √ Erfassen, speichern und starten Sie TestBothPhotoresistors.bs2.
- √ Stellen Sie den Boe-Bot auf die Fläche, wo er dem Taschenlampenlicht folgen soll. Sorgen Sie dafür, dass er immer noch mit dem seriellen Kabel angeschlossen ist und dass die Messungen im Debug Terminal angezeigt werden.
- √ Notieren Sie die Werte beider Zeitmessungen in der ersten Spalte von Table 6-1.

√ Schalten Sie Ihre Taschenlampe ein und fokussieren Sie den Lichtkegel vor dem Boe-Bot.
√ Ihre Zeitmessungen sollten nun deutlich tiefer sein als die erste Messreihe. Notieren Sie diese neuen Werte beider Zeitmessungen in der zweiten Spalte Table 6-1.

Table 6-1: RC-Zeitmessungen mit und ohne Taschenlampenstrahl

Zeitwerte		Beschreibung
timeLeft	**timeRight**	
		Zeitmessung ohne Taschenlampenlicht (Umgebungslicht).
		Zeitmessung mit dem Taschenlampenstrahl fokussiert vor dem Boe-Bot.


```
' Robotics with the Boe-Bot - TestBothPhotoresistors.bs2
' Test Boe-Bot RC photoresistor circuits.

' {$STAMP BS2}                                 ' Stamp directive.
' {$PBASIC 2.5}                                ' PBASIC directive.

timeLeft      VAR     Word                    ' Variable declarations.
timeRight     VAR     Word

DEBUG "PHOTORESISTOR VALUES", CR,             ' Initialization.
      "timeLeft  timeRight", CR,
      "--------  ---------"

DO                                            ' Main routine.

  HIGH 6                                      ' Left RC time measurement.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                      ' Right RC time measurement.
  PAUSE 3
  RCTIME 3,1,timeRight

  DEBUG CRSRXY, 0, 3,                         ' Display measurements.
        DEC5 timeLeft,
        "     ",
        DEC5 timeRight

  PAUSE 100
```

```
LOOP
```

Sie sind dran

- √ Versuchen Sie den Boe-Bot in verschiedene Richtungen zu drehen und wiederholen Sie die Messungen.
- √ Um bessere Ergebnisse zu erhalten können Sie Mittelwerte Ihrer Messungen für "mit Taschenlampenlicht" und "ohne Taschenlampenlicht" bilden und die Werte in Table 6-1 durch Ihre Mittelwerte ersetzen .

Dem Taschenlampenlicht folgen

Bis hierher haben Sie Variablen-Deklarationen verwendet. So vergibt zum Beispiel **`counter VAR Nib`** den Namen **`counter`** an einen speziellen Speicherplatz im RAM der BASIC Stamp. Nachdem Sie die Variable deklariert haben, benutzt das PBASIC Programm jedes Mal, wenn die Variable **`counter`** vorkommt, den Wert der an diesem speziellen Speicherplatz im RAM gespeichert ist..

Sie können auch Konstanten deklarieren . Mit anderen Worten, wenn Sie planen, eine Zahl in Ihrem Programm zu verwenden, geben Sie dieser einen nützlichen Namen. Statt der **`VAR`** Direktive verwenden Sie die **`CON`** Direktive. Hier sind einige **`CON`** Direktiven aus dem nächsten Beispielprogramm:

```
LeftAmbient     CON     108
RightAmbient    CON     114
LeftBright      CON     20
RightBright     CON     22
```

Überall, wo im Programm der Name **`LeftAmbient`** vorkommt, verwendet die BASIC Stamp nun die Zahl 108. Überall wo **`RightAmbient`** vorkommt, verwendet die BASIC Stamp den Wert 114. Entsprechend steht überall, wo **`LeftBright`** erscheint, eigentlich der Wert 20, und **`RightBright`** ist 22. Sie werden dann Ihre Werte aus der Table 6-1 einsetzen, bevor Sie das Programm starten.

Konstanten können sogar benutzt werden, um andere Konstanten zu berechnen. Hier ist ein Beispiel zweier Konstanten, genannt **`LeftThreshold`** und **`RightThreshold`** die aus den eben diskutierten vier anderen Konstanten berechnet werden. Die **`LeftThreshold`** und **`RightThreshold`** Konstanten werden in diesem Programm benutzt, um herauszufinden, ob oder ob nicht ein Taschenlampenstrahl erkannt wurde..

```
'                        Average                        Scale factor

LeftThreshold  CON     LeftBright + LeftAmbient / 2   * 5 / 8
RightThreshold CON     RightBright + RightAmbient / 2 * 5 / 8
```

Auf diesen Konstanten wird zunächst ein Mittelwert und dann eine Skalierung berechnet. Die Mittelwertberechnung für **`LeftThreshold`** ist **`LeftBright`** + **`LeftAmbient`** / 2. Das Ergebnis wird multipliziert mit 5 und dividiert durch 8. Das bedeutet, dass **`LeftThreshold`** eine Konstante mit dem Wert von $^{5}/_{8}$ des Mittelwerts aus **`LeftBright`** und **`LeftAmbient`**.

Mathematische Ausdrücke werden in PBASIC von links nach rechts ausgeführt. Zuerst wird **`LeftBright`** zu **`LeftAmbient`** addiert. Dieser Wert wird durch 2 dividiert. Das Resultat wird dann mit 5 multipliziert und durch 8 dividiert.

Versuchen Sie: **`LeftBright`** + **`LeftAmbient`** = 20 + 108 = 128.

128 / 2 = 64.

64 * 5 = 320

320 / 8 = 40

Sie können Klammern einsetzen um eine andere Reihenfolge der Berechnung zu erzwingen. Sie können zum Beispiel diese Codezeile PBASIC so umschreiben:

```
pulseRight = 2 - distanceRight * 35 + 750
```

like this:

```
pulseRight = 35 * (2 – distanceRight) + 750
```

In diesem Ausdruck wird 35 mit dem Ergebnis von (**`2 – distanceRight`**) multipliziert, und das Resultat wird dann zu 750 hinzuaddiert.

Beispielprogramm: FlashlightControlledBoeBot.bs2

- √ Erfassen Sie FlashlightControlledBoeBot.bs2 im BASIC Stamp Editor.
- √ Ersetzen Sie Ihre eigenen **`timeLeft`** Messergebnisse für ohne Taschenlampe (aus Table 6-1) anstelle des Wertes 108 in der **`LeftAmbient CON`** Direktive.
- √ Ersetzen Sie Ihre eigenen **`timeRight`** Messergebnisse für ohne Taschenlampe anstelle des Wertes 114 in der **`RightAmbient CON`** Direktive.
- √ Ersetzen Sie Ihre eigenen **`timeLeft`** Messergebnisse mit Taschenlampenstrahl anstelle des Wertes 20 in der **`LeftBright CON`** Direktive.
- √ Ersetzen Sie Ihre eigenen **`timeRight`** Messergebnisse mit Taschenlampenstrahl anstelle des Wertes 22 in der **`RightBright CON`** Direktive.

- √ Schalten Sie den Strom an Ihrem Board und den Servos wieder ein.
- √ Speichern und starten Sie FlashlightControlledBoeBot.bs2.
- √ Experimentieren Sie nun und finden Sie heraus, wo genau Sie den Lichtstrahl fokussieren müssen, damit die Vorwärts-, Links/Rechtsdrehungs-Bewegungen ausgeführt werden.
- √ Benutzten Sie nun Ihre Taschenlampe um Ihren Boe-Bot durch verschiedene Pisten mit Hindernissen und Kurven zu lenken.

```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - FlashlightControlledBoeBot.bs2
' Boe-Bot follows flashlight beam focused in front of it.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

' -----[ Constants ]-------------------------------------------------------

' REPLACE THESE VALUES WITH THE VALUES YOU DETERMINED AND ENTERED INTO
' TABLE 6.1.

LeftAmbient    CON     108
RightAmbient   CON     114
LeftBright     CON     20
RightBright    CON     22

'                       Average                       Scale factor

LeftThreshold  CON     LeftBright + LeftAmbient / 2   * 5 / 8
RightThreshold CON     RightBright + RightAmbient / 2 * 5 / 8

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

' Declare variables for storing measured RC times of the
' left & right photoresistors.

timeLeft       VAR     Word
timeRight      VAR     Word

' -----[ Initialization ]--------------------------------------------------

FREQOUT 4, 2000, 3000

' -----[ Main Routine ]----------------------------------------------------

DO
```

```
  GOSUB Test_Photoresistors
  GOSUB Navigate

LOOP

' -----[ Subroutine - Test_Photoresistors ]----------------------------------

Test Photoresistors:

  HIGH 6                                    ' Left RC time measurement.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                    ' Right RC time measurement.
  PAUSE 3
  RCTIME 3,1,timeRight

  RETURN

' -----[ Subroutine - Navigate ]-------------------------------------------

Navigate:

  IF (timeLeft < LeftThreshold) AND (timeRight < RightThreshold) THEN
    PULSOUT 13, 850                         ' Both detect flashlight beam,
    PULSOUT 12, 650                         ' full speed forward.
  ELSEIF (timeLeft < LeftThreshold) THEN    ' Left detects flashlight beam,
    PULSOUT 13, 700                         ' pivot left.
    PULSOUT 12, 700
  ELSEIF (timeRight < RightThreshold) THEN  ' Right detects flashlight beam,
    PULSOUT 13, 800                         ' pivot right.
    PULSOUT 12, 800
  ELSE
    PULSOUT 13, 750                         ' No flashlight beam, sit still.
    PULSOUT 12, 750
  ENDIF

  PAUSE 20                                  ' Pause between pulses.

  RETURN
```

Wie das Programm funktioniert

Dies sind die vier Konstantendeklarationen die Sie benutzten, mit Ihren eigenen Werten aus der Table 6-1.

```
LeftAmbient    CON     108
RightAmbient   CON     114
LeftBright     CON     20
RightBright    CON     22
```

Nachdem nun die vier Konstanten deklariert sind, ermitteln die nächsten beiden Zeilen die Schwellwerte für das Programm, indem sie die Werte mitteln und skalieren.. Diese Schwellwerte werden nun mit der aktuellen Messung für **timeLeft** und **timeRight** verglichen und so entschieden, ob die Fotowiderstände Umgebungslicht oder einen fokussierten Lichtstrahl wahrnehmen.

```
'                        Average                        Scale

LeftThreshold  CON     LeftBright + LeftAmbient / 2   * 5 / 8
RightThreshold CON     RightBright + RightAmbient / 2 * 5 / 8
```

Diese Variablen werden benutzt um die **RCTIME** Messungen zu speichern.

```
timeLeft       VAR     Word
timeRight      VAR     Word
```

Dies ist der Reset-Indikator, der in den meisten Programmen dieses Textes vorkommt.

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

Der Abschnitt der Hauptroutine enthält nur zwei Subroutinenaufrufe. Die echte Arbeit wird in diesem Programm von den zwei Subroutinen erledigt. **Test_Photoresistors** übernimmt die **RCTIME** Messungen für beide RC Fotowiderstandsschaltungen, und die **Navigate** Subroutine trifft die Entscheidungen und generiert die Servopulse.

```
DO

  GOSUB Test_Photoresistors
  GOSUB Navigate

LOOP
```

Dies ist die Subroutine welche die **RCTIME** Messungen an beiden Fotowiderstands - RC-Schaltungen vornimmt. Das Messergebnis aus dem linken Schaltkreis wird in der **timeLeft** Variablen, das Messergebnis aus dem rechten Schaltkreis in der **timeRight** Variablen gespeichert.

```
Test_Photoresistors:

  HIGH 6
  PAUSE 3
```

```
RCTIME 6,1,timeLeft

HIGH 3
PAUSE 3
RCTIME 3,1,timeRight

RETURN
```

Die Navigate Subroutine benutzt ein **IF**…**THEN** Statement um die **timeLeft** Variable mit der **LeftThreshold** Konstanten, und die **timeRight** Variable mit der **RightThreshold** Konstanten zu vergleichen. Erinnern Sie sich, wenn der **RCTIME** Messwert klein ist, bedeutet es, dass helles Licht wahrgenommen wurde, und wenn der Messwert gross ist, bedeutet dies eher düstere Beleuchtung. Wenn also eine der Variablen mit den aktuellen **RCTIME** Messwerten kleiner ist als die Schwellwert-Konstante, bedeutet das, dass ein Taschenlampenlicht entdeckt wurde, andernfalls wurde kein solches Licht entdeckt. Abhängig von der Situation, welche die Subroutine erkennt (beide, links, rechts, keiner), werden die korrekten Navigationsimpulse verwendet, gefolgt von einer **PAUSE**, bevor der **RETURN** Befehl die Subroutine wieder abschliesst.


```
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
```

Sie sind dran – Anpassen der Performance and Ändern des Verhaltens

Sie können die Performance des Programms verbessern, indem Sie den Skalierungsfaktor in der Konstantendeklaration verändern:

```
'                        Average                          Scale factor

LeftThreshold  CON     LeftBright + LeftAmbient / 2    * 5 / 8
RightThreshold CON     RightBright + RightAmbient / 2 * 5 / 8
```

Wenn Sie den Skalierungsfaktor von $^{5}/_{8}$ auf $^{1}/_{2}$ ändern, reagiert Ihr Boe-Bot weniger empfindlich auf den Lichtstrahl, was wiederum die Lichtsteuerung verbessern kann (oder auch nicht).

√ Probieren Sie verschiedene Skalierungsfaktoren auch wie $^{1}/_{4}$, $^{1}/_{2}$, $^{1}/_{3}$, $^{2}/_{3}$ und $^{3}/_{4}$ und notieren Sie die Unterschiede im Antwortverhalten des Boe-Bot auf den Lampenstrahl.

Indem Sie das **IF…THEN** Statement im Beispielprogramm ändern können Sie das Verhalten des Boe-Bot so ändern, dass er versucht, das Licht aus seinen Augen zu bekommen.

√ Ändern Sie das **IF…THEN** Statement so, dass der Boe-Bot rückwärts wegfährt, wenn er den Lichtstrahl mit beiden Fotowiderstandsschaltungen entdeckt, und dass er sich wegdreht, wenn er den Lichtstrahl nur mit einem seiner Fotoaugen sieht.

AKTIVITÄT 6: STREBEN NACH LICHT

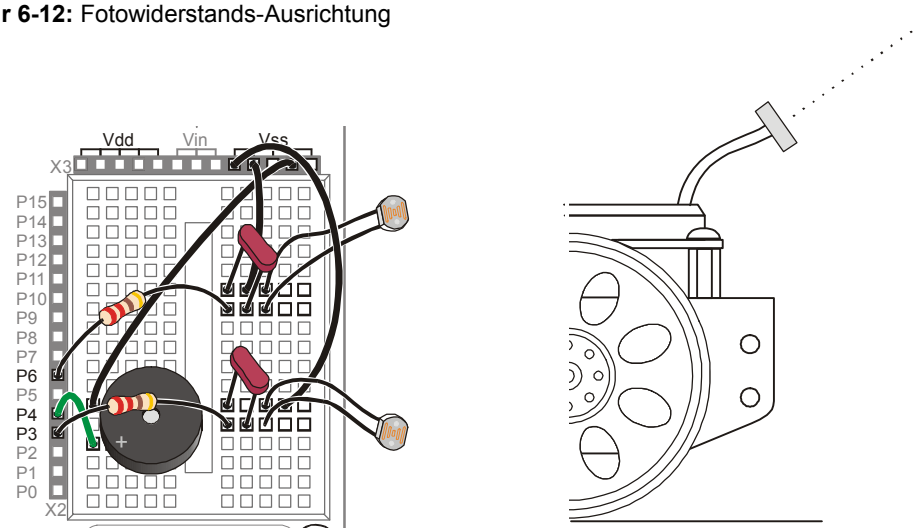
Das Beispielprogramm in dieser Aktivität kann dazu benutzt werden, den Boe-Bot aus einem ziemlich dunklen Raum in Richtung eines Türspalts zu leiten, durch den helleres Licht in den Raum fällt. Man kann damit auch die Kontrolle über das Fahrverhalten des Boe-Bot verbessern, wenn Sie mit Ihrer Hand Schatten auf den Fotowiderständen erzeugen.

Neuausrichtung der Fotowiderstände

Dies funktioniert am besten, wenn die lichtsensitiven Flächen der Fotowiderstände nach oben und aussen zeigen.

√ Zielen Sie mit den lichtempfindlichen Flächen Ihrer Fotowiderstände aufwärts und nach aussen, wie in Figur 6-12 gezeigt.

Figur 6-12: Fotowiderstands-Ausrichtung



Programmieren des Lichtsucher - Verhaltens

Der Schlüssel für das Streben nach helleren Lichtquellen heisst, geradeaus fahren, wenn die Unterschiede zwischen den Fotowiderstandsmessungen klein sind, und in Richtung des kleineren Messwerts fahren, wenn ein grosser Unterschied zwischen den beiden Messwerten vorliegt. Per Saldo bedeutet das, dass der Boe-Bot in Richtung des hellen Lichts dreht.

Zunächst sieht das nach einer ziemlich leichten Programmieraufgabe für **`IF`**…**`THEN`** Entscheidungsfindung aus; das Beispiel unten müsste eigentlich funktionieren. Das Problem ist, dass es nicht funktioniert, weil der Boe-Bot sich in Rechts- und Linksdrehungen festbeisst, weil die Änderungen in **`timeLeft`** und **`timeRight`** zu gross sind. Jedes Mal, wenn der Boe-Bot sich ein bisschen dreht, ändern die **`timeRight`** und **`timeLeft`** Variablen so stark, dass der Boe-Bot versucht, das zu korrigieren, und sich wieder zurückdreht. Er kommt einfach nicht dazu Vorwärtspulse zu generieren.

```
IF (timeLeft > timeRight) THEN              ' Turn right.
  PULSOUT 13, 850
  PULSOUT 12, 850
ELSEIF (timeRight > timeLeft) THEN          ' Turn left.
  PULSOUT 13, 650
```

```
  PULSOUT 12, 650
ELSE                                        ' Go forward.
  PULSOUT 13, 850
  PULSOUT 12, 650
ENDIF
```

Der folgende Codeblock funktioniert ein bisschen besser. Dieser Codeblock beseitigt das Problem mit dem Vor-/Zurückdrehen unter bestimmten Bedingungen. Die **`timeLeft`** Variable muss nun um mindestens 15 grösser sein, als die **`timeRight`** bevor der Boe-Bot einen Linkspuls produziert. Entsprechend muss **`timeRight`** um 15 grösser sein als **`timeLeft`** bevor der Boe-Bot nach links korrigiert. Das gibt dem Boe-Bot eine Chance, genügend Vorwärtspulse abzusetzen, bevor er wieder korrigieren muss – aber nur bei bestimmten Lichtverhältnissen.

```
IF (timeLeft > timeRight + 15) THEN         ' Turn right.
  PULSOUT 13, 850
  PULSOUT 12, 850
ELSEIF (timeRight > timeLeft + 15) THEN     ' Turn left.
  PULSOUT 13, 650
  PULSOUT 12, 650
ELSE                                        ' Go forward.
  PULSOUT 13, 850
  PULSOUT 12, 650
ENDIF
```

Das Problem mit diesem Codeblock ist, dass es nur in mittelhellem Licht funktioniert. Wenn man den Boe-Bot in eine deutlich dunklere Gegend bringt, beginnt der Boe-Bot wieder mit dem Drehen/Zurückdrehen-Tänzchen und bringt überhaupt keine Vorwärtspulse mehr auf die Räder. Wenn Sie den Boe-Bot an einen sehr hellen Ort bringen, fährt der Boe-Bot einfach geradeaus und macht gar nie eine Korrektur nach rechts oder links.

Warum passiert das?

Hier ist die Antwort: Wenn der Boe-Bot in einer dunklen Ecke des Raumes ist, sind die Messungen für jeden Fotowiderstand gross. Damit sich der Boe-Bot entscheidet, einer Lichtquelle zuzuwenden, müsste auch der Unterschied zwischen den zwei Messwerten gross (grösser als die oben verwendeten 15) sein. Wenn der Boe-Bot an einen hellen Platz gestellt wird, werden die Messwerte der Fotowiderstände kleiner. Damit der Boe-Bot sich für eine Drehung entscheiden kann muss auch die Differenz deutlich kleiner sein, als in einer dunkleren Gegend (also kleiner als die 15). Die Lösung für dieses

Dilemma ist, die entscheidende Differenz von den Umgebungsbedingungen abhängig zu machen. Im konkreten Fall definieren wir die Differenz als einen Bruchteil des Mittelwerts von **`timeRight`** und **`timeLeft`**. Damit hat sie immer den richtigen Wert, egal ob die Beleuchtung hell oder gedämpft ist.

```
average = timeRight + timeLeft / 2
difference = average / 6
```

Diese Variable **`difference`** kann nun im **`IF`**…**`THEN`** Statement verwendet werden, und wird entsprechend gross, wenn das Licht schwach ist, und klein, wenn das Licht hell ist.

```
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
```


Beispielprogramm: RoamingTowardTheLight.bs2

Im Gegensatz zu RoamingWithPhotoresistorDividers.bs2 auf Seite 200, reagiert dieses Programm sehr rasch auf den Schatten Ihrer Hand, egal ob das Licht hell oder gedämpft ist. Mit diesem Programm müssen auch keine Widerstände ausgewechselt und an die Lichtverhältnisse angepasst werden. Es berücksichtigt statt dessen die Lichtverhältnisse und die Software nimmt die Anpassungen mit Hilfe der Variablen **`average`** und **`difference`** automatisch vor.

Damit dieses Programm gut funktioniert, sollten Ihre Fotowiderstände vergleichbar auf ähnliche Lichtstärken reagieren. Wenn die RC Schaltungen grob unausgeglichen sind, werden Ihre Messwerte von Table 6-1 bei gleichen Lichtverhältnissen sehr unterschiedlich sein. Sie können diese Fehlanpassungen korrigieren, indem Sie die Techniken aus dem Appendix F: Balancing Photoresistors anwenden.

Dieses Programm misst den gesamten Mittelwert von **`timeLeft`** und **`timeRight`** und benutzt diesen um die **`difference`** zwischen den **`timeLeft`** und **`timeRight`** zu berechnen, die nötig ist, damit ein Drehpuls gerechtfertigt ist.

- √ Erfassen, speichern und starten Sie RoamingTowardTheLight.bs2
- √ Bringen Sie den Boe-Bot an verschiedene Orte, lassen Sie ihn laufen und überprüfen Sie, ob Sie seinen Kurs unabhängig von den Lichtverhältnissen ändern können, indem Sie einen der Fotowiderstands-RC-Schaltungen abschatten.
- √ Setzen Sie Ihren Boe-Bot auch versuchshalber in ein Zimmer, das nur schwach beleuchtet ist, aber in das durch eine Türöffnung helles Licht aus dem Nebenzimmer oder dem Flur fällt. Schauen Sie, ob Ihr Boe-Bot den Weg aus dem dunklen Zimmer findet.

```
' -----[ Title ]-------------------------------------------------------------
' Robotics with the Boe-Bot - RoamingTowardTheLight.bs2
' Boe-Bot roams, and turns away from dark areas in favor of brighter areas.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

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

' Declare variables for storing measured RC times of the
' left & right photoresistors.

timeLeft       VAR     Word
timeRight      VAR     Word
average        VAR     Word
difference     VAR     Word

' -----[ Initialization ]-----------------------------------------------------

FREQOUT 4, 2000, 3000

' -----[ Main Routine ]-------------------------------------------------------

DO

  GOSUB Test Photoresistors
  GOSUB Average And Difference
  GOSUB Navigate

LOOP

' -----[ Subroutine - Test Photoresistors ]-----------------------------------

Test_Photoresistors:

  HIGH 6                                   ' Left RC time measurement.
```

```
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                    ' Right RC time measurement.
  PAUSE 3
  RCTIME 3,1,timeRight

  RETURN

' -----[ Subroutine - Average And Difference ]-------------------------------

Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / 6

  RETURN

' -----[ Subroutine - Navigate ]-----------------------------------------------

Navigate:

  ' Shadow significantly stronger on left detector, turn right.
  IF (timeLeft > timeRight + difference) THEN
    PULSOUT 13, 850
    PULSOUT 12, 850
  ' Shadow significantly stronger on right detector, turn left.
  ELSEIF (timeRight > timeLeft + difference) THEN
    PULSOUT 13, 650
    PULSOUT 12, 650
  ' Shadows in same neighborhood of intensity on both detectors.
  ELSE
    PULSOUT 13, 850
    PULSOUT 12, 650
  ENDIF

  PAUSE 10

  RETURN
```

Warum PAUSE 10 statt PAUSE 20? Weil die **`Test_Photoresistors`** Subroutine zwei **`PAUSE`** Befehle hat, insgesamt 6 ms plus etwas extra Zeit für die Ausführung der **`RCTIME`** Befehle. Beide verlängern die Zeitdauer zwischen zwei Servopulsen, darum muss die **`PAUSE`** in der **`Navigate`** Subroutine reduziert werden. Nach einigem Herumexperimentieren mit verschiedenen Werten schien uns **`PAUSE 10`** insgesamt die zuverlässigste Leistung über den grössten Helligkeitsbereich zu geben.

Sie sind dran – Anpassung der Empfindlichkeit an Helligkeitsunterschiede

So wie das Programm jetzt steht ist die **difference** Variable das **average** dividiert durch 6. Sie können **average** durch einen kleineren Wert dividieren, wenn der Boe-Bot weniger empfindlich, oder durch eine grössere Zahl, wenn der Boe-Bot empfindlicher auf Helligkeitsunterschiede reagieren soll sein soll.

- √ Statt durch 6, dividieren Sie die **average** Variable einmal durch 3, 4, 5, 7 und 9.
- √ Starten Sie das Programm und testen Sie die Fähigkeit des Boe-Bot den Weg aus einem dunklen Raum zu finden mit jedem Teilerwert.
- √ Entscheiden Sie selbst, welches aus Ihrer Sicht der beste Teilerfaktor ist.

```
Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / 6

  RETURN
```

Sie können den Teiler auch als Konstante definieren, etwa so:

```
Denominator CON 6
```

Dann können Sie in Ihrer Average_And_Difference Subroutine die 6 (bzw. Ihre optimierte Zahl) durch die **Denominator** Konstante ersetzen, also so:

```
Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / Denominator

  RETURN
```

- √ Nehmen Sie diese Änderungen vor und überzeugen Sie sich, dass das Programm immer noch richtig funktioniert.

Sie können sich in diesem Programm noch eine weitere Variable sparen. Wie Sie sehen wird **average** Variable nur benutzt, um den Mittelwert vorübergehend zu speichern, dann wird er **Denominator** dividiert und in der **difference** Variable gespeichert. Die **difference** Variable wird später noch gebraucht, aber die **average** Variable nicht. Ein Weg, die Sache zu bereinigen ist, einfach die **difference** Variable anstatt der **average**

Variable zu verwenden. Das funktioniert bestens und die **average** Variable wird nicht mehr gebraucht. So sieht die Subroutine dann aus:

```
Average_And_Difference:

  difference = timeRight + timeLeft / 2
  difference = difference / Denominator

  RETURN
```

Es gibt aber noch eine elegantere Lösung.

√ Lassen Sie die **Average_And_Difference** Routine so:

```
Average_And_Difference:

  average = timeRight + timeLeft / 2
  difference = average / Denominator

  RETURN
```

√ Ändern Sie nun die Variablen Deklarationen wie folgt:

Figur 6-13: Änderungen an RoamingTowardTheLight.bs2 um ein Word des RAM zu sparen

```
' Unveränderter Code                ' Geändert, um RAM zu sparen

average         VAR     Word       average         VAR     Word
difference      VAR     Word       difference      VAR     average
```

Wir benötigen die **average** Variable nicht wirklich, aber das Programm erscheint viel sinnvoller für jemanden, der es versucht zu verstehen, wenn wir das Wort **average** in der ersten Zeile und das Wort **difference** in der zweiten Zeile verwenden. So deklariert man ein Alias (*alias name*) **difference** für die **average** Variable.

```
difference      VAR     average
```

Nun referenzieren sowohl **average** als auch **difference** auf dasselbe Wort im RAM.

√ Testen Sie das geänderte Programm und überprüfen Sie, ob es immer noch korrekt läuft.

ZUSAMMENFASSUNG

Dieses Kapitel befasste sich vor allem mit der Messung von Unterschieden in der Lichtstärke und der Programmierung des Boe-Bot, so dass er auf diese Unterschiede reagiert. Wir benutzen ein Paar Cadmiumsulfid-Fotowiderstände (CdS), um Differenzen im sichtbaren Licht zu messen. Die CdS-Fotowiderstände wurden zunächst mit Widerständen verbunden, um einen Spannungsteiler zu bilden. Die BASIC Stamp überwachte dann die Spannung an der Kontaktstelle zwischen dem Fotowiderstand und dem Festwiderstand. Wenn diese Spannung unter 1.4 V fiel oder darüber hinaus anstieg, wurde im Input Register des angeschlossenen I/O-Pin entsprechend eine 0 oder eine 1 gespeichert. Der Boe-Bot wurde programmiert, aufgrund dieser Binärwerte Entscheidungen zu treffen, analog zur Situation mit den Fühlern.

Die Schaltungstechnik mit dem Spannungsteiler funktioniert, solange man die richtigen Widerstände wählt und sich die Lichtverhältnisse nicht ändern. Allerdings gibt es eine viel universeller verwendbare Methode, Lichtstärken mit der BASIC Stamp zu messen: Man setzt den CdS Fotowiderstand in einem RC-Netzwerk ein, lädt den Kondensator auf und misst die Entladedauer. RC steht für Widerstand/Kondensator (engl: **r**esistor/**c**apacitor). Der Kondensator wurde in diesem Kapitel eingeführt zusammen mit einem Schaltkreis, der es der BASIC Stamp erlaubt, die RC-Entladedauer zu messen. Dies kann bei der BASIC Stamp am einfachsten mit dem RCTIME Befehl erreicht werden, der speziell dafür entwickelt wurde, RC Lade- und Entladezeiten zu messen.

Konstanten wurden als hilfreiche Möglichkeit eingeführt, wie in einem PBASIC Programm aussagekräftige Bezeichnungen anstelle von Zahlen eingesetzt werden können. Ebenfalls wurden die Mittelwertbildung und Skalierung eingeführt. Mittelwertbildung wurde benutzt, um einen Grenzwert dafür festzusetzen, ob oder ob nicht ein Lichtstrahl einer Taschenlampe festgestellt wurde. Sie wurde ausserdem verwendet, um die mittlere Helligkeit in einem Raum basierend auf den zwei RC-Zeitmessungen zu bestimmen. Diese wurde benutzt, um einen sich automatisch selbst an die allgemeinen Lichtverhältnisse anpassenden Grenzwert zu erzeugen. Damit kann man auf das Auswechseln der Widerstände im Falle von Änderungen der Lichtverhältnisse verzichten.

Fragen

1. Was bedeutet die Abkürzung LDR?

2. Wie reagiert der Widerstandswert eines Fotowiderstandes auf helles und schwaches Licht? Was passiert, wenn die Helligkeit zwischen hell und schwach drin liegen?
3. Hat ein I/O Pin irgendeinen Effekt auf einen Schaltkreis, wenn er als Input definiert wird? Was veranlasst das Input Register eines I/O Pin eine 0 oder eine 1 zu speichern, wenn der Pin als Input definiert ist?
4. Was bedeutet Schwellenspannung (engl: threshold voltage)? Wie hoch ist die Schwellenspannung eines I/O Pin der BASIC Stamp?
5. Was ist ein Spannungsteiler (engl: voltage divider)?
6. Hinsichtlich Figur 6-4 auf Seite 195, was veranlasst Vo über die Schwellenspannung eines I/O Pin der BASIC Stamp anzusteigen oder darunter zu fallen? Was ist mit der Schaltung, welche die Spannungsänderung von Vo bewirkt?
7. Welche Änderungen können Sie an der Schaltung von Figur 6-2 vornehmen, damit sie in einer hellbeleuchteten Umgebung besser funktioniert? Was für eine relativ dunkle Umgebung?
8. Welche Änderungen müssen Sie an RoamingWithWhiskers.bs2 mindestens vornehmen, damit es mit Fotowiderständen funktioniert?
9. Wie unterscheiden sich die Programme ShadowGuidedBoeBot.bs2 und RoamingWithPhotoresistorDividers.bs2? Welche Auswirkungen hat dies auf die Leistungsfähigkeit des Boe-Bot?
10. Was ist der Unterschied zwischen einem Farad (F) und einem Mikrofarad (µF)?
11. Welche Auswirkungen hat **`HIGH 6`** in Figur 6-10 auf Seite 209 auf die Spannung über dem Kondensator? Was bewirkt **`PAUSE 3`**? Was macht der **`RCTIME`** Befehl?
12. Was ist eine Konstanten-Deklaration? Was macht sie? Wie können Sie diese in einem Programm verwenden?
13. Wie werden mathematische Ausdrücke in PBASIC berechnet?
14. Welchen Unterschied macht es, ob in einer **`IF`**…**`THEN`** Bedingung eine Zahl oder eine Konstante verwendet wird? Welchen Unterschied macht es, ob wir in einem **`IF`**…**`THEN`** Befehl eine **`RCTIME`** Messung oder den Wert des Input Registers eines I/O Pin verwenden?
15. Welches sind die beiden Beispiele in diesem Kapitel, bei denen PBASIC benutz wurde, um einen Mittelwert zu berechnen? Wodurch unterscheiden sie sich? Wieweit sind sie gleich?
16. Sie können in der Variablendeklaration die Modifier: **`Bit`**, **`Nib`**, **`Byte`** und **`Word`** verwenden, aber können Sie auch einen Variablennamen deklarieren um eine

andere, bereits deklarierte Variable zu referenzieren? Wenn ja, wie funktioniert das? Wenn nein, warum nicht?

Übungen

1. Berechnen Sie Vo in Figur 6-4 auf Seite 195 für R = 10 kΩ. Ebenfalls für R = 30 kΩ.
2. Berechnen Sie Vo für Figur 6-4 auf Seite 195 für R = 20 kΩ. Wiederholen Sie diese Berechnung, aber ersetzen Sie den im Bild gezeigten 2 kΩ Widerstand mit folgenden Werten: 220 Ω, 470 Ω, 1 kΩ, 4.7 kΩ, 10 kΩ.
3. Wenn Vo in Figur 6-4 auf Seite 195 1.4 V ist, wie gross ist dann R? Wiederholen Sie diese Aufgabe mit Vo = 1 V und Vo = 3 V.
4. Schreiben Sie einen **`IF…THEN`** Ausdruck, der bewirkt, dass ein mit Fotowiderstands-Spannungsteilern ausgerüsteter Boe-Bot vorwärts fährt, wenn er helles Licht sieht, und rückwärts, wenn er schwaches Licht feststellt. Der **`IF…THEN`** Ausdruck soll auch bewirken, dass der Boe-Bot sich von Schatten abwendet.
5. Schreiben Sie eine Konstantendeklaration, die mit einem **`PAUSE`** Befehl für die Low-Zeit zwischen Servo Pulsen benutzt werden kann. Schreiben Sie den **`PAUSE`** Befehl so um, dass er die Konstante verwendet.
6. Wiederholen Sie die vorhergehende Aufgabe um die I/O Pins 13 und 12 zu verwenden. Schreiben Sie einige Beispielbefehle **`PULSOUT`**, welche Ihre Pin Konstanten verwenden.
7. Angenommen, Sie haben drei Variablen: **`firstValue`**, **`secondValue`** und **`thirdValue`**. Schreiben Sie einen Befehl, der den Mittelwert dieser drei Variablen in einer neuen Variablen namens **`myAverage`** erzeugt.. Schreiben Sie einen Befehl, der 7/8 dieses Mittelwertes in einer Variablen **`myScaledAverage`** speichert. Schreiben Sie die Variablendeklarationen, die nötig sind, damit Ihre Befehle in einem Programm funktionieren können, zuerst mit **`myAverage`** und **`myScaledAverage`** als separate Variablen, dann mit einer davon als Alias der anderen.

Projekte

1. Entwickeln Sie ein Programm, mit dem der Boe-Bot den Unterschied zwischen Schwarz und Weiss erkennt, wenn die Fotowiderstände vorne montiert und nach unten ausgerichtet sind. Suche Sie eine grosse weisse Fläche und legen Sie einige kräftig schwarze Blätter Papier darauf. Entwickeln Sie ein Programm, mit dem der Boe-Bot die schwarzen Blätter vermeidet. Hinweis: Sorgen Sie dafür,

dass Sie verstehen, was ein Boe-Bot sieht, wenn er auf den weissen Hintergrund schaut und was er sieht, wenn er auf ein schwarzes Papier fokussiert. Verwenden Sie Beispielprogramme aus den letzten drei Aktivitäten in diesem Kapitel.

Die RC Entladezeit-Schaltung und -Programme nützen für dieses Programm viel mehr, als die Fotowiderstand-Spannungsteilertechnik. Sorgen Sie auch dafür, dass dieser Hindernisparcours in einem gleichmässig ausgeleuchteten Bereich liegt. Helles Sonnenlicht von den Fenstern und Schattenwurf durch Zuschauer können die Demonstration beeinträchtigen.

2. Wenn Sie das Projekt 1 erfolgreich abgeschlossen haben, experimentieren Sie damit, den Boe-Bot so einzuschränken, dass er sich nur in einem durch schwarze Blätter abgegrenzten Bereich bewegen kann.
3. Entwickeln Sie einen Hindernisparcours dessen Hindernisse der Boe-Bot mit seinen Fühlern feststellen kann, und montieren Sie eine Tischlampe ans Ende der Strecke. Veranstalten Sie ein Wettrennen um herauszufinden, wer ein Programm schreiben kann, mit dem der Boe-Bot am effizientesten vom dunklen Startpunkt durch die Hindernisse zur Tischlampe kommt. Hinweis: Sie müssen die Schaltungen und Programme dieses Kapitels mit solchen aus früheren Kapiteln kombinieren. Verwenden Sie dabei auch wieder Programmelemente, die mit RC Schaltungen und **`RCTIME`** Befehlen Helligkeitsinformationen sammeln. Vermeiden Sie Schaltungen mit Foto-Spannungsteilern. Die Fotowiderstände sind zweckmässig, aber in Situationen wo die Helligkeit sorgfältig durch Lichtquellen und Abdeckungen gesteuert wird.

WARNUNG: Passen Sie auf, dass die Fühler die Anschlussdrähte der Fotowiderstände nicht berühren können! Testen Sie das zuerst mit einer Anzahl Versuchskollisionen, solange die Fotowiderstände noch nicht an Vdd, Vss, und die I/O Pins angeschlossen sind. Umwickeln Sie nötigenfalls Teile der Anschlussdrähte der Fotowiderstände mit Isolierband, um Sie von den Fühlern zu isolieren.

Kapitel 7: Navigation mit Infrarot-Scheinwerfern

Die heissesten Produkte scheinen heutzutage eine Gemeinsamkeit zu haben: Drahtlose Kommunikation. Personal Organizer senden Daten auf den Desktop Computer und mit der Fernbedienung können wir durch die TV-Kanäle zappen. Viele Personal Organizer, Handhelds und Fernbedienungen nutzen Signale im infraroten Frequenzbereich, unterhalb des sichtbaren Lichtspektrums, um zu kommunizieren. Mit ein paar kostengünstigen und leicht erhältlichen Bauelementen kann die BASIC Stamp ebenfalls Infrarotsignale senden und empfangen.

MIT INFRAROT-SCHEINWERFERN DIE STRASSE SEHEN

Um Objekte auch ohne Fühler erkennen zu können verlangt nichts so Kompliziertes wie elektronische Bilderkennung. Gewisse Roboter benutzen RADAR oder SONAR (manchmal auch als SODAR bezeichnet, wenn man es in der Luft statt im Wasser verwendet). Ein noch einfacheres System benutzt Infrarot-Licht (IR) um den Fahrweg des Roboters auszuleuchten und zu entscheiden, ob das Licht von einem Objekt reflektiert wird. Dank der grossen Verbreitung von IR-Fernbedienungen sind IR-Leuchtdioden und IR-Detektoren leicht erhältlich und billig.

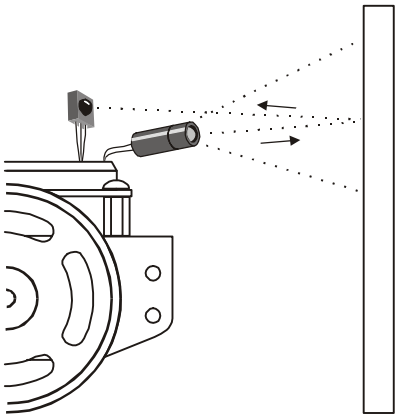
Infrarot: Infra heisst unter, also ist Infra-Rot licht (oder elektromagnetische Strahlung) von tieferer Frequenz, oder längerer Wellenlänge als rotes Licht. Table 7-1 zeigt die Wellenlängen für bekannte Farben gemeinsam mit dem Infrarotspektrum. Unsere IR LED und Detektoren arbeiten auf 980 nm (Nanometer), was als nahes Infrarot betrachtet wird. Nachtsichtbrillen und IR-Temperaturfühler benutzen ferne Infrarotwellenlängen von 2000-10,000 nm, je nach Anwendungsbereich. Table 7-1 zeigt die Wellenlängen für übliche Farben und das Infrarot-Spektrum

Table 7-1: Farben und approximative Wellenlängen in Nanometern (nm)

Farbe	Wellenlänge	Farbe	Wellenlänge
Violett	400	Rot	780
Blau	470	Nahes Infrarot	800-1000
Grün	565	Infrarot	1000-2000
Gelb	590	Fernes Infrarot	2000-10,000
Orange	630		

Infrarot Scheinwerfer

Das Infrarot-Objekterkennungssystem, das wir auf dem Boe-Bot aufbauen, entspricht den Scheinwerfern eines Autos in verschiedener Hinsicht. Wenn das Licht der Autoscheinwerfer von einem Hindernis reflektiert wird, entdecken Ihre Augen die Hindernisse und Ihr Hirn verarbeitet diese Information und lässt Ihren Körper das Auto entsprechend lenken. Der Boe-Bot verwendet Infrarot-LEDs als Scheinwerfer, wie Figur 7-1 gezeigt. Diese strahlen Infrarot ab, und in gewissen Fällen wird das Infrarot von einem Objekt reflektiert und in Richtung des Boe-Bot zurückgeworfen. Die Augen des Boe-Bot sind die Infrarotdetektoren. Diese senden Signale an die BASIC Stamp die anzeigen, ob sie reflektiertes Infrarot empfangen haben oder nicht. Das Hirn des Boe-Bot, die BASIC Stamp, entscheidet und steuert die Servomotoren basieren auf diesem Sensor-Input.



Figur 7-1
Objekterkennung mit IR Scheinwerfern

Der IR Detektor hat einen eingebauten optischen Filter, der ausser dem 980 nm Infrarot, das wir mit seiner internen Fotodiode erfassen wollen, nur wenig Licht durchlässt. Der Infrarot-Detektor hat auch einen elektronischen Filter der nur Signale von rund 38.5 kHz durchlässt. Mit anderen Worten, Der Detektor sucht nur nach Infrarot, das 38’500 mal pro Sekunde aufblitzt. Dies verhindert IR Störungen durch andere Infrarotquellen wie Sonnenschein oder Kunstlicht. Sonnenschein ist eine Gleichstrom-Störung (*DC interference*) (0 Hz), und Kunstlicht oszilliert in der Regel mit entweder 100 oder120 Hz, je nach lokaler Netzfrequenz (in Europa 50Hz, in den USA 60 Hz). Da 100 Hz weit ausserhalb der Durchlassfrequenz (*band pass frequency*) von 38.5 kHz des elektronischen Filters liegt, wird es von den IR-Detektoren völlig ignoriert.

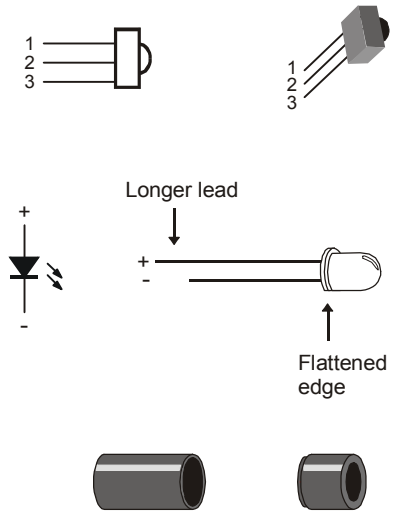
Einige Fluoreszenzröhren (*fluorescent lights*) generieren tatsächlich Signale, die von den IR-Detektoren aufgenommen werden können. Diese Leuchten können Störungen in den IR-Detektoren verursachen. Sie entwickeln deshalb unter anderem in diesem Kapitel einen Infrarot-"Schnüffler", den Sie zur Überprüfung der Fluoreszenzröhren rund um die Fahrpiste Ihres Boe-Bot verwenden können.

AKTIVITÄT 1: AUFBAU UND TEST DER IR PAARE

In diesem Abschnitt bauen und testen Sie die Infrarot Sender/Empfänger-Paare.

Stückliste:

(2) Infrarot-Detektoren (*infrared detectors*)
(2) Infrarot LED (klares Gehäuse)
(2) IR LED Abschirmungs-Kombi
(2) Widerstände - 220 Ω (rot-rot-braun)
(2) Widerstände – 1 kΩ (braun -schwarz- rot)



Figur 7-2 Neue Teile in diesem Kapitel

IR Detektor (oben)

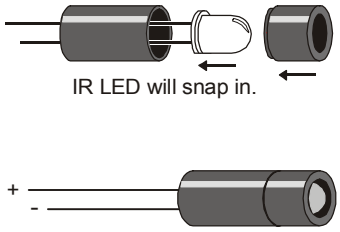
IR LED (mitte)

IR LED Schutzröhre (unten)

7

Montage der IR Scheinwerfer

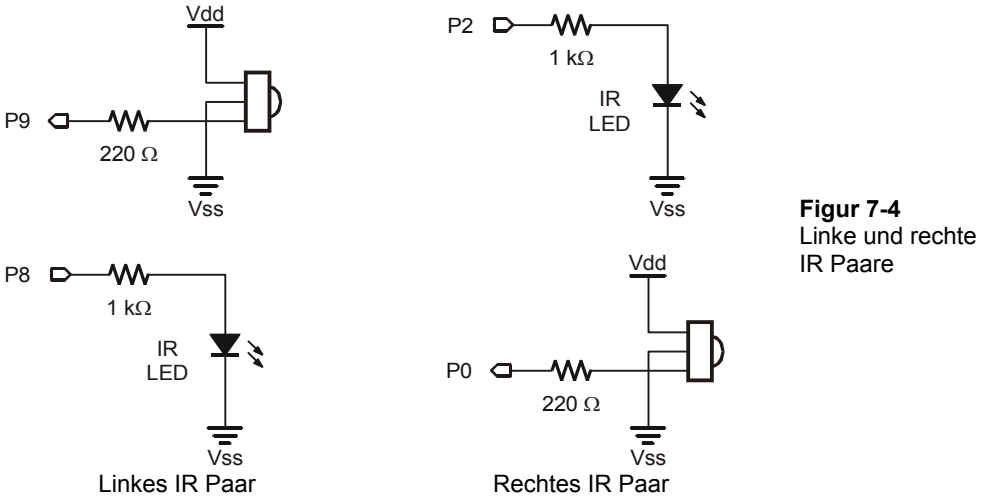
√ Bauen Sie die Infrarot-LED in das Abschirmungs-Kombi ein wie in Figur 7-3 gezeigt.
√ Vergewissern Sie sich, dass die LED im grösseren Abschirmungsteil einrastet.
√ Stülpen Sie den kleineren Abschirmungsteil über die LED und rasten Sie ihn im grösseren Teil ein.



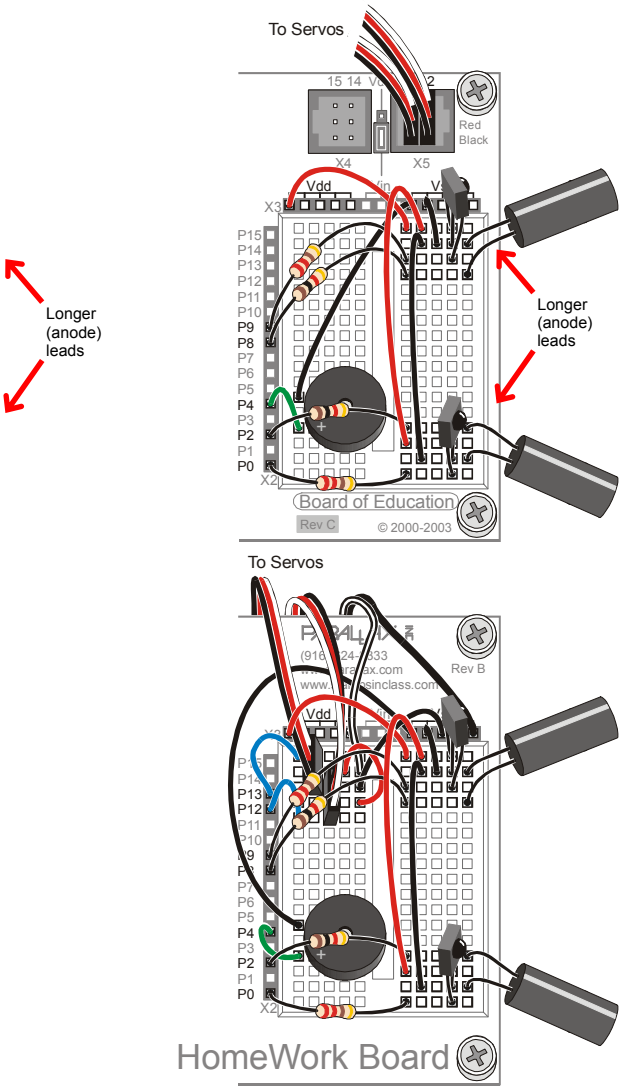
Figur 7-3
Einstecken der IR LED in die Schutzröhre

Je ein IR Paar (IR LED und Detektor) wird in den Ecken des Steckbrettes aufgebaut. Figur 7-4 zeigt die IR Scheinwerferschaltung als Schema und Figur 7-5 zeigt die Schaltung als Verdrahtungsdiagramm.

- √ Schalten Sie den Strom an Ihrem Board und den Servos aus.
- √ Bauen Sie den Schaltkreis wie im Schema von Figur 7-4 auf. Verwenden Sie das Verdrahtungsdiagramm von Figur 7-5 als Hinweis für die Platzierung der Bauteile.



Figur 7-4
Linke und rechte IR Paare



Figur 7-5 Verdrahtungs-Diagramme für Infrarot Sender und Empfänger Schaltungen

Board of Education (links) and HomeWork Board (rechts).

7

Test der IR-Paare mit dem FREQOUT-Trick

Der **FREQOUT** Befehl wurde hauptsächlich dafür erfunden, um Audiotöne zu erzeugen. Der tatsächliche Frequenzbereich des **FREQOUT** Befehls ist 1 bis 32768 Hz. Ein

interessantes Phänomen von digital synthetisierten Tönen ist, dass sie Signalanteile enthalten, die man Harmonische (*harmonics*) nennt. Eine Harmonische ist ein Ton höherer Frequenz, der in den erwünschten Ton eingemischt ist. Diese Tonfrequenzen liegen ausserhalb des menschlichen Hörbereichs, der etwa von 20 Hz bis 20 kHz reicht. Die vom **FREQOUT** Befehl generierten Harmonischen gehen von 32769 an aufwärts. Sie können diese Harmonischen direkt steuern, indem ***Freq1*** Argumente über 32768 verwenden. In dieser Aktivität verwenden Sie den Befehl **FREQOUT** 8, 1, 38500 um eine Harmonische von 38.5 kHz mit 1 ms Dauer an P8 senden. Die Infrarot-LED Schaltung an P8 wird diese Harmonische ausstrahlen. Wenn das Infrarotlicht von einem Objekt im Weg des Boe-Bot reflektiert wird, gibt Infrarot-Detektor der BASIC Stamp ein Signal, um Sie wissen zu lassen, dass das reflektierte Infrarotlicht detektiert worden ist.

FREQOUT Befehl – Grundton (*fundamental*) und Harmonische (*harmonics*)

Die Grundschwingung (*fundamental frequency*) ist der Wert des ***Freq1*** Arguments, wenn dieses 32768 oder weniger beträgt. Wenn Sie den **FREQOUT** Befehl verwenden um einen Ton in diesem Bereich zu senden, enthält er auch den verborgenen (harmonischen) Ton. Die Gleichung für die Harmonisch ist:

Harmonische = 65536 – Freq1,
wobei: Freq1 <= 32678

Wenn Sie den **FREQOUT** Befehl mit einem ***Freq1*** Argument über 32768 verwenden, um eine Harmonische zu senden, enthält diese den Grundton. Die Gleichung für den Grundton ist:

Grundton = 65536 – Freq1,
wobei: Freq1 > 32768

Entscheidend für das Funktionieren jedes IR LED/Detektor-Paares ist, dass Sie 1 ms der 38.5 kHz **FREQOUT** Harmonischen senden, und unmittelbar danach den Output des IR-Detektors in einer Variablen speichern. Hier ist ein Beispiel, das ein 38.5 kHz Signal an die IR LED an P8 sendet, und dann den Output des IR Detektors an P9 in einer Bit Variablen genannt **irDetectLeft** speichert.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9
```

Der Output des IR Detektors, wenn er kein IR Signal empfängt, ist High. Wenn der IR Detektor eine 38’500 Hz Harmonische sieht, die von einem Objekt reflektiert wurde, ist sein Output Low. Der Output des IR Detektors bleibt nur noch für den Bruchteil einer Millisekunde low, wenn der **FREQOUT** Befehl mit Senden der Harmonischen fertig ist. Es

ist darum unerlässlich den Output des IR Detektors unmittelbar nach dem **FREQOUT** Befehl abzuspeichern. Der in dieser Variablen gespeicherte Wert kann dann im Debug Terminal angezeigt, oder vom Boe-Bot für Navigationsentscheidungen verwendet werden.

Beispielprogramm: TestLeftIrPair.bs2

- √ Schalten Sie den Strom an Ihrem Board wieder ein.
- √ Erfassen, speichern und starten Sie TestLeftIrPair.bs2.

```
' Robotics with the Boe-Bot - TestLeftIrPair.bs2
' Test IR object detection circuits, IR LED connected to P8 and detector
' connected to P9.

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

irDetectLeft   VAR     Bit

DO

  FREQOUT 8, 1, 38500
  irDetectLeft = IN9

  DEBUG HOME, "irDetectLeft = ", BIN1 irDetectLeft
  PAUSE 100

LOOP
```


- √ Lassen Sie den Boe-Bot noch am seriellen Kabel angeschlossen, weil Sie den Debug Terminal für den Test der IR-Paare verwenden.
- √ Platzieren Sie ein Objekt, z.B. Ihre Hand oder ein Blatt Papier, senkrecht etwa 3 cm vor dem linken IR-Paar, wie in Figur 7-1 auf Seite 234 gezeigt.
- √ Kontrollieren Sie, dass das Debug Terminal eine 0 anzeigt, wenn Sie ein Objekt vor das IR-Paar halten, und dass es 1 anzeigt, wenn Sie das Objekt wieder entfernen.
- √ Wenn das Debug Terminal die erwarteten Werte für kein Objekt detektiert (1) und Objekt detektiert (0) anzeigt, gehen Sie direkt zum Abschnitt “Sie sind dran” anschliessend an das Beispielprogramm.
- √ Wenn das Debug Terminal nicht die erwarteten Werte anzeigt, probieren Sie die Schritte im Kasten “Fehlersuche” (*trouble shooting*).

Fehlersuche (*Trouble-Shooting*)

Wenn das Debug Terminal nicht die erwarteten Werte anzeigt, überprüfen Sie die Schaltung und das Programm auf Fehler.

Wenn Sie immer eine 0 erhalten, auch wenn kein Objekt vor dem Boe-Bot steht, könnte ein anderes Objekt in der Nähe das Infrarot reflektieren. Die Tischfläche vor dem Boe-Bot ist häufig schuld. Stellen Sie Ihren Boe-Bot so auf, dass die IR LED und der Detektor unmöglich von irgend einem reflektierenden Objekt in der Nähe beeinflusst werden können.

Wenn die Anzeige meistens 1 ist, wenn es kein Objekt vor dem Boe-Bot hat, aber gelegentlich eine 0 aufflackert, kann das heissen, dass Sie Störungen von einer Leuchtstoffröhre in der Nähe haben. Schalten Sie alle Leuchtstoffröhren ("Neonröhren") im Raum aus und wiederholen Sie Ihre Tests.

Sie sind dran

- √ Speichern Sie TestLeftIrPair.bs2 als TestRightIrPair.bs2.
- √ Ändern Sie den **`DEBUG`** Befehl, Titel und Kommentare, damit sie sich auf das rechte IR-Paar beziehen.
- √ Ändern Sie den Variablennamen von **`irDetectLeft`** nach **`irDetectRight`**. (Das muss an vier Stellen im Programm gemacht werden.)
- √ Ändern Sie das ***`Pin`*** Argument des **`FREQOUT`** Befehls von 8 auf 2.
- √ Ändern Sie das Input Register, das von **`irDetectRight`** überwacht wird von **`IN9`** auf **`IN0`**.
- √ Wiederholen Sie die Testschritte für das rechte IR-Paar mit der IR LED Schaltung an P2 und dem Detektor an P0.

AKTIVITÄT 2: FELDTEST ZUR OBJEKTERKENNUNG UND INFRAROTSTÖRUNG

In dieser Aktivität bauen und testen Sie Indikator-LEDs, die Ihnen auch ohne Debug Terminal anzeigen, ob ein Objekt erkannt wurde. Das ist praktisch, wenn Sie nicht gerade in der Nähe eines PC oder Laptop sind und Sie Fehler an Ihren IR-Detektorschaltungen beseitigen müssen. Sie schreiben ebenfalls ein Programm um nach Infrarotstörungen von Leuchtstoffröhren zu "schnüffeln". Gewisse Leuchtstoffröhren emittieren Signale die den Signalen Ihrer Infrarot-LEDs ähneln. Das Bauteil in einer Leuchtstoffröhrenfassung, das die Hochspannung für die Leuchte steuert, nennt man Vorschaltgerät (*ballast)* . Gewisse Vorschaltgeräte arbeiten im selben Frequenzbereich von 38.5 kHz wie der IR-Detektor, was dazu führt, dass die Lampe ebenfalls (IR-)Signale in dieser Frequenz emittieren. Wenn Sie die IR-Objekterkennung mit der

Navigationslogik verbinden, kann diese Interferenz zu bizarrem Verhalten des Boe-Bot führen!

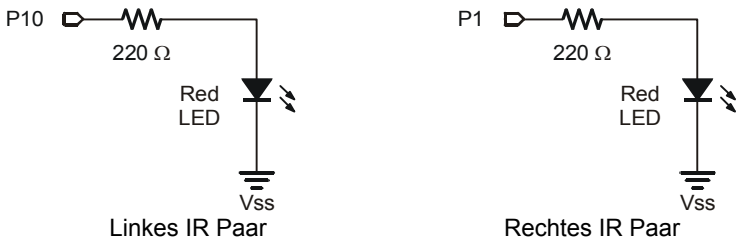
Wiederaufbau der LED Indikatorschaltungen

Das sind dieselben LED-Indikatorschaltungen, die Sie mit den Fühlern verwendet haben.

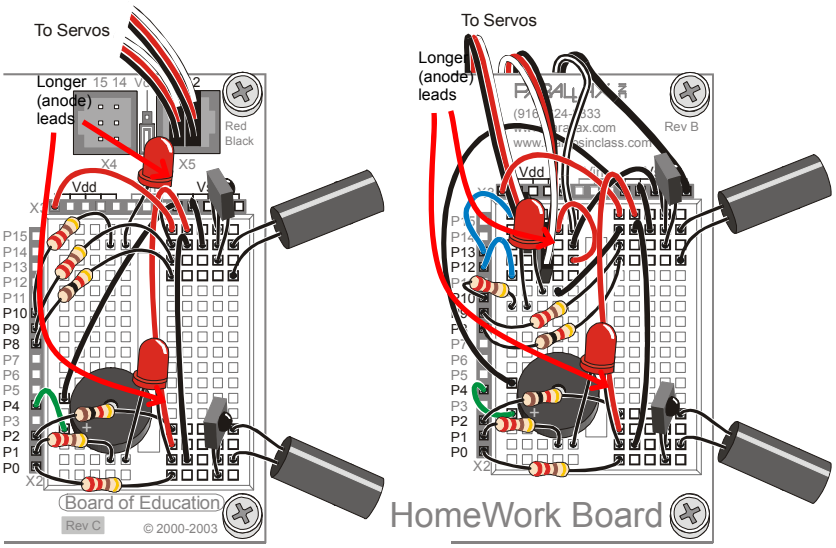
Parts List:

(2) Rote LEDs
(2) Widerstände – 220 Ω (rot-rot-braun)

√ Schalten Sie den Strom am Board und den Servos aus.
√ Bauen Sie die Schaltung nach Figur 7-6 mit Figur 7-7 als Referenz.



Figur 7-6
Linke und Rechte Indikator LEDs



Figur 7-7
Verdrahtungs-Diagramme für rote LED Indikatoren mit den IR Objekterkennungs-Schaltungen

Board of Education (links) and HomeWork Board (rechts).

Test des Systems

Es sind inzwischen einige Bauteile in diesem System vorhanden und dies erhöht die Wahrscheinlichkeit eines Verdrahtungsfehlers. Deshalb ist es wichtig ein Testprogramm zu haben das zeigt, was die Infrarot-Detektoren erfassen. Sie können das Programm verwenden um sicherzustellen, dass alle Schaltkreise richtig funktionieren, bevor Sie den Boe-Bot vom seriellen Kabel abhängen und andere Objekte testen.

Beispielprogramm: TestIrPairsAndIndicators.bs2

- √ Schalten Sie den Strom an Ihrem Board wieder ein.
- √ Erfassen, speichern und starten Sie TestIrPairsAndIndicators.bs2.
- √ Überprüfen Sie, dass der Lautsprecher einen klaren, hörbaren Ton von sich gibt, während das Debug Terminal “Testing Piezospeaker” anzeigt.
- √ Benutzen Sie das Debug Terminal um zu kontrollieren, dass die BASIC Stamp immer noch von beiden IR-Detektoren eine Null erhält, wenn ein Objekt davor gestellt wird.
- √ Kontrollieren Sie, dass die LED neben jedem Detektor aufleuchtet, wenn der Detektor ein Objekt erkennt. Wenn eine oder beide LED nicht richtig zu funktionieren scheinen, überprüfen Sie die Verdrahtung und das Programm.

```
' Robotics with the Boe-Bot - TestIrPairsAndIndicators.bs2
' Test IR object detection circuits.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

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

irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit

' -----[ Initialization ]--------------------------------------------------

DEBUG "Testing piezospeaker..."
FREQOUT 4, 2000, 3000

DEBUG CLS,
      "IR DETECTORS", CR,
      "Left   Right", 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
```

Sie sind dran – Abstandstest und Reichweitetest

Sie können nun die LED Detektoren benutzen, um Ihren Boe-Bot zu nehmen und Ihre IR Detektoren an Objekten zu überprüfen, die nicht gerade in Reichweite des seriellen Kabels Ihres Computers sind.

- √ Ziehen Sie das serielle Kabel aus Ihrem Boe-Bot und bringen Sie Ihren Boe-Bot zu einer Auswahl von Objekten und testen Sie die Reichweite Ihrer IR-Detektoren.
- √ Testen Sie die Erkennungsreichweite von verschiedenfarbigen Objekten. Welche Farbe wird aus der grössten Entfernung erkannt? Welche Farbe aus der kürzesten Distanz?

Schnüffeln nach IR-Interferenz

Wenn Sie zufällig bemerkt haben, dass Ihr Boe-Bot einen Kontakt anzeigte, obwohl nichts in der Nähe war, könnte das bedeuten, dass eine Lichtquelle in der Nähe eine IR-Strahlung mit einer Frequenz nahe bei 38.5 kHz erzeugt. Wenn Sie versuchen, einen Boe-Bot Wettbewerb oder eine Demo unter solchem Licht durchzuführen könnte die Leistung Ihres Infrarotsystems dabei ziemlich arm aussehen. Das Letzte, das sich jemand bei einer öffentlichen Vorführung wünscht, ist ein Roboter, der sich nicht wie angekündigt benimmt. Überprüfen Sie also Ihren künftigen Demo-Raum vorgängig mit dem folgenden IR-Interferenz – "Schnüffler" – Programm.

Das Konzept hinter diesem Programm ist einfach: Strahlen Sie kein IR-Licht durch die IR-LEDs, überwachen Sie lediglich die Detektoren um zu sehen, ob irgendeine IR-Strahlung festgestellt wird. Wenn IR festgestellt wird, löse den Alarm mit dem Piezolautsprecher aus.

Sie können eine gewöhnliche Fernsteuerung von irgendeinem Gerät verwenden, um IR-Interferenz zu erzeugen. TV, Videos, CD/DVD Spieler und Projektoren verwenden alle dieselben IR LEDs und IR Detektoren, die Sie gerade jetzt auch auf Ihrem Boe-Bot haben, um Signale an den IR-Detektor im TV, Videogerät etc. zu übermitteln.

Beispielprogramm: IrInterferenceSniffer.bs2

- √ Erfassen, speichern uns starten Sie IrInterferenceSniffer.bs2.
- √ Testen Sie das Programm um sicherzustellen, dass der Boe-Bot Alarm schlägt, wenn er IR-Interferenz entdeckt. Sie können das mit einem zweiten Boe-Bot tun, auf dem das Programm TestIrPairsAndIndicators.bs2 läuft. Wenn Sie

keinen zweiten Boe-Bot in der Nähe haben, verwenden Sie einfach eine normale Fernsteuerung eines TV, Video- oder CD/DVD-Spielers oder eines anderen Elektronikgerätes. Zielen Sie einfach mit der Fernsteuerung auf den Boe-Bot und drücken Sie irgendeine Taste. Wenn der Boe-Bot antwortet, indem der Alarm ertönt, wissen Sie, dass Ihr IR-Schnüffler funktioniert.

```
' Robotics with the Boe-Bot - IrInterferenceSniffer.bs2
' Test fluorescent lights, infrared remotes, and other sources
' of 38.5 kHz IR interference.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

counter        VAR     Nib

DEBUG "IR interference not detected, yet...", CR

DO
  IF (IN0 = 0) OR (IN9 = 0) THEN
    DEBUG "IR Interference detected!!!", CR
    FOR counter = 1 TO 5
      HIGH 1
      HIGH 10
      FREQOUT 4, 50, 4000
      LOW 1
      LOW 10
      PAUSE 20
    NEXT
  ENDIF
LOOP
```

Sie sind dran – Test für Fluoreszenzlicht, das Störungen bewirkt

√ Trennen Sie Ihren Boe-Bot vom seriellen Kabel und zielen Sie auf irgend eine Leuchtstoffröhre in der Nähe des Ortes, wo sie ihn laufen lassen wollen. Besonders wenn Sie häufig Alarm bekommen, sollten Sie die Leuchtstoffröhre abschalten, bevor Sie IR-Objekterkennung darunter verwenden.

Verwenden Sie immer den IrInterferenceSniffer.bs2 um sicherzustellen, dass jede Fläche, auf der Sie den Boe-Bot verwenden, frei von Infrarotstörsignalen ist. .

AKTIVITÄT 3: REICHWEITEN-ANPASSUNG DER INFRAROTERKENNUNG

Sie haben wohl schon bemerkt, dass hellere Scheinwerfer (oder eine hellere Taschenlampe) ermöglichen, im Dunkeln weiter entfernte Objekte zu sehen. Indem man die LED-Scheinwerfer des Boe-Bot heller macht, können Sie die Reichweite der Detektoren erhöhen. Ein kleinerer Widerstand lässt mehr Strom durch eine LED, weil er dem elektrischen Strom weniger Widerstand leistet. Mehr Stromfluss durch die LED bewirt, dass sie heller leuchtet. In der folgenden Aktivität werden Sie den Effekt verschiedener Widerstandswerte sowohl auf rote als auch auf Infrarot-LEDs untersuchen.

Bauteile:

Sie benötigen einige zusätzliche Bauteile für diese Aktivität:

(2) Widerstände – 470 Ω (gelb-violett-braun)
(2) Widerstände– 220 Ω (rot- rot - braun)
(1) Widerstand – 1 kΩ (braun -schwarz- rot)

Serienwiderstand und LED Helligkeit

Zunächst wollen wir eine der roten LEDs nehmen um den Unterschied zu “sehen”, den ein Widerstand auf die Leuchtkraft einer LED macht. Alles was wir benötigen ist ein Programm, das ein HIGH-Signal an die LED sendet.

Beispielprogramm: P1LedHigh.bs2

- √ Erfassen, speichern und starten Sie P1LedHigh.bs2.
- √ Überprüfen Sie, dass die LED im Schaltkreis an P1 aufleuchtet.

```
' Robotics with the Boe-Bot - P1LedHigh.bs2
' Set P1 high to test for LED brightness testing with each of
' these resistor values in turn: 220 ohm , 470 ohm, 1 k ohm.

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

DEBUG "Program Running!"

HIGH 1

STOP
```

Der Befehl **STOP** wird hier statt **END** verwendet, da **END** die BASIC Stamp in den Stromsparmodus schalten würde.

Sie sind dran – Test der LED Helligkeit

Denken Sie daran den Strom auszuschalten, bevor Sie Änderungen an einer Schaltung vornehmen. Denken Sie auch daran, dass das geladene Programm erneut startet, wenn Sie den Strom wieder einschalten. Sie können also dort weitermachen, wo Sie beim letzten Test aufgehört haben.

- √ Merken Sie sich, wie hell die LED in der Schaltung an P1 mit dem 220 Ω Widerstand leuchtet.
- √ Ersetzen Sie den 220 Ω Widerstand an P1 und der Kathode der rechten LED mit einem 470 Ω Widerstand.
- √ Halten Sie fest, wie hell die LED jetzt leuchtet.
- √ Wiederholen Sie das mit einem 1 kΩ Widerstand.
- √ Ersetzen Sie den 1 kΩ Widerstand wieder durch den 220 Ω Widerstand bevor Sie zum nächsten Teil dieser Aktivität gehen.
- √ Erklären Sie in eigenen Worten den Zusammenhang zwischen LED-Helligkeit und Serienwiderstand.

Serienwiderstand und IR-Reichweite

Wir wissen nun, dass ein kleinerer Serienwiderstand die LED heller leuchten lässt. Eine vernünftige Annahme wäre nun, dass eine hellere IR LED ermöglicht, Objekte zu erkennen, die weiter weg sind.

- √ Öffnen und starten Sie TestIrPairsAndIndicators.bs2 (von Seite 242).
- √ ¨Überprüfen Sie, dass beide Detektoren korrekt funktionieren.

Sie sind dran – Test der IR-LED Reichweite

- √ Messen Sie mit einem Metermass die längste Distanz zur IR LED, auf die ein auf die IR LED ausgerichtetes Blatt Papier detektiert werden kann.
- √ Ersetzen Sie die 1 kΩ Widerstände die P2 and P8 mit den Anoden der IR LED verbinden durch 470 Ω Widerstände.
- √ Bestimmen Sie die grösste Distanz auf die das gleiche Blatt Papier nun detektiert werden kann.
- √ Wiederholen Sie das Experiment mit den 220 Ω Widerständen.

√ Tragen Sie Ihre Daten in der Table 7-2 ein.

Table 7-2: Erkennungsreichweite vs. Widerstand	
IR LED Serien-Widerstand, [Ω]	**Maximale Erkennungsreichweite, Markieren Sie die benutzte Einheit: [in / cm]**
1000	
470	
220	

√ Bringen Sie Ihre IR-Paare wieder in die Originalkonfiguration (mit 1 kΩ Serienwiderständen bei beiden IR LED), bevor Sie fortfahren.
√ Kontrollieren Sie vor dem nächsten Schritt auch diese Änderung nochmals mit TestIrPairsAndIndicators.bs2 um sicherzugehen, dass beide IR LED/Detektor-Paare richtig funktionieren.

AKTIVITÄT 4: OBJEKTERKENNUNG UND -VERMEIDUNG 4:

Eine interessante Sache an den IR Detektoren ist, dass Ihre Outputs genau wie Fühler reagieren. Wenn kein Objekt erkannt wurde, ist der Output High. Wenn ein Objekt erkannt wurde, ist der Output Low. In diesem Schritt wird RoamingWithWhiskers.bs2 von Seite 179 so abgeändert, dass es mit den IR Detektoren funktioniert.

Umbau des Fühler-Progrmms zur IR Objekterkennung/Vermeidung

Dieses Beispielprogramm begann als RoamingWithWhiskers.bs2. Nebst der Anpassung von Name und Beschreibung wurden zwei Bit Variablen hinzugefügt, um den Status der IR-Detektoren zu speichern.

```
irDetectLeft  VAR     Bit
irDetectRight VAR     Bit
```

Ferner wurde eine Routine hinzugefügt, um die IR Paare zu lesen.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9
```

Die **IF…THEN** Statements worden so abgeändert, dass sie die Variablen abfragen, in denen die IR-Detektorergebnisse gespeichert sind, statt die Fühler Inputs.

```
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
```

Example Program – RoamingWithIr.bs2

- √ Öffnen Sie RoamingWithWhiskers.bs2
- √ Ändern Sie es so, dass es dem Programm unten entspricht.
- √ Schalten Sie den Strom an Ihrem Board und den Servos wieder ein.
- √ Speichern und starten Sie es.
- √ Überprüfen Sie, ob es sich – abgesehen von der Tatsache, dass kein körperlicher Kontakt nötig ist – genauso verhält wie RoamingWithWhiskers.bs2.

```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - RoamingWithIr.bs2
' Adapt RoamingWithWhiskers.bs2 for use with IR pairs.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

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

  irDetectLeft  VAR     Bit
  irDetectRight VAR     Bit
  pulseCount    VAR     Byte

' -----[ Initialization ]---------------------------------------------------

FREQOUT 4, 2000, 3000                       ' Signal program start/reset.

' -----[ Main Routine ]-----------------------------------------------------

DO

  FREQOUT 8, 1, 38500                       ' Store IR detection values in
```

```
  irDetectLeft = IN9                            ' bit variables.

  FREQOUT 2, 1, 38500
  irDetectRight = IN0

  IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    GOSUB Back_Up                               ' Both IR pairs detect obstacle
    GOSUB Turn Left                             ' Back up & U-turn (left twice)
    GOSUB Turn Left
  ELSEIF (irDetectLeft = 0) THEN                ' Left IR pair detects
    GOSUB Back Up                               ' Back up & turn right
    GOSUB Turn_Right
  ELSEIF (irDetectRight = 0) THEN               ' Right IR pair detects
    GOSUB Back Up                               ' Back up & turn left
    GOSUB Turn Left
  ELSE                                          ' Both IR pairs 1, no detects
    GOSUB Forward Pulse                         ' Apply a forward pulse
  ENDIF                                         ' and check again

LOOP

' -----[ Subroutines ]-----------------------------------------------------

Forward_Pulse:                                  ' Send a single forward pulse.
  PULSOUT 13,850
  PULSOUT 12,650
  PAUSE 20
  RETURN

Turn_Left:                                      ' Left turn, about 90-degrees.
  FOR pulseCount = 0 TO 20
    PULSOUT 13, 650
    PULSOUT 12, 650
    PAUSE 20
  NEXT
  RETURN

Turn Right:
  FOR pulseCount = 0 TO 20                      ' Right turn, about 90-degrees.
    PULSOUT 13, 850
    PULSOUT 12, 850
    PAUSE 20
  NEXT
  RETURN

Back Up:                                        ' Back up.
  FOR pulseCount = 0 TO 40
    PULSOUT 13, 650
    PULSOUT 12, 850
    PAUSE 20
  NEXT
```

```
  RETURN
```

Sie sind dran

√ Ändern Sie RoamingWithIr.bs2 so, dass die IR-Paare in einer Subroutine abgefragt werden.

AKTIVITÄT 5: HOCHLEISTUNGS-IR-NAVIGATION

Die Sorte vorprogrammierter Bewegungen, die in der letzten Aktivität benutzt wurden, sind gut für Fühler, aber unnötig langsam, wenn man IR-LEDs und Detektoren verwendet. Man kann die Leistungsfähigkeit des Boe-Bot deutlich erhöhen, indem auf Hindernisse geprüft wird vor jeder Pulsserie an die Servos. Das Programm kann die beste Bewegung für jeden einzelnen Moment der Navigation auswählen. So dreht der Boe-Bot nie weiter weg, als nötig und kann geschickt seinen Weg um die Hindernisse nehmen und auch anspruchsvollere Strecken erfolgreich meistern.

Zwischen allen Pulsen Abtasten um Kollisionen zu vermeiden

Das tolle am Entdecken eines Hindernisses, bevor man mit ihm zusammenstösst ist, dass dem Boe-Bot so noch etwas Platz bleibt, um darum herum zu navigieren. Der Boe-Bot kann einen Puls senden, um vom Objekt weg zu drehen, überprüfen, ob das Objekt immer noch im Weg ist, und einen weiteren Puls zur Vermeidung absetzen. Der Boe-Bot kann so lange checken und Pulse absetzen, bis der Weg frei ist. Dann kann er mit den Vorwärts-Pulsen weitermachen. Nachdem Sie mit dem folgenden Beispielprogramm etwas experimentiert haben werden Sie wahrscheinlich zustimmen, dass dies eine viel bessere Methode für den Boe-Bot ist, eine Strecke abzufahren.

Beispielprogramm: FastIrRoaming.bs2

√ Erfassen, speichern und starten Sie FastIrRoaming.bs2.

```
' Robotics with the Boe-Bot - FastIrRoaming.bs2
' Higher performance IR object detection assisted navigation

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

DEBUG "Program Running!"

irDetectLeft   VAR     Bit                   ' Variable Declarations
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
```

```
pulseRight     VAR     Word

FREQOUT 4, 2000, 3000                          ' Signal program start/reset.

DO                                             ' Main Routine

  FREQOUT 8, 1, 38500                          ' Check IR Detectors
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                               ' Decide how to navigate.
  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                         ' Apply the pulse.
  PULSOUT 12,pulseRight
  PAUSE 15

LOOP                                           ' Repeat main routine
```

Wie das Programm funktioniert

Dieses Programm wählt einen leicht anderen Ansatz für das Senden der Pulse. Nebst den zwei Bits zur Speicherung der IR Detektor Outputs benutzt es zwei Word Variablen um die Pulsdauer des **PULSOUT** Befehls zu setzen.

```
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word
```

Innerhalb des **DO**…**LOOP** werden die **FREQOUT** Befehle benutzt um ein 38.5 kHz IR Signal an beide IR LEDs zu senden. Unmittelbar nach dem 1 ms Burstsignal speichert eine Bit Variable den Output Status der IR Detektoren. Das ist notwendig, weil wenn Sie länger als die Dauer eines Befehls warten, werden die IR-Detektoren einen Status 1 (nichts erkannt) zurückmelden, unabhängig davon ob sie ein Objekt erkannt haben oder nicht.

```
FREQOUT 8, 1, 38500
irDetectLeft = IN9
FREQOUT 2, 1, 38500
irDetectRight = IN0
```

In den **IF…THEN** Statements setzt dieses Programm die Variablenwerte, die als ***Duration*** Argumente im **PULSOUT** Befehl verwendet werden, statt die Pulse zu erzeugen oder eine Navigationsroutine aufzurufen.

```
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
```

Bevor sich der **DO…LOOP** wiederholt, werden als letztes noch die Pulse für die Servos generiert. Beachten Sie dass der **PAUSE** Befehl nicht länger 20 ist. Statt dessen ist er 15, weil ungefähr 5 ms verbraucht wurden, um die IR LEDs abzufragen.

```
PULSOUT 13,pulseLeft                        ' Apply the pulse.
PULSOUT 12,pulseRight
PAUSE 15
```

Sie sind dran

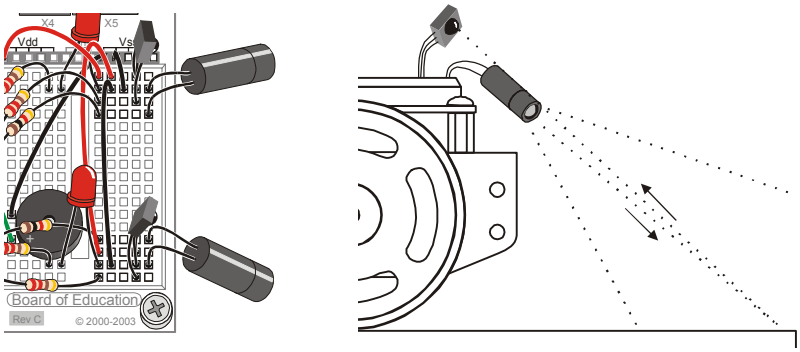
- √ Speichern Sie FastIrRoaming.bs2 als FastIrRoamingYourTurn.bs2.
- √ Ändern Sie das Programm so, dass der Piezolautsprecher verschiedene Töne zwischen den Pulsen generiert. Wählen Sie Töne für die vier verschiedenen Pulsbreiten-Kombinationen (Vorwärts, Rückwärts, Rechts und Links). Verkürzen Sie den **PAUSE** Befehl auf 7 ms, und nutzen Sie die anderen 7 ms für einen **FREQOUT** Befehl.
- √ Benutzen Sie die LEDs um anzuzeigen, dass der Boe-Bot ein Objekt erkannt hat.

√ Versuchen Sie die Werte, auf die pulseLeft und pulseRight gesetzt sind so zu ändern, dass der Boe-Bot alles mit halber Geschwindigkeit macht.

AKTIVITÄT 6: DER ABSTURZKANTEN-DETEKTOR

Bis jetzt wurde der Boe-Bot hauptsächlich darauf programmiert, Ausweichmanöver einzuleiten, wenn ein Objekt erkannt wurde. Es gibt auch Situationen, in denen der Boe-Bot Ausweichmanöver einleiten muss, wenn kein Objekt erkannt wird. Wenn der Boe-Bot zum Beispiel auf einem Tisch fährt, könnten die IR-Detektoren nach unten auf die Tischoberfläche schauen wie in Figur 7-8. Das Programm sollte den Boe-Bot vorwärts fahren lassen, solange beide IR-Detektoren eine Tischoberfläche "sehen" können.

√ Schalten Sie den Strom an Ihrem Board und den Servos ab.
√ Zielen Sie mit Ihren IR-Paaren nach unten und aussen, wie in Figur 7-8 gezeigt.



Draufsicht Seitenansicht

Figur 7-8
IR Paare zeigen abwärts um die Absturzkante zu erkennen

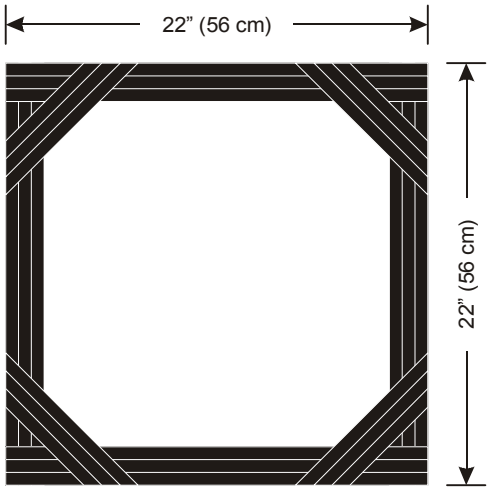
Empfohlenes Material:

(1) Rolle schwarzes Isolierband – 19 mm (¾") breit.
(1) Stück weissen Plakatkarton – 56 X 71 cm (22 X 28 in) empfohlen.

Simulation einer Absturzkante mit Isolierband

Ein Stück weissen Plakatkartons mit einem Rand aus Isolierband gibt eine praktische Lösung um die Absturzkante einer Tischkante zu simulieren – mit viel weniger Risiko für Ihren Boe-Bot.

- √ Kleben Sie ein Spielfeld ähnlich dem in Figur 7-9 gezeigten, mit Isolierband hergestellten Spielfeld her. Verlegen Sie mindestens drei Streifen Isolierband nahtlos von Rand zu Rand, ohne sichtbaren Karton zwischen den Streifen..
- √ Kontrollieren Sie, dass Sie 1 kΩ Widerstände (braun-schwarz-rot) verwenden, um P2 und P8 mit ihren IR LED zu verbinden. Wir möchten, dass der Boe-Bot für diese Aktivität kurzsichtig ist.
- √ Schalten Sie den Strom an Ihrem Board ein.
- √ Benutzen Sie TestIrPairsAndIndicators.bs2 (Seite 242) um zu überprüfen, dass der Boe-Bot die Kartonfläche, nicht aber das Isolierband erkennt.
- √ Starten Sie IrInterferenceSniffer.bs2 (Seite 244) um sicherzustellen, dass keine Fluoreszenzröhre die IR Detektoren Ihres Boe-Bot stören.



Figur 7-9
Isolierband simuliert Tischkanten

Wenn Sie es auf einem Tisch ausprobieren, nachdem Sie mit dem Isolierband-Spielfeld Erfolg hatten:

Halten Sie sich bereit, solange Ihr Boe-Bot den Tisch erkundet:

- √ Sorgen Sie dafür, dass Sie der Beschützer Ihres Boe-Bot sind. Seien Sie jederzeit bereit, den Boe-Bot von oben zu packen, wenn er sich beim Navigieren der Tischkante nähert. Wenn der Boe-Bot versucht, über die Tischkante zu fahren, heben Sie ihn auf, bevor er stürzt. Sonst könnte aus Ihrem Boe-Bot ein Not-Bot werden!

Wenn Sie Ihren Boe-Bot beschützen während er den Absturzkanten ausweicht, halten Sie sich bereit, ihn von oben anzuheben. Andernfalls könnte der Boe-Bot Ihre Hände sehen, statt der Absturzkante und sich nicht wie erwartet verhalten. Der Boe-Bot könnte auch Sie selbst erkennen, wenn Sie in seiner Sichtlinie stehen. Versuchen Sie also ausserhalb seines Sichtfeldes zu bleiben, während Sie ihn beschützen.

Führen Sie immer folgende Instruktionen aus, bevor Sie das Programm starten:

- √ Kontrollieren Sie, dass Sie 1 kΩ Widerstände (braun-schwarz-rot) um P2 und P8 mit ihren IR LED zu verbinden. Wir wollen einen kurzsichtigen Boe-Bot für diese Aktivität.
- √ Benutzen Sie TestIrPairsAndIndicators.bs2 (Seite 242) um sicherzustellen, dass der Boe-Bot den Karton, aber nicht das Isolierband erkennt.
- √ Benutzen Sie IrInterferenceSniffer.bs2 (Seite 244) um sicherzustellen, dass keine Leuchtstoffröhre die IR-Detektoren Ihres Boe-Bot stören..

Programmierung der Absturz-Erkennung

Im wesentlichen bedeutet die Programmierung Ihres Boe-Bot, so dass er auf einem Tisch herumfährt ohne über den Rand zu fallen, lediglich die Anpassung der **`IF...THEN`** Statements von FastIrNavigation.bs2. Die primäre Anpassung liegt darin, dass die Servos den Boe-Bot vorwärts rollen sollten, wenn **`irDetectLeft`** und **`irDetectRight`** beide 0 sind, was anzeigt, dass ein Objekt (die Tischfläche) erkannt wurde. Der Boe-Bot muss sich auch von einem Detektor abwenden, der anzeigt, dass er kein Objekt erkannt hat. Wenn also **`irDetectLeft`** 1 ist, wäre es gesünder für den Boe-Bot, nach rechts zu drehen.

Eine zweite Eigenschaft des Programms zur Absturzvermeidung ist die anpassbare Distanz. Sie möchten wohl, dass Ihr Boe-Bot nur einen Vorwärtspuls nimmt zwischen der Kontrolle der Detektoren, aber sobald eine Absturzkante entdeckt wurde, sollte der Boe-Bot vielleicht besser eine mehrere Pulse lange Drehung vornehmen, bevor er die Detektoren wieder abfragt.

Allerdings müssen Sie nicht gleich in die Navigation im Fühler-Stil zurückfallen, bloss weil Sie mehrere Pulse in einem Ausweichmanöver nehmen. Statt dessen können Sie eine **pulseCount** Variable hinzufügen, die Sie verwenden um die Anzahl Pulse für ein Manöver festzulegen. Der **PULSOUT** Befehl kann in eine **FOR…NEXT** Schleife eingebaut werden, die **FOR 1 TO pulseCount** Pulse abgibt. Für einen Vorwärtspuls kann **pulseCount** 1 sein, für zehn Linkspulse kann **pulseCount** auf 10 gesetzt werden und so weiter.

Beispielprogramm: AvoidTableEdge.bs2

- √ Öffnen Sie FastIrNavigation.bs2 und speichern Sie es als AvoidTableEdge.bs2.
- √ Ändern Sie das Programm so, dass es dem Beispiel entspricht. Das umfasst das Hinzufügen von Variablen, Ändern der **IF…THEN** Statements, und das Verschachteln der **PULSOUT** Befehle in eine **FOR…NEXT** Schleife. Seien Sie vorsichtig, dass alle **pulseLeft** und **pulseRight** Variablenwerte im **IF…THEN** Statement richtig angepasst sind. Ihre Werte unterscheiden sich von denen in FastIrNavigation.bs2 weil die regeln der Strecke verschieden sind.
- √ Schalten Sie Ihr Board und die Servos wieder ein.
- √ Testen Sie das Programm auf ihrem Isolierbandbegrenzten Spielfeld.
- √ Wenn Sie es auf einem Tisch ausprobieren wollen, denken Sie daran die Test und Rettungstipps zu befolgen, die wir oben diskutiert haben.

```
' Robotics with the Boe-Bot - AvoidTableEdge.bs2
' IR detects object edge and navigates to avoid drop-off.

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

DEBUG "Program Running!"

irDetectLeft   VAR     Bit                  ' Variable declarations.
irDetectRight  VAR     Bit
pulseLeft      VAR     Word
pulseRight     VAR     Word
loopCount      VAR     Byte
pulseCount     VAR     Byte

FREQOUT 4, 2000, 3000                       ' Signal program start/reset.

DO                                          ' Main Routine.

  FREQOUT 8, 1, 38500                       ' Check IR detectors.
```

```
  irDetectLeft = IN9
  FREQOUT 2, 1, 38500
  irDetectRight = IN0
                                          ' Decide navigation.

  IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    pulseCount = 1                        ' Both detected,
    pulseLeft = 850                       ' one pulse forward.
    pulseRight =  650
  ELSEIF (irDetectRight = 1) THEN         ' Right not detected,
    pulseCount = 10                       ' 10 pulses left.
    pulseLeft = 650
    pulseRight =  650
  ELSEIF (irDetectLeft = 1) THEN          ' Left not detected,
    pulseCount = 10                       ' 10 pulses right.
    pulseLeft = 850
    pulseRight =  850
  ELSE                                    ' Neither detected,
    pulseCount = 15                       ' back up and try again.
    pulseLeft = 650
    pulseRight =  850
  ENDIF

  FOR loopCount = 1 TO pulseCount         ' Send pulseCount pulses
    PULSOUT 13,pulseLeft
    PULSOUT 12,pulseRight
    PAUSE 20
  NEXT

LOOP
```

Wie das Programm funktioniert

Da das Programm eine modifizierte Version von FastIrRoaming.bs2 ist werden hier nur die Programmänderungen diskutiert.

Eine **FOR**…**NEXT** Schleife wird dem Programm hinzugefügt um die Anzahl generierter Impulse zu steuern, die in jedem Schleifendurchlauf der Hauptroutine (**DO**…**LOOP**) abgesetzt werden. Ferner werden zwei Variablen hinzugefügt, **loopCount** als Index für eine **FOR**…**NEXT** Schleife und **pulseCount** als das ***EndValue*** Argument.

```
      loopCount      VAR     Byte
      pulseCount     VAR     Byte
```

Die **IF…THEN** Statements bestimmen nun die Werte sowohl von **pulseCount** als auch von **pulseRight** und **pulseLeft**. Wenn beide Detektoren den Tisch sehen können, mache einen vorsichtigen Impuls vorwärts.

```
IF (irDetectLeft = 0) AND (irDetectRight = 0) THEN
    pulseCount = 1
    pulseLeft = 850
    pulseRight =  650
```

Andernfalls, wenn der rechte IR Detektor keinen Tisch sieht, drehe 10 Pulse links.

```
ELSEIF (irDetectRight = 1) THEN
  pulseCount = 10
  pulseLeft = 650
  pulseRight =  650
```


Andernfalls, wenn der linke IR Detektor keinen Tisch sieht, drehe 10 Pulse rechts.

```
ELSEIF (irDetectLeft = 1) THEN
  pulseCount = 10
  pulseLeft = 850
  pulseRight =  850
```

Andernfalls, wenn keiner der Detektoren den Tisch sehen kann, fahre 15 Pulse retour und hoffe, dass einer der Detektoren die Absturzkante etwas vor dem anderen sehen wird.

```
ELSE
  pulseCount = 15
  pulseLeft = 650
  pulseRight =  850
ENDIF
```

Nachdem nun die Werte von **pulseCount**, **pulseLeft**, und **pulseRight** gesetzt sind, liefert diese **FOR…NEXT** Schleife die spezifizierte Anzahl Pulse für das von den **pulseLeft** und **pulseRight** Variablen bestimmte Fahrmanöver.

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

Sie sind dran

Sie können mit verschiedenen Werten für **`pulseLeft`**, **`pulseRight`**, und **`pulseCount`** im **`IF`**…**`THEN`** Statement experimentieren. Wenn zum Beispiel der Boe-Bot nicht zu weit dreht, kann er möglicherweise den Rand des Isolierband-Spielfeldes abfahren. Rückwärtswende um eines der Räder statt drehen an Ort kann ebenfalls zu interessantem Verhalten führen.

- √ Ändern Sie AvoidTableEdge.bs2 so dass er dem Rand des Isolierband-Spielfeldes folgt, indem Sie die **`pulseCount`** Werte so anpassen, dass der Boe-Bot nicht zu weit von der Kante weg dreht.
- √ Experimentieren Sie mit einer Rückwärtswende als Variante um Ihren Boe-Bot innerhalb des Perimeters herumfahren zu lassen, statt dass er der Kante folgt.

ZUSAMMENFASSUNG

Dieses Kapitel befasste sich mit einer speziellen Technik zur Infrarot-Objekterkennung, welche die Infrarot-LED verwendet, wie sie in handelsüblichen Fernsteuerungen vorkommen und einen Infrarot-Detektor, wie er in den meisten TV-Geräten, CD/DVD-Spielern und anderen Apparaten vorkommen, die mit solchen Fernsteuerungen bedient werden. Eine Infrarot-LED Schaltung wurde verwendet, um ein 38.5 kHz Signal auszustrahlen. Wir verwendeten dazu eine Eigenheit des **`FREQOUT`** Befehls genannt *Harmonische*, welche bei allen digital synthetisierten Signalen vorkommt. Indem man die Spur des Boe-Bot mit Infrarotlicht beleuchtet und dann seine Reflexionen aufspürt, kann man Objekterkennung erreichen, ohne das Objekt physisch berühren zu müssen.

Ein Programm zur Infraroterkennung wurde entwickelt, um die IR LED/Detektorenpaare offline (nicht am PC angeschlossen) testen zu können. Ein Schnüffelprogramm für Infrarot-Interferenzen wurde ebenfalls entwickelt, um infrarote Lichteinstreuungen durch gewisse Fluoreszenzröhren in Deckenbeleuchtungen feststellen zu können. Da die Signale aus den IR Detektoren denen von Fühlern sehr ähnlich sind, haben wir RoamingWithWhiskers.bs2 auf die Infrarotdetektoren angepasst. Ein Programm, das zwischen jedem Servo-Puls die IR Detektoren abfragt, wurde vorgestellt, um die höhere Leistungsfähigkeit bei der Fahrt mit Vermeidung von Hindernissen zu demonstrieren. Dieses Programm wurde dann abgeändert, um die Kante einer durch Isolierband abgegrenzten Fläche zu vermeiden. Da dunkles Isolierband Infrarotstrahlung absorbiert kann durch das Umrahmen eines grossen Zeichenpapiers mit Isolierband eine Tischkante simuliert werden, ohne dass die Gefahr besteht, dass unser Boe-Bot tatsächlich vom Tisch fällt.

Fragen

1. Was bedeutet Infrarot? Wie unterscheidet sich Infrarot von nahem Infrarot?
2. Welches sind die zwei Sorten Filter, die in den hier verwendeten IR Detektoren eingebaut sind? Was bewirken sie?
3. Welche Frequenzen haben die Harmonischen, die durch **`FREQOUT 2, 1, 38500`** erzeugt werden? Welche Frequenz hat die Grundschwingung die mit diesem Befehl ausgestrahlt wird? Für wie lange werden diese Signale gesendet? An welchen I/O Pin muss die IR LED Schaltung angeschlossen werden, damit das Signal ausgestrahlt wird?

4. Welcher Befehl muss unmittelbar auf den **FREQOUT** Befehl folgen, damit man genau entscheiden kann, ob oder ob nicht ein Objekt erkannt wurde?
5. Was bedeutet es, wenn der IR Detektor ein Low Signal sendet? Was bedeutet ein High Signal?
6. Was passiert, wenn Sie den Wert eines Serienwiderstandes in einem LED Schaltkreis ändern? Was passiert, wenn Sie den Wert eines Serienwiderstandes in einem Infrarot-Schaltkreis ändern?
7. Inwiefern ähnelt RoamingWithIr.bs2 RoamingWithWhiskers.bs2? Inwiefern unterscheiden sie sich?
8. Inwiefern unterscheiden sich RoamingWithIr.bs2 und FastIrRoaming.bs2? Wie lassen sie sich mit AvoidTableEdge.bs2 vergleichen?

Übungen

1. Welchen Befehl verwenden Sie, um mit dem IR LED Schaltkreis des Boe-Bot eine 39 kHz Harmonische Infrarotwelle auszustrahlen? Schreiben Sie den Sendebefehl für das 39 kHz Signal auf.
2. Modifizieren Sie eine Codezeile in IrInterferenceSniffer.bs2 so, dass es nur noch eines der IR LED/Detektor Paare überwacht.
3. Schreiben Sie die "Apply the pulse." Routine aus FastIrRoaming.bs2 so um, dass sie drei statt einen Puls erzeugt. Erläutern Sie auch allfällige Befehle, die früher im Programm benötigt werden könnten, damit das Programm seine Aufgabe erfüllen kann.
4. Erläutern Sie die Funktion von **pulseCount** in AvoidTableEdge.bs2. Was hat dies mit Ihrer Antwort auf Übung 3 zu tun?

Projekte

1. Entwickeln Sie einen Boe-Bot der stillsitzt, bis Sie mit Ihrer Hand vor ihm winken, und dann beginnt, seine Umgebung zu erkunden.
2. Entwickeln Sie einen Boe-Bot der sich langsam am Ort dreht, bis er ein Objekt erkennt. Sobald er ein Objekt erkennt, soll er sich an dieses heften und es verfolgen. Das ist ein klassisches SumoBot Verhalten.
3. Entwickeln Sie einen Boe-Bot der herumfährt, bis er Infrarot-Interferenz feststellt, dann einen kurzen Alarmton ausstösst, und dann weiterfährt. Der Warnton sollte sich vom Batterie-Leer-Alarm unterscheiden.
4. Montieren Sie Fühler an Ihren Infrarot - Boe-Bot. Umwickeln Sie alle Teile der Fühler, die unter Umständen die IR LED oder den Detektor berühren könnten, mit Isolierband. Erstellen Sie ein Labyrinth mit mehreren Räumen aus

Isolierband. Platzieren Sie ein Objekt in die Mitte eines der Räume. Das Ziel des Boe-Bot ist, sich innerhalb der mit Isolierband abgegrenzten Piste zu bewegen, bis er mit einem Fühler das Hindernis berührt.

5. Entfernen Sie die Fühler und bauen Sie Lichtdetektoren ein. Ändern Sie Ihr Labyrinth so, dass eine Taschenlampe oder eine Tischlampe eine bestimmte Fläche beleuchtet. Das neue Ziel des Boe-Bot ist es nun, sich innerhalb des Labyrinths zu bewegen, bis er die von der Lampe erleuchtete Fläche findet.

Kapitel 8: Robotersteuerung mit Distanzerkennung

In Kapitel 7 benutzten wir Infrarot-Sensoren zur Erkennung, ob ein Objekt dem Boe-Bot im Weg ist, ohne dieses wirklich berühren zu müssen. Wäre es nicht schön zu wissen, wie weit das Objekt entfernt ist? Normalerweise ist das eine Aufgabe für Sonar, der einen Tonpuls emittiert und misst, wie lange das Echo braucht, um wieder zurückzukommen. Aus der Antwortzeit kann man dann die Distanz des Objektes berechnen. Es gibt aber auch eine Möglichkeit, die Distanzmessung mit genau dem Schaltkreis aus dem letzten Kapitel vorzunehmen. Wenn Ihr Boe-Bot in der Lage ist, den Abstand zu einem Objekt festzustellen, kann man ihn programmieren, dass er einem sich bewegenden Objekt folgt, ohne mit diesem Zusammenzustossen. Der Boe-Bot kann auch darauf programmiert werden, einer schwarzen Linie auf einem weissen Grund zu folgen.

DISTANZBESTIMMUNG MIT DER IR-LED/DETEKTOR SCHALTUNG

Wir verwenden den Schaltkreis aus dem vorhergehenden Kapitel um die Distanz zu messen.

- √ Wenn die Schaltung immer noch auf Ihrem Boe-Bot aufgebaut ist, lassen Sie ihn so.
- √ Wenn Sie die Schaltung des vorangehenden Kapitels bereits zerlegt haben, wiederholen Sie die Schritte in Kapitel 7, Aktivität 1, (Aktivität 1) auf Seite 235.

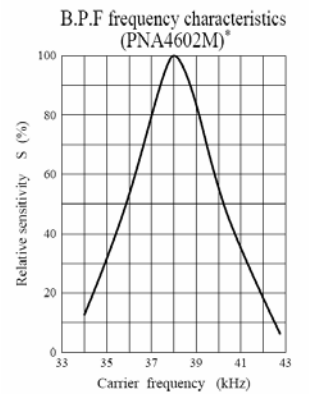
Empfohlene Ausrüstung und Materialien:

(1) Massstab, Klappmeter oder Messband
(1) Blatt Papier

AKTIVITÄT 1: TEST DES FREQUENZDURCHLAUFS (*SWEEP*)

Figur 8-1 zeigt einen Ausschnitt aus dem Datenblatt Ihres IR-Detektors (Panasonic PNA4602M). Die Kurve zeigt, um wie viel weniger empfindlich der IR-Detektor wird, wenn das Empfangssignal mit einer anderen Frequenz als 38.5 kHz oszilliert. Wenn Sie ihm z.B. ein IR mit 40 kHz senden, ist es nur 50% so empfindlich, wie er bei 38.5 kHz wäre. Wenn das IR mit 42 kHz schwingt, sinkt die Empfindlichkeit des Empfängers auf 20%. Speziell für Frequenzen, auf denen der Empfänger weniger empfindlich ist, muss

das Objekt näher stehen um das IR heller zu reflektieren, damit der Detektor es erkennen kann.



Figur 8-1
Die Filter Empfindlichkeit (*sensitivity*) hängt von der Träger-Frequenz (*carrier frequency*) ab.

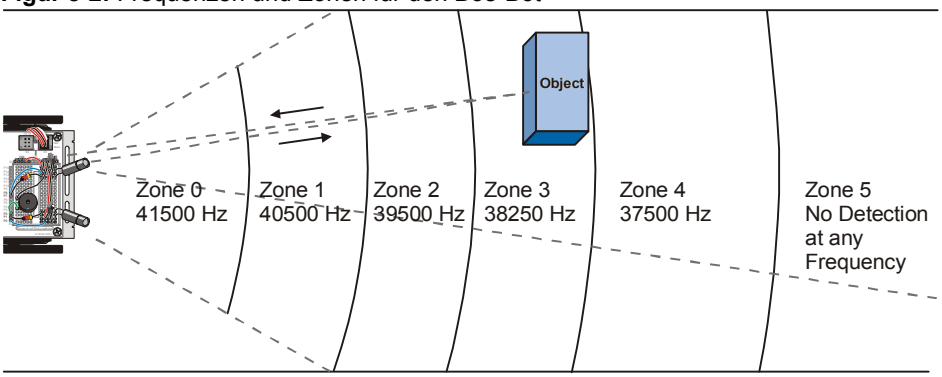
Man kann das auch so sehen, dass der Detektor auf der empfindlichste Frequenz die am weitesten entfernten Objekte sieht, während die weniger empfindlichen Frequenzen nur für näher liegende Objekte benutzt werden können. Damit wird die Distanzerkennung einfach. Nehmen Sie 5 Frequenzen und testen Sie diese von der empfindlichsten zur wenigst empfindlichen durch. Beginnen Sie mit der empfindlichsten Frequenz. Wenn ein Objekt detektiert wurde, überprüfen Sie, ob es mit der nächst weniger empfindlichen Frequenz auch noch sichtbar ist, und so weiter. Aus der Frequenz, bei welcher das Objekt nicht mehr sichtbar ist, können Sie leicht auf die Distanz des Objektes schliessen.

Frequenzdurchlauf (*Frequency Sweep*) ist die Technik, den Output eines Schaltkreises zu testen, indem man eine Anzahl verschiedener Input-Frequenzen verwendet..

Programmierung des Frequenz-Sweep für Distanzerkennung

Figur 8-2 zeigt ein Beispiel, wie der Boe-Bot mit Hilfe der Frequenzen die Distanz ausloten kann. In diesem Beispiel ist das Objekt in Zone 3. Das bedeutet, dass das Objekt auf den Frequenzen 37500 Hz und 38250 Hz erkannt werden kann, aber nicht mit den Frequenzen 39500 Hz, 40500 Hz, and 41500 Hz. Wenn Sie das Objekt in die Zone 2 stellen würden, könnte es mit den Frequenzen 37500 Hz, 38250 Hz, and 39500 Hz, nicht aber mit 40500 Hz und 41500 Hz detektiert werden.

Figur 8-2: Frequenzen und Zonen für den Boe-Bot



Sie fragen sich vielleicht, warum der Wert von Zone 4 37.5 kHz statt 38.5 kHz ist. Der Grund dafür, dass es nicht die Frequenzen sind, die Sie aufgrund des %-Empfindlichkeits-Graphen erwarten würden, liegt darin, dass der FREQOUT Befehl ein etwas stärkeres (harmonisches) Signal auf 37.5 kHz sendet, als auf 38.5 kHz. Die Frequenzen in Figur 8-2 sind die optimalen Frequenzen, die sie der BASIC Stamp einprogrammieren, um die Distanz eines Objektes zu erkennen. Diese Frequenzen wurde aufgrund von Tests ermittelt, die im Anhang beschrieben sind.

Damit Sie den IR Detektor auf jeder der Frequenzen abfragen können, müssen Sie mit **`FREQOUT`** die fünf verschiedenen Frequenzen senden und bei jeder testen, ob der IR-Detektor ein Objekt erkennen konnte. Die Schritte zwischen den Frequenzen sind nicht gleichmässig genug für den Einsatz des **`STEP`** Operators einer **`FOR`**…**`NEXT`** Schleife. Sie könnten **`DATA`** und **`READ`** verwenden, aber das wäre etwas mühsam. Sie können fünf verschiedene **`FREQOUT`** Befehle nehmen, aber das wäre eine Verschwendung von Programmspeicherplatz. Der beste Ansatz zur Speicherung einer kurzen Werteliste, die Sie hintereinander verwenden wollen, ist der Befehl **`LOOKUP`**. Die Syntax für den **`LOOKUP`** Befehl ist:

`LOOKUP` ***Index, [Value0, Value1, …ValueN], Variable***

Wenn das ***`Index`*** Argument 0 ist, wird ***`Value0`*** aus der Liste in den eckigen Klammern in die ***`Variable`*** eingesetzt. Wenn ***`Index`*** 2 ist, wird ***`Value1`*** aus der Liste in die ***`Variable`*** eingesetzt. Man kann bis zu 256 Werte auflisten, aber für das nächste Beispielprogramm genügen 5. Hier ist ein Beispiel, wie man das benutzt:

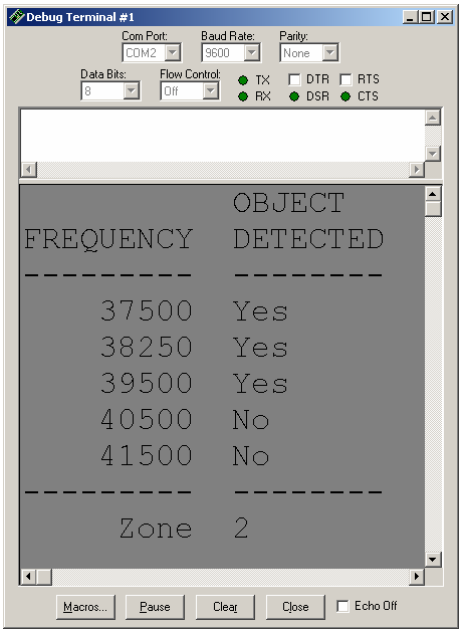
```
FOR freqSelect = 0 TO 4
```

```
    LOOKUP freqSelect,[37500,38250,39500,40500,41500],irFrequency
    FREQOUT 8,1, irFrequency
    irDetect = IN9
      ' Commands not shown...

NEXT
```

Beim ersten Durchlauf der **FOR**…**NEXT** Schleife ist **freqSelect** 0, also setzt der **LOOKUP** Befehl den wert 37500 in die **irFrequency** Variable. Da **irFrequency** nach dem **LOOKUP** Befehl 37500 enthält, sendet der **FREQOUT** Befehl diese Frequenz an die IR LED an P8. Wie im vorangehenden Kapitel wird der Wert von **IN9** dann in der Variablen **irDetect** gespeichert. Beim zweiten Durchlauf der **FOR**…**NEXT** Schleife ist der Wert von **freqSelect** nun 1, was bedeutet, dass der **LOOKUP** Befehl 38250 in die Variable **irFrequency** setzt, und der Sende/Empfangsprozess wird für diese höhere Frequenz wiederholt. Beim dritten Durchlauf wird alles mit 39500 wiederholt, und so weiter. Das Ergebnis ist bemerkenswert, besonders wenn Sie berücksichtigen, dass Sie Bauteile verwenden, die für einen völlig anderen Zweck entworfen wurden, nämlich um IR-Kommunikation zwischen einer Fernsteuerung und einem TV-Gerät zu ermöglichen.

Beispielprogramm: TestLeftFrequencySweep.bs2

TestLeftFrequencySweep.bs2 macht zwei Dinge. Zunächst testet es das IR LED/Detektorpaar (an P8 und P9) um sicherzugehen, dass sie für die Distanzerkennung richtig funktionieren. Darüber hinaus zeigt es auch, wie der Frequenz-Sweep aus Figur 8-2 erreicht wird. Wenn Sie das Programm starten, sollte die Anzeige etwa so aussehen wie in Figur 8-3.



Figur 8-3
Test der Distanz-Erkennung

Denken Sie daran, dass diese Distanzmessungen relativ sind, und nicht notwendigerweise genau oder in gleichen Abständen. Trotzdem geben diese Messungen dem Boe-Bot genügend Eindruck der Objektdistanz um Linien oder Objekten nachfahren zu können.

- √ Erfassen, speichern und starten Sie TestLeftFrequencySweep.bs2.
- √ Benutzen Sie ein Blatt Papier oder einen Karton um die Distanzerkennung von IR LED/Detektor zu testen.
- √ Starten Sie mit dem Blatt sehr nahe an der IR-LED, etwa 1 cm (¼ in) von der IR LED entfernt. Die Zone in Ihrem Debug Terminal sollte nun 0 oder 1 sein.
- √ Verschieben Sie das Papier allmählich weiter von der IR LED weg, und notieren Sie sich jede Abstandsmessung, bei welcher die Zone eine Stufe höher geht.

Die Zonen 1-4 liegen typischerweise im Bereich von 15 bis 30 cm (6 bis 12 in) für die abgeschirmten LEDs mit 1 kΩ Widerstand. Für ältere, in Schrumpfschlauch gepackte LED werden die Distanzen geringer sein. So lange also die Objekte bis etwa 10 cm erkannt werden können, funktionieren die Experimente in diesem Kapitel. Wenn der Distanzbereich der Abstandserkennung geringer ist, was mit geschrumpften LEDs zu erwarten ist, sollten Sie versuchen, die Serienwiderstände von 1 kΩ auf 470 Ω oder 220 Ω zu reduzieren.

```
' -----[ Title ]-----------------------------------------------------------
' Robotics with the Boe-Bot - TestLeftFrequencySweep.bs2
' Test IR detector distance responses to frequency sweep.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

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

freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetect       VAR     Bit
distance       VAR     Nib

' -----[ Initialization ]--------------------------------------------------

DEBUG CLS,
      "           OBJECT", CR,
      "FREQUENCY  DETECTED", CR,
      "---------  --------"

' -----[ Main Routine ]----------------------------------------------------

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,
        "Zone       ", DEC1 distance
LOOP
```

Sie sind dran – Test des Rechten IR LED/Detektor Paars

Mit der Änderung von nur zwei Zeilen können Sie dieses Programm für den Test der rechten IR LED/Detektor Kombination. Denken Sie aber daran, auch die Bezeichnungen im Programm richtig anzupassen. Ändern Sie also die folgenden Zeilen:

```
FREQOUT 8,1, irFrequency
irDetect = IN9
```

so ab, dass sie wie folgt aussehen:

```
FREQOUT 2,1, irFrequency
irDetect = IN0
```


- √ Modifizieren Sie TestLeftFrequencySweep.bs2 für den Test und die Messung des rechten IR LED/Detektorpaars ab.
- √ Starten Sie das Programm und überprüfen Sie, dass diese Kombination ähnliche Distanzen messen kann.

Anzeige beider Distanzen

Ein kleines Programm, das beide Distanzmesser des Boe-Bot gleichzeitig messen kann, ist oft nützlich. Dieses Programm ist in Subroutinen gegliedert, die sich auch gut eignen, um mit copy/paste in andere Programme übernommen zu werden, wo Sie Abstandsmessungen benötigen.

Beispielprogramm: DisplayBothDistances.bs2

- √ Erfassen, speichern und starten Sie DisplayBothDistances.bs2.
- √ Wiederholen Sie die Distanzmessungsübung mit einem Blatt Papier für jede LED einzeln, dann mit beiden LEDs gleichzeitig.

```
' -----[ Title ]-----------------------------------------------------------------
' Robotics with the Boe-Bot - DisplayBothDistances.bs2
' Test IR detector distance responses of both IR LED/detector pairs to
' frequency sweep.

' {$STAMP BS2}                              ' Stamp directive.
```

```
' {$PBASIC 2.5}                              ' PBASIC directive.

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

freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
distanceLeft   VAR     Nib
distanceRight  VAR     Nib

' -----[ Initialization ]---------------------------------------------------

DEBUG CLS,
      "IR OBJECT ZONE", CR,
      "Left   Right", CR,
      "-----  -----"

' -----[ Main Routine ]-----------------------------------------------------

DO

  GOSUB Get Distances
  GOSUB Display_Distances

LOOP

' -----[ Subroutine - 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
```

Sie sind dran – Mehr Distanzmessungen

√ Versuchen Sie, die Distanz verschiedener Objekte zu messen und prüfen Sie, ob die Farbe und/oder die Oberflächenbeschaffenheit (Textur) einen Einfluss auf die Abstandsmessung haben.

AKTIVITÄT 2: BOE-BOT SCHATTENFAHRZEUG

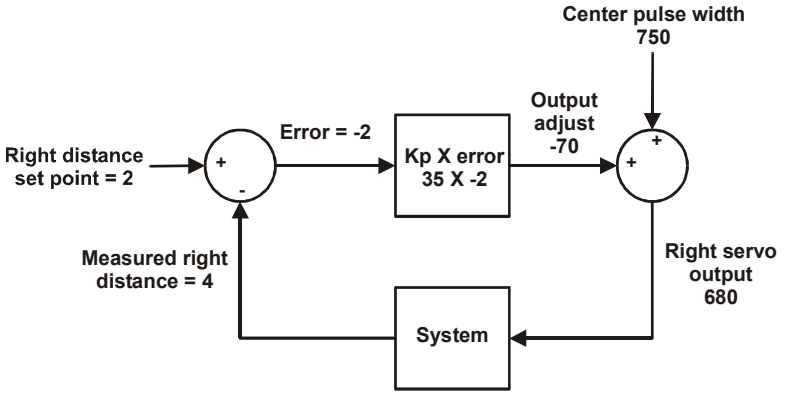
Wenn ein Boe-Bot dem anderen nachfährt, muss der hinterherfahrende Boe-Bot, auch Schattenfahrzeug (*shadow vehicle*)genannt, wissen, wie weit voraus des Führungsfahrzeug (*lead vehicle*) ist. Wenn das Schattenfahrzeug zurückfällt, muss es das erkennen und beschleunigen. Wenn das Schattenfahrzeug zu nahe am Führungsfahrzeug ist, muss es das ebenfalls erkennen und abbremsen. Wenn der Abstand stimmt, kann es weiterfahren bis die Messungen wieder einen zu kleinen oder zu grossen Abstand anzeigen.

Distanzen sind nur eine Art von Werten, für welche Roboter und andere automatischen Maschinen verantwortlich sind. Wenn eine Maschine gebaut wurde, um einen bestimmten Wert einzuhalten, wie etwa ein Abstand, ein Druck oder ein Füllstand, ist normalerweise ein Steuerungssystem (*control system*) beteiligt. Solche Systeme bestehen manchmal aus Sensoren und Ventilen, oder Sensoren und Motoren, oder, wie im Falle des Boe-Bot, aus Sensoren und Freilaufservos. Ausserdem gibt es eine Art Prozessor, der die Messungen der Sensoren übernimmt und diese in eine mechanische Bewegung umsetzt. Der Prozessor muss so programmiert sein, dass er Entscheidungen auf Basis der Sensorinputs trifft, und dann den mechanischen Output entsprechend ansteuert. Im Falle des Boe-Bot ist dieser Prozessor die BASIC Stamp 2.

Der Regelkreis (*closed loop control*) ist eine gängige Methode, um ein Niveau aufrechtzuerhalten, und sie hilft auch dem Boe-Bot, seinen Abstand von einem Objekt aufrechtzuerhalten. Es gibt viele verschiedene Sorten von Regelkreisen. Einige der verbreitetsten sind die Hysteresesteuerung (*hysteresis control*), Proportionalsteuerung (*proportional control*, Integralsteuerung (*integral control*), und die Differenzregelung

(*derivative control*). Alle diese verschiedenen Formen von Steuerungen werden detailliert in dem (englischen) Lehrtext des Stamps in Class-Programms *Industrial Control*, diskutiert, das im Vorwort erwähnt wurde.

Die meisten Steuerungstechniken können mit einigen wenigen Zeilen PBASIC Code implementiert werden. Tatsächlich reduziert sich der grösste Teil des Proportional-Regelkreises in Figur 8-4 auf gerade mal eine einzige Zeile PBASIC Programmcode. Dieses Diagramm nennt man Blockdiagramm (*block diagram*). Es beschreibt die Schritte des Proportionalsteuerprozesses, der vom Boe-Bot verwendet wird um mit seinem rechten IR LED/Detektorpaar den Abstand zu messen und die gewünschte Distanz mit seinem rechten Servo einzuhalten.



Figur 8-4
Proportional-Steuerungs-Block-Diagram für den rechten Servo und das IR LED/Detektor Paar

Schauen wir uns die Zahlen von Figur 8-4 genauer an, um herauszufinden, wie Proportionalsteuerung funktioniert. Dieses spezielle Beispiel ist für die rechte IR LED/Detektorkombination und den rechten Servo geschrieben. Der Sollwert (*set point*) ist 2, was bedeutet, wir möchten, dass der Boe-Bot einen Abstand von 2 zwischen sich und jedem entdeckten Objekt einhält. Die gemessene Distanz ist 4, was zu weit weg ist. Der Fehler ist der Sollwert abzüglich die gemessene Distanz, also 2 – 4 = -2. Dies ist im Diagramm mit den Symbolen innerhalb des Kreises links angegeben. Diesen Kreis nennt man Summierglied (*summing junction*). Der Fehlerwert wird nun in ein Berechnungselement (*operator block*) eingegeben. Dieses Element zeigt, dass der Fehlerwert mit einem festen Wert multipliziert wird, der Proportionalkonstante (*proportional constant*) "Kp" heisst. Der Wert von Kp ist 35. Der Output des Elementes zeigt das Ergebnis von -2 × 35 = –70, das Korrekturwert (*output adjust*) heisst. Dieser Korrekturwert fliesst in ein weiteres Summierglied und wird diesmal zur Mitten-

Pulslänge des Servos von 750 addiert. Das Ergebnis ist eine Pulsweite von 680, was den Servo des Boe-Bot mit etwa ¾ Geschwindigkeit im Uhrzeigersinn drehen lässt. Diese Drehung bewirkt, dass das rechte Rad des Boe-Bot vorwärts, zum Objekt hin rollt. Die Korrektur fliesst ins Gesamtsystem ein, das aus dem Boe-Bot und dem Objekt besteht, das in einem Abstand von 4 festgestellt wurde.

Beim nächsten Schleifendurchlauf kann sich der gemessene Abstand ändern, aber das ist Ok, weil dieser Regelkreis unabhängig von der gemessenen Distanz einen Korrekturwert berechnet, der eine Servobewegung zur Korrektur des Fehlers auslösen wird. Die Korrektur ist immer proportional zum gemessenen Fehlerwert, der die Differenz zwischen eingegebenem Sollwert und effektiv gemessenem Abstand (=Istwert) darstellt.

Ein Regelkreis hat immer einen Satz von Gleichungen, die das System steuern. Das Blockdiagramm von Figur 8-4 ist eine Möglichkeit, diesen Satz von Gleichungen visuell zu beschreiben. Hier sind die Gleichungen, die aus dem Diagramm abgeleitet werden können, zusammen mit Lösungen.

$$
\begin{aligned}
\textit{Fehler} &= \textit{Rechter Sollabstand} - \textit{Gemessener rechter Abstand} \\
&= 2 - 4 \\
\textit{Korrekturwert} &= \textit{Fehler} \times \textit{Proportionalkonstante } K_p \\
&= -2 \times 35 \\
&= -70 \\
\textit{Rechter Servo Output} &= \textit{Korrekturwert} + \textit{Mittenpulsbreite} \\
&= -70 + 750 \\
&= 680
\end{aligned}
$$

Durch einige Ersetzungen können die drei Gleichungen oben in die folgende zusammengefasst werden, welche das selbe Ergebnis liefert.

$$
\begin{aligned}
\textit{Rechter Servo Output} = {} & (\textit{Rechter Sollabstand} - \textit{Gemessener rechter Abstand}) \times Kp \\
& + \textit{Mittenpulsbreite}
\end{aligned}
$$

Wenn wir in die Gleichung die Werte aus dem Beispiel einsetzen sehen wir, dass die Gleichung immer noch funktioniert:

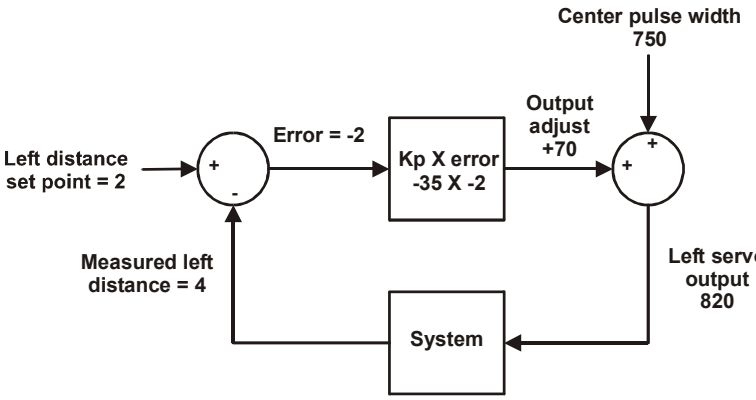
$$
\begin{aligned}
&= ((2 - 4) \times 35) + 750 \\
&= 680
\end{aligned}
$$

Servo und IR-Paar links haben einen gleichartigen Algorithmus (Figur 8-5). Der Unterschied ist, dass Kp nun –35 statt +35 ist. Unter der Annahme, dass wir denselben

Wert wie am rechten IR-Paar messen ergibt sich ein Korrekturwert für eine Pulsbreite von 820. Hier sind die Gleichung und die Berechnung für dieses Blockdiagramm:

$$\begin{aligned} \textit{Linker Servo Output} &= \textit{(Linker Sollabstand – Gemessener linker Abstand)} \times \textit{Kp} \\ &\quad + \textit{Mittenpulsbreite} \\ &= \textit{((2 – 4)} \times \textit{–35) + 750} \\ &= \textit{820} \end{aligned}$$

Das Ergebnis aus diesem Regelkreis ist eine Pulsbreite, die den linken Servo mit etwa ¾ Geschwindigkeit im Gegenuhrzeigersinn drehen lässt. Dies entspricht einer Vorwärtsbewegung des linken Rades.. Die Idee der Rückkoppelung (*feedback*) ist, dass der Effekt des Output des Systems vom Schatten-Bot durch eine neue Distanzmessung wieder aufgenommen wird. Dann wiederholt sich der Regelkreis immer wieder…etwa 40 mal pro Sekunde.



Figur 8-5
Proportional-Steuerungs-Block-Diagram für den linken Servo und das linke IR LED/ Detektor Paar

Programmierung des Boe-Bot als Schattenfahrzeug

Erinnern Sie sich, dass die Gleichung für den Output des rechten Servos war:

$$\begin{aligned} \textit{Right servo output} &= \textit{(Right distance set point – Measured right distance)} \times \textit{Kp} \\ &\quad + \textit{Center pulse width} \end{aligned}$$

Hier ist ein Beispiel, wie man diese Gleichung in PBASIC lösen kann. Der rechte Abstandssollwert ist 2, die gemessene Ist-Distanz ist in einer Variablen namens `distanceRight` gespeichert, welche die IR-Distanzmessung aufnimmt, Kp ist 35, und die Mittenpulslänge ist 750:

```
pulseRight = 2 - distanceRight * 35 + 750
```

Erinnern Sie sich, dass mathematische Ausdrücke in PBASIC von links nach rechts ausgeführt werden. Zuerst wird **`distanceRight`** von **2**. subtrahiert. Das Ergebnis der Subtraktion wird dann mit **`Kpr`** multipliziert und das Produkt wird danach zur Mittenpulslänge addiert.

Sie können Klammern verwenden, um die vorgängige Berechnung eines Terms, der weiter rechts in einer Zeile PBASIC Programmcode steht zu erzwingen. Erinnern Sie sich an dieses Beispiel: Sie können die folgende Zeile PBASIC Code:

```
pulseRight = 2 - distanceRight * 35 + 750
```

auch folgendermassen schreiben:

```
pulseRight = 35 * (2 – distanceRight) + 750
```

In diesem Ausdruck wird 35 mit dem Ergebnis von (**2 – distanceRight**) multipliziert, und dann das Produkt daraus zu 750 addiert.

8

Der linke Servo ist unterschiedlich, da Kp für dieses System –35 ist:

```
pulseLeft = 2 - distanceLeft * (-35) + 750
```

Da die Werte -35, 35, 2, und 750 alle einen Namen haben, sind einige Konstanten-Deklarationen hier definitiv angebracht:

```
Kpl             CON     -35
Kpr             CON     35
SetPoint        CON     2
CenterPulse     CON     750
```

Mit diesen Konstanten Deklarationen im Programm können Sie den Namen **`Kpl`** anstelle von -35, **`Kpr`** anstelle von 35, **`SetPoint`** anstelle von 2, und **`CenterPulse`** anstelle von 750 einsetzen. Nach diesen Deklarationen und Ersetzungen lesen sich die Berechnungen für die Proportionalsteuerung wie folgt:

```
pulseLeft  = SetPoint - distanceLeft  * Kpl + CenterPulse
pulseRight = SetPoint - distanceRight * Kpr + CenterPulse
```

Das Praktische an der Deklaration dieser Konstanten ist, dass Sie diese an einem einzigen Ort ändern können, nämlich am Anfang des Programms. Alle Änderungen, die Sie am Anfang des Programms machen werden automatisch überall nachgeführt, wo diese

Konstanten vorkommen. Wenn Sie zum Beispiel die **`Kpl CON`** Direktive von -35 auf –40 ändern, ändert auch jedes Vorkommen von **`Kpl`** im gesamten Programm von -35 auf -40. Dies ist überaus hilfreich beim Experimentieren und Tunen der rechten und linken Regelkreise.

Beispielprogramm: FollowingBoeBot.bs2

FollowingBoeBot.bs2 wiederholt die diskutierte Proportionalregelung mit jedem Servopuls. Mit anderen Worten, vor jedem Puls wird der Abstand gemessen und der Fehlerwert bestimmt. Dann wird der Fehler mit Kp multipliziert und der resultierende Wert wird zu/von der Pulsbreite des linken/rechten Servos addiert/subtrahiert.

- √ Erfassen, speichern und starten Sie FollowingBoeBot.bs2.
- √ Richten Sie den Boe-Bot auf ein Blatt Papier (A4, 8 ½ × 11") aus, das Sie vor den Boe-Bot halten, wie wenn es eine Hindernismauer wäre. Der Boe-Bot sollte einen stets gleichbleibenden Abstand zwischen sich und dem Blatt Papier einhalten.
- √ Drehen Sie das Papier vorsichtig ein bisschen. Der Boe-Bot sollte ebenfalls drehen.
- √ Benutzen Sie das Blatt Papier, um den Boe-Bot herumzuführen. Der Boe-Bot sollte dem Papier folgen.
- √ Bewegen Sie das Papier zu nahe an den Boe-Bot, sollte er rückwärts vom Papier weg fahren.

```
' -----[ Title ]-------------------------------------------------------------
' Robotics with the Boe-Bot - FollowingBoeBot.bs2
' Boe-Bot adjusts its position to keep objects it detects in zone 2.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

DEBUG "Program Running!"

' -----[ Constants ]----------------------------------------------------------

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

' -----[ Initialization ]--------------------------------------------------

FREQOUT 4, 2000, 3000

' -----[ Main Routine ]-----------------------------------------------------

DO

  GOSUB Get Ir Distances

  ' Calculate proportional output.

  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
```

Wie das Programm funktioniert

FollowingBoeBot.bs2 deklariert mit der **CON** Direktive vier Konstanten, **Kpr**, **Kpl**, **SetPoint**, und **CenterPulse**. Überall wo Sie **SetPoint** sehen, ist es in Wahrheit die Zahl 2 (eine Konstante). Entsprechend ist überall wo **Kpl** steht, eigentlich die Zahl –35, **Kpr** ist in Wirklichkeit 35, und **CenterPulse** ist 750.

```
Kpl             CON     -35
Kpr             CON     35
SetPoint        CON     2
CenterPulse     CON     750
```

Das erste, was die Hauptroutine tut, ist die **Get_Ir_Distances** Subroutine aufzurufen. Nachdem die **Get_Ir_Distances** Subroutine fertig ist, enthalten **distanceLeft** und **distanceRight** je eine Zahl für das linke und rechte IR-Paar, entsprechend dem Distanzbereich, in welcher ein Objekt erkannt wurde.

```
DO

  GOSUB Get_Ir_Distances
```

Die folgenden zwei Zeilen implementieren die Berechnungen der Proportionalsteuerung für beide Servos.

```
  ' Calculate proportional output.

  pulseLeft =  SetPoint - distanceLeft  * Kpl + CenterPulse
  pulseRight = SetPoint - distanceRight * Kpr + CenterPulse
```

Nachdem nun die **pulseLeft** und **pulseRight** Berechnungen gemacht sind, kann die **Send_Pulse** Subroutine aufgerufen werden.

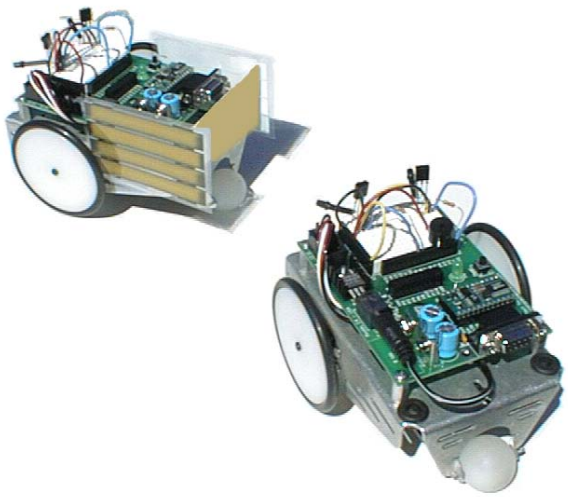
```
  GOSUB Send_Pulse
```

Der **LOOP** Teil des **DO**…**LOOP** sendet das Programm zurück zu dem Befehl, der unmittelbar auf das **DO** am Anfang der Hauptroutine folgt.

```
LOOP
```

Sie sind dran

Figur 8-6 zeigt einen Führungs-Bot gefolgt von einem Schatten-Bot. Auf dem Führungs-Bot läuft eine modifizierte Version von FastIrRoaming.bs2, während auf dem Schatten-Bot FollowingBoeBot.bs2 läuft. Proportionalsteuerung macht aus dem Schatten-Bot einen wirklich treuen Begleiter. Ein Führungs-Bot kann eine Kette von 6 bis 7 Schatten-Bots hinter sich herziehen. Sie brauchen nur die übrigen Boe-Bots in der Beschattungskette auch mit Seitenwänden und einer Heckklappe zu versehen.



Figur 8-6
Führungs Boe-Bot (links) und Schatten- Boe-Bot (rechts)

- √ Wenn Sie in einer Schulklasse sind, montieren Sie Kartonwände ans Heck und die Seitenwände des Führungs-Boe-Bot wie in Figur 8-6.
- √ Wenn Sie nicht zu einer Klasse gehören (und nur einen Boe-Bot haben) wird das Schattenfahrzeug einem Stück Papier oder Ihrer Hand genauso treu folgen, wie es einem Führungs-Boe-Bot folgt.
- √ Ersetzen Sie die 1 kΩ Widerstände die beim Führungs-Boe-Bot P2 und P8 mit den IR LEDs verbinden mit 470 Ω oder 220 Ω Widerständen.
- √ Programmieren Sie den Führungs-Boe-Bot zwecks Objektvermeidung um, indem Sie eine modifizierte Version von FastIrRoaming.bs2 verwenden. Öffnen Sie FastIrRoaming.bs2, und nennen Sie es neu SlowerIrRoamingForLeadBoeBot.bs2.
- √ Nehmen Sie an SlowerIrRoamingForLeadBoeBot.bs2 folgende Änderungen vor:

- √ Erhöhen Sie alle **PULSOUT *Duration*** Arguments von 650 auf 710.
- √ Reduzieren Sie alle **PULSOUT *Duration*** Argument von 850 auf 790.

√ Der Schatten-Bot sollte weiterhin unverändert mit FollowingBoeBot.bs2 laufen.

√ Wenn auf beiden Boe-Bots die entsprechenden Programme laufen, stellen Sie den Schatten-Bot hinter den Führungs-Boe-Bot. Der Schatten-Bot sollte ihm in festem Abstand nachfahren, solange er nicht von einem anderen Objekt, wie zum Beispiel Ihrer Hand oder einer Wand in der Nähe abgelenkt wird.

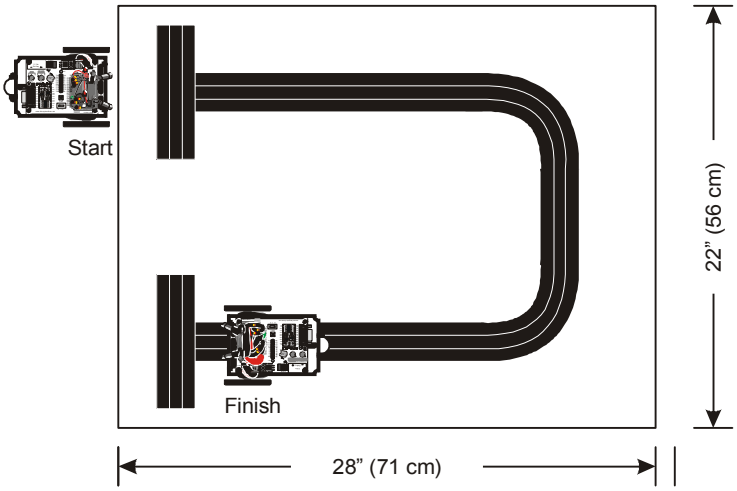
Sie können die Sollwerte und die Proportionalkonstante anpassen, um das Verhalten des Schatten-Bot zu ändern. Benutzen Sie ein Blatt Papier oder Ihre Hand, um den Schatten-Bot währen dieser Übung zu führen:

√ Lassen Sie mal FollowingBoeBot.bs2 mit Werten für **Kpr** und **Kpl** zwischen 15 und 50 laufen. Beachten Sie den Unterschied wie und wann der verantwortliche Boe-Bot einem Objekt folgt.

√ Versuchen Sie Einstellungen im Wert der **SetPoint** Konstanten. Nehmen Sie Werte zwischen 0 und 4.

AKTVITÄT 3: EINEM STREIFEN FOLGEN

Figur 8-7 zeigt ein Beispiel einer Rennstrecke, die Sie aufbauen können und für die der Boe-Bot programmiert werden kann. Jeder Streifen in dieser Strecke besteht aus drei Stück 19mm (¾ in) Plastik-Isolierband (*electrical tape*), die dicht an dicht nebeneinander auf einen weissen Karton geklebt sind. Zwischen den Streifen sollte kein Karton mehr hervorlugen.



Figur 8-7
Spurfolger
Piste

Aufbau und Test der Strecke

Für die erfolgreiche Navigation über diese Strecke sind ein bisschen Testen und Anpassungsarbeit am Boe-Bot nötig.

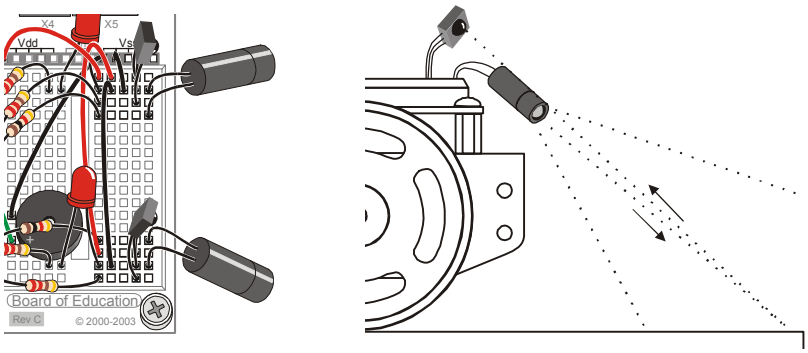
Benötigtes Material

(1) Ein Stück Karton – ungefähre Ausmasse: 56 x 71 cm (22 x 28 in)
(1) Rolle schwarzes Plastik-Isolierband – 19 mm (¾") breit.

√ Bauen Sie mit dem Karton und dem Isolierband die Strecke aus Figur 8-7.

Test des Streifens

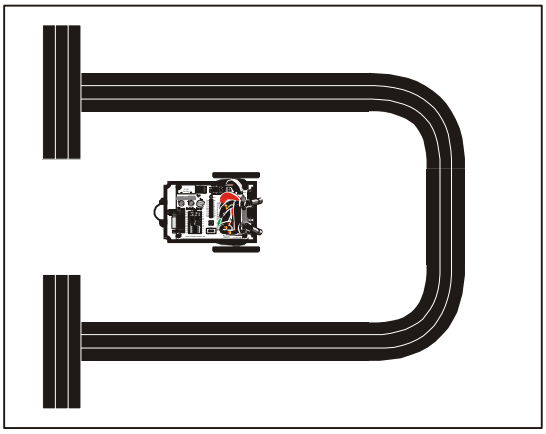
√ Zielen Sie mit Ihren IR-Paaren abwärts und nach aussen, wie in Figur 8-8 gezeigt (entspricht der Figur 7-8 von Seite 253 und ist hier bequemlichkeitshalber nochmals abgebildet).



Draufsicht Seitenansicht

Figur 8-8
Nach unten gerichtete IR Paare suchen nach dem Streifen

- √ Überprüfen Sie, dass Ihre Isolierbandstrecke frei von Leuchtstoffröhren-Störeinflüssen ist. . Vergleichen Sie dazu den Anhang: Schnüffeln nach IR-Störungen.
- √ Starten Sie DisplayBothDistances.bs2 von Seite 269. Lassen Sie Ihren Boe-Bot am seriellen Kabel angeschlossen, so dass Sie die angezeigten Abstände sehen können.
- √ Stellen Sie Ihren Boe-Bot zunächst so auf, dass er direkt auf den weissen Hintergrund Ihres Kartons schaut wie in Figur 8-9.
- √ Überprüfen Sie, dass die Zonenanzeigen ein Objekt in sehr kurzem Abstand anzeigen. Beide Sensoren sollten eine 1 oder 0 Ablesung haben.

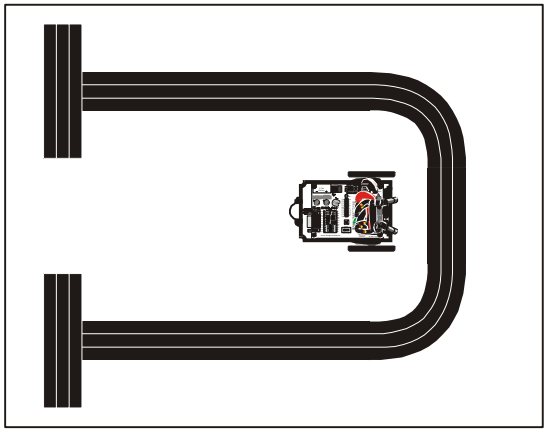


Figur 8-9
Test auf tiefe Zonenzahl - Draufsicht

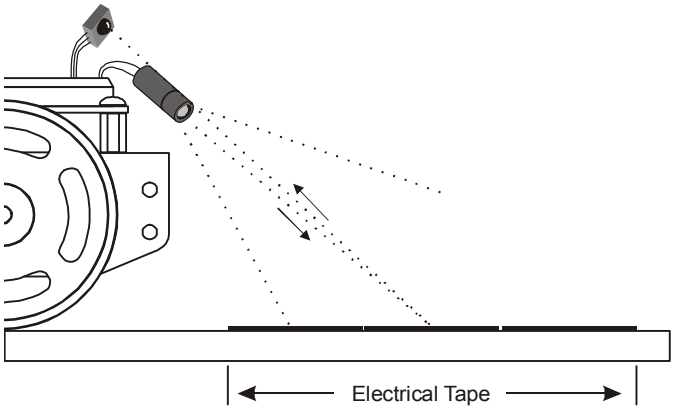
√ Stellen Sie nun Ihren Boe-Bot so auf, dass beide IR-LED/Detektor-Paare direkt auf die Mitte Ihrer Isolierbandpiste zeugen (vgl. Figur 8-10 und Figur 8-11).

√ Verändern Sie nun die Position Ihres Boe-Bot (näher an den Streifen oder weiter weg), bis beide Zonenanzeigen Level 4 oder 5 erreichen und so anzeigen, dass kein, oder ein weit entferntes Objekt erkannt wurde.

√ Wenn Sie Schwierigkeiten haben, die höheren Ablesungen mit Ihrer Isolierbandstrecke zu erhalten, konsultieren Sie den Anhang Fehlersuche an der Isolierbandstrecke auf Seite 284.



Figur 8-10
Test auf hohe Zonenzahl - Draufsicht



Figur 8-11
Test auf hohe Zonenzahl - Seitenansicht

Fehlersuche an der Isolierbandstrecke

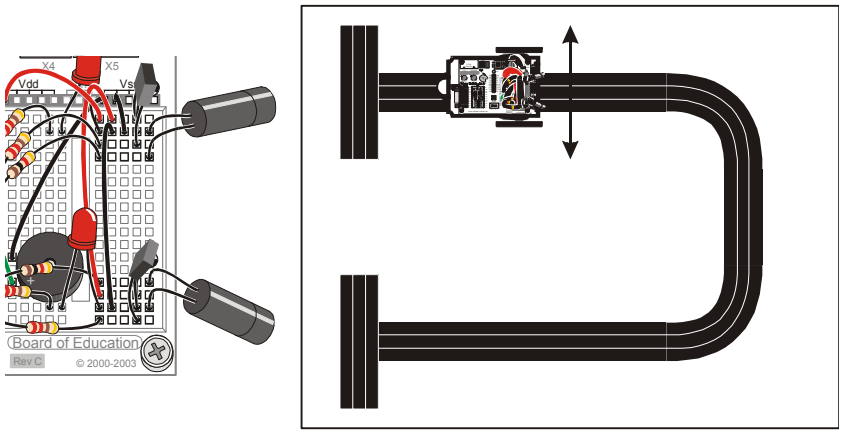
Wenn es Ihnen nicht gelingt, einen hohen Zonenwert zu erhalten, wenn die IR-Detektoren auf das Isolierband fokussiert sind, nehmen Sie ein separates Stück Papier und machen Sie eine Strecke, die vier Klebstreifen breit ist, statt drei. Wenn die Zonenangaben immer noch tief sind, überprüfen Sie, dass Sie 1 kΩ Widerstände (braun-schwarz-rot) in Serie mit Ihren IR-LEDs verwenden. Wenn nichts davon funktioniert, versuchen Sie es mit einer anderen Sorte schwarzem Isolierband. Auch eine neue Ausrichtung von IR-LED und Detektor (vgl. Figur 8-11), so dass der Fokus näher am Boe-Bot oder weiter davon entfernt liegt, kann helfen.

Wenn Sie die älteren IR LEDs im Schrumpfschlauch verwenden, anstelle derjenigen mit den 2-teiligen Plastikabdeckungen, könnten Sie Mühe haben, eine tiefe Zonennummer zu erhalten, wenn die IR LED/Detektorkombination auf den weissen Untergrund ausgerichtet ist. Diese LEDs benötigen oft einen 470 Ω (gelb-violett-braun) oder 220 Ω (rot-rot-braun) Widerstand in Serie. Versichern Sie sich auch, dass sich die Anschlüsse der IR-LEDs, nicht berühren.

√ Stellen Sie nun den Boe-Bot so auf die Strecke, dass die schwarze Piste zwischen den Rädern des Boe-Bot liegt. Die IR-Detektoren sollten leicht nach aussen schauen. Siehe dazu die Detailzeichnung in Figur 8-12. Überprüfen Sie, dass die Abstandsanzeigen für beide IR-Paare wiederum 0 oder 1 ist. Wenn die Ablesungen höher sind, so müssen Sie die IR-Paare noch etwas mehr nach aussen biegen, vom Rand der Piste weg.

Wenn Sie den Boe-Bot in eine der vom Doppelpfeil angegebenen Richtungen drehen wird eines der IR-Paare auf das Isolierband fokussieren. Wenn das passiert sollten sich die Ablesungen für dieses Paar auf 4 bis 5 erhöhen. Denken Sie daran, wenn Sie den Boe-Bot nach links schieben, sollte der rechte Detektor die Werte erhöhen, und wenn Sie den Boe-Bot nach rechts bewegen, sollte der linke Detektor die höheren Ablesungen zeigen.

√ Justieren Sie die IR LED/Detektor Paare so lange, bis der Boe-Bot diesen letzten Test besteht. Dann sind Sie bereit für den Versuch, der Piste zu folgen.



IR Paare Vergrösserung

Boe-Bot über dem Streifen (Draufsicht)

Figur 8-12
Streifen-Erkennungs-Test

Programmierung zur Streifenverfolgung

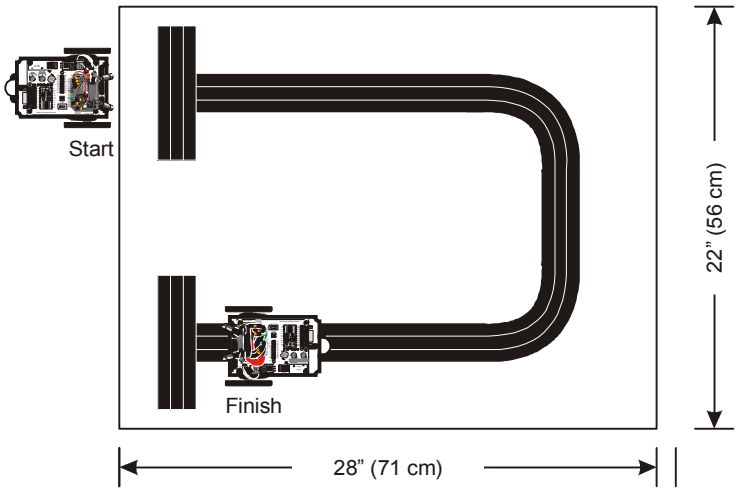
Es braucht nur einige kleine Änderungen an FollowingBoeBot.bs2. von Seite 276 damit es für die Streifenverfolgung funktioniert. Zunächst einmal sollte der Boe-Bot sich auf die Objekte zu bewegen, die näher liegen als der **`SetPoint`** Sollwert, und von Objekten weg, die weiter weg sind als der **`SetPoint`** Sollwert. Das ist das Gegenteil vom Verhalten von FollowingBoeBot.bs2. Um die Fahrtrichtung des Boe-Bot umzudrehen, wenn er ein Objekt ausserhalb der **`SetPoint`** Distanz erkennt, genügt es, die Vorzeichen von **`Kpl`** und **`Kpr`** zu tauschen. Mit anderen Worten, ändern Sie **`Kpl`** von -35 auf 35 und **`Kpr`** von 35 auf -35. Sie werden wohl ein bisschen mit Ihrem **`SetPoint`** Sollwert experimentieren müssen. Werte von 2 bis 4 funktionieren meist am besten. Das folgende Beispielprogramm verwendet einen **`SetPoint`** von 3.

Beispielprogramm: StripeFollowingBoeBot.bs2

- √ Öffnen Sie FollowingBoeBot.bs2 und speichern Sie es als StripeFollowingBoeBot.bs2.
- √ Ändern Sie die **`SetPoint`** Deklaration von **`SetPoint CON 2`** auf **`SetPoint CON 3`**.
- √ Ändern Sie **`Kpl`** von -35 to 35.
- √ Ändern Sie **`Kpr`** von 35 to -35.
- √ Starten Sie das Programm wie unten gezeigt.
- √ Setzen Sie Ihren Boe-Bot auf den Startplatz wie in Figur 8-13. Der Boe-Bot sollte dort warten, bis Sie Ihre Hand vor seine IR-Paare halten. Er wird dann

vorwärts rollen. Wenn es den Startstreifen überquert, nehmen Sie Ihre Hand weg. Er sollte nun beginnen, dem Streifen zu folgen. Wenn er den Zielstreifen sieht, sollte er anhalten und dort warten.

√ Angenommen Sie erhalten Distanzanzeigen zwischen 5 auf dem Isolierband und 0 auf dem Kartonhintergrund, sollten auch Werte für die **`SetPoint`** Konstante von 2, 3 und 4 funktionieren. Probieren Sie verschiedene **`SetPoint`** Werte aus und notieren Sie die Unterschiede in der Leistung des Boe-Bot auf der Piste.



Figur 8-13
Piste zur Streifen-Verfolgung

```
' -----[ Title ]-------------------------------------------------------------
' Robotics with the Boe-Bot - StripeFollowingBoeBot.bs2
' Boe-Bot adjusts its position to move toward objects that are closer than
' zone 3 and away from objects further than zone 3.  Useful for following a
' 2.25 inch wide vinyl electrical tape stripe.

' {$STAMP BS2}                                ' Stamp directive.
' {$PBASIC 2.5}                               ' PBASIC directive.

DEBUG "Program Running!"

' -----[ Constants ]----------------------------------------------------------

Kpl             CON     35                   ' Change from -35 to 35
Kpr             CON     -35                  ' Change from 35 to -35
SetPoint        CON     3                    ' Change from 2 to 3.
CenterPulse     CON     750

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

```
freqSelect     VAR     Nib
irFrequency    VAR     Word
irDetectLeft   VAR     Bit
irDetectRight  VAR     Bit
distanceLeft   VAR     Nib
distanceRight  VAR     Nib
pulseLeft      VAR     Word
pulseRight     VAR     Word

' -----[ Initialization ]----------------------------------------------------

FREQOUT 4, 2000, 3000

' -----[ Main Routine ]-------------------------------------------------------

DO

  GOSUB Get_Ir_Distances

  ' Calculate proportional output.

  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
```

Sie sind dran – Spurverfolgungs - Wettbewerb

Sie können aus diesem Experiment einen Wettbewerb machen, wer die kürzeste Fahrzeit erreicht, vorausgesetzt der Boe-Bot wartet getreulich an den “Start” und “Ziel”-Markierungen. Sie können auch andere Strecken herstellen. Um die maximale Leistung zu erreichen, experimentieren Sie mit verschiedenen Werten für **`SetPoint`**, **`Kpl`**, und **`Kpr`**.

ZUSAMMENFASSUNG

Der Frequenz-Sweep wurde vorgestellt als Möglichkeit, mit der IR LED und dem Detektor des Boe-Bot Distanzen zu bestimmen. Wie nutzen **FREQOUT** um IR Signale im Frequenzbereich von 37.5 kHz (am empfindlichsten) bis 41.5 kHz (am wenigsten empfindlich). Die Entfernung wurde bestimmt, indem überprüft wurde, welche Frequenzen im IR Detektor ein Erkennungssignal erzeugten, und welche nicht. Da nicht alle Frequenzen denselben Abstand voneinander haben, wurde der **LOOKUP** Befehl eingeführt als einfache Art, um mit der Zählsequenz einer **FOR**…**NEXT** Schleife in eine einfache Zahlenreihe zu verweisen.

Verschiedene Steuersysteme wurden betrachtet zusammen mit dem geschlossenen Regelkreis (closed loop control). Proportionalsteuerung (proportional control) in einem geschlossenen Regelkreis ist ein Algorithmus, bei dem die gemessene Abweichung (error) mit einer Proportionalitätskonstante (proportionality constant) multipliziert wird, um den Korrekturfaktor zu bestimmen. Die Abweichung entspricht dem vom geregelten System produzierte Output (Istwert) abzüglich dem eingestellten Sollwert (set point). Beim Boe-Bot erschienen sowohl der Systemoutput als auch der Sollwert als Distanzen. Die BASIC Stamp wurde in PBASIC so programmiert, dass sie sowohl die linken und rechten Servos, als auch die Distanzdetektoren in einem geschlossenen Regelkreis steuerte. Durch Nachmessen der Distanzen und Anpassung der Impulslänge für die Servos konnte der Regelkreis den Boe-Bot auf Bewegungen des Objektes reagieren lassen. Der Boe-Bot war dank der Proportionalsteuerung in der Lage, sich an Objekte zu heften und ihnen zu folgen, wie auch um einen Streifen Isolierband zu erkennen und ihm zu folgen.

Fragen

1. Wie hoch wäre die relative Empfindlichkeit des IR Detektors, wenn Sie den **FREQOUT** Befehl verwenden würden um eine 35 kHz Harmonische zu senden? Wie hoch ist die relative Empfindlichkeit mit einer 36 kHz Harmonischen?
2. Wenn Sie ein Objekt in die Zone 1 bringen, mit welchen Frequenzen kann das Objekt erkannt werden und mit welchen nicht? Wie ist das in Zone 4?
3. Betrachten Sie den Code-Schnipsel unten. Wenn die **index** Variable 4 ist, welche Zahl wird dann in die **prime** Variable in diesem **LOOKUP** Befehl eingesetzt? Welche Werte wird **prime** speichern, wenn **index** 0, 1, 2 und 7 ist?

```
LOOKUP index, [2, 3, 5, 7, 11, 13, 17, 19], prime
```

4. Was bewirkt der Befehl `irDetect = IN9` in den Beispielprogrammen in diesem Kapitel?
5. In Figur 8-4, was wäre der Fehler, wenn der Sollwert 4 ist und die gemessene Distanz rechts 1? Wie gross wäre die Output-Anpassung? Was ist mit dem rechten Servo Output?
6. In welcher Reihenfolge werden in PBASIC mathematische Ausdrücke berechnet? Wie können Sie diese Reihenfolge übersteuern?
7. Welche PBASIC Direktive verwenden Sie um eine Konstante zu deklarieren? Wie geben Sie der Zahl 100 den Namen "Siedepunkt"?
8. Wie unterscheiden sich StripeFollowingBoeBot.bs2 und FollowingBoeBot.bs2? Wie ermöglicht es dieser Unterschied dem Boe-Bot, einem Streifen zu folgen? Was passiert mit der gemessenen Distanz, wenn der Boe-Bot beginnt vom Streifen wegzudriften? Wie sorgen die Proportionalberechnungen dafür, dass der Boe-Bot zurück auf den richtigen Weg findet?

Übungen

1. Erstellen Sie eine Liste der Empfindlichkeit des IR Detektors für alle Frequenzen (kHz) in Figur 8-1.
2. Erstellen Sie eine Liste der Ja/Nein-Muster, die TestLeftFrequencySweep.bs2 zeigt, wenn ein Objekt in jede der Zonen von Figur 8-2 gestellt wird.
3. Schreiben Sie ein Codesegment, das den Frequenz-Sweep nur für vier statt für fünf Frequenzen durchführt.
4. Führen sie die Berechnungen der Proportionalsteuerung für den rechten Servo (Figur 8-4) für jede mögliche gemessene Distanz (0 bis 5) durch. Wiederholen Sie das für Figur 8-5.
5. Erstellen Sie eine zusammengefasste Checkliste für die Tests, die vorgenommen werden sollten, um eine spurtreue Linienfolge zu gewährleisten

Projekte

1. Ändern Sie TestLeftFrequencySweep.bs2 von Seite 268 so, dass es die Piezolautsprecher verwendet, um die gemessene Distanz anzuzeigen. Dabei gibt es zwei Varianten, die eine nähere Betrachtung rechtfertigen. Eine Variante verwendet nur einen Ton, aber lässt ihn schneller piepsen, je näher das Objekt kommt. Die andere Variante belässt das Piepsen in einer konstanten Rate, aber erhöht die Tonlage, wenn das Objekt näher kommt, bzw. piepst in tieferer Frequenz, wenn das Objekt weiter weg ist.

2. Entwerfen Sie ein System, in dem beide IR Detektoren die gemessenen Distanzen mit dem Blinken der linken bzw. rechten LED anzeigen, wobei die Blinkgeschwindigkeit die Distanz zum festgestellten Objekt anzeigt. Hinweis: Beginnen Sie mit dem Schema der LED Schaltung und dem Verdrahtungsdiagramm aus Figur 7-4 und Figur 7-5 (Seite 236) und DisplayBothDistances.bs2 (Seite 269).

   Dieses Projekt ist nicht so einfach, wie es aussieht. Es verlangt die Entwicklung eines freilaufenden Zählers innerhalb der **DO**…**LOOP** Schleife, der immer grösser wird. Der **DO**…**LOOP** sollte auch einen **PAUSE** Befehl haben mit einer *duration* die zu ihrer Mindest-Blinkgeschwindigkeit passt. Verwenden Sie bei jedem Durchlauf durch die Schleife **IF**…**THEN** um den Zähler mit dem Wert des Zählers zu vergleichen, der den Umschaltzeitpunkt der LED steuert. Jedes Mal, wenn die LED umschaltet müssen Sie die Distanz erneut überprüfen und berechnen, wie viele Zählerschritte Sie bis zum nächsten Umschalten der LED noch warten müssen. Beachten Sie ausserdem, dass als Word (8 Bit) definiert Variablen keine Zahlen grösser als 65535 speichern können. Es ist vielleicht nützlich, den Zähler auf Null zurückzustellen, bevor er diesen Wert erreicht. Sonst könnte es zu überraschenden Effekten kommen.

3. Wenn der Boe-Bot das Programm FollowingBoeBot.bs2 ausführt, und Sie ihn vor ein unbewegliches Objekt stellen, wird der Boe-Bot diesem getreulich folgen, was nicht besonders spannend ist zum Zuschauen, weil der Boe-Bot ja einfach stillsteht. Das Ziel wäre es, dass der Boe-Bot realisiert, dass er nichts tut, dass er darauf aus der Proportionalsteuerschleife ausbricht, z.B. eine 120° Drehung ausführt, und darauf wieder in die Proportionalsteuerung zurückkehrt. Hinweis: Sie können das erreichen, indem Sie zwei Zähler vorsehen. Einer, der bei jedem Durchlauf durch die **DO**…**LOOP** Schleife um 1 hochzählt, und ein anderer, der nur hochzählt, wenn der Boe-Bot einen Vorwärts-Puls an die Servos liefert. Wenn der Schleifenzähler 60 erreicht, können Sie mit **IF**…**THEN** prüfen, wie viele Vorwärts-Pulse generiert worden sind. Wenn weniger als 20 Pulse generiert wurden, könnte der Boe-Bot festsitzen, also rufen Sie eine Subroutine auf, die eine Drehung (ohne Proportionalsteuerung) ausführt. Vergessen Sie nicht, die beiden Zähler auf Null zurückzusetzen, wenn der Schleifenzähler 60 erreicht hat.

4. *Projekt für Fortgeschrittene*- Erstellen Sie verschiedene Kreuzungen von Isolierband-Strecken und Programmieren Sie den Boe-Bot darauf zu erkennen, was für eine Art Kreuzung es ist. Die Kreuzungen können z.B. 90° links, 90° rechts, Dreiweg (Y) oder Vierweg (X) sein. Hinweis: Dies verlangt, dass der Boe-Bot erkennt, dass er eine Kreuzung vor sich hat, was ähnlich ist wie in Projekt 3. Nachdem der Boe-Bot die Kreuzung erkannt hat, muss er eine vorprogrammierte Bewegung machen um sich auf die Mitte der Kreuzung zu stellen und herauszufinden, ob er rechts und links Isolierband oder weissen Untergrund sieht. Dann muss er sich ein bisschen vorwärts bewegen und vielleicht etwas drehen, um herauszufinden, ob die Spur weiterführt oder nicht.

5. *Projekt für Fortgeschrittene* – Wenn Sie Projekt 4 erfolgreich abgeschlossen haben, fügen Sie weitere vorprogrammierte Bewegungen hinzu, um auf den Kreuzungen zu navigieren. Erstellen Sie ein Labyrinth aus Isolierbandstreifen. Schauen Sie sich den **`RANDOM`** Befehl im *BASIC Stamp Manual* nach, und benutzen Sie diesen, um die Fahrtrichtung an Y und X Kreuzungen zu bestimmen. Das Ziel ist, zu vermeiden, dass der Boe-Bot bei jedem Versuch denselben Weg durch das Labyrinth nimmt.

6. *Projekt für Fortgeschrittene* – Entwickeln Sie ihre eigene Wettbewerbsaufgabe für ein Streifen-Labyrinth, und programmieren Sie Ihren Boe-Bot für die Lösung!

A

Anhang A: Fehlersuche in der PC – BASIC Stamp Kommunikation

Hier finden Sie eine Liste der Dinge, die sie unternehmen können, um schnell Probleme bei der Kommunikation zwischen dem BASIC Stamp Editor und der BASIC Stamp zu beheben:

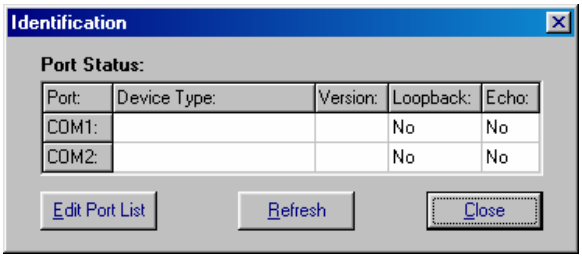
- √ Wenn Sie mit einem Board of Education Rev C arbeiten, versichern Sie sich, dass der Stromschalter auf Position 1 ist.
- √ Versichern Sie sich, dass das Problem nicht bei leeren Batterien oder einer falschen oder gar defekten Stromversorgung liegt, indem Sie eine neue 9 V Batterie benützen.
- √ Versichern Sie sich, dass das serielle Kabel fest im COM Port des Computers und dem DB9 Anschluss des Board of Education oder des BASIC Stamp HomeWork Board eingesteckt ist.
- √ Versichern Sie sich, dass sie ein normales serielles Kabel verwenden. BENÜTZEN SIE KEIN NULL-MODEM KABEL. (Ein Null-Modem Kabel ist ein serielles Kabel, bei dem Sende- und Empfangsleitung gekreuzt sind. Man verwendet solche Kabel, um zwei Computer direkt – ohne Modem, daher der Name "Null-Modem"– miteinander zu verbinden.) Die meisten Null-Modem Kabels sind mit NULL oder Null_Modem angeschrieben; untersuchen Sie das Kabel nach solch einer Kennzeichnung. Wenn Sie eine finden, können Sie das Kabel nicht zum Programmieren der BASIC Stamps gebrauchen.
- √ Stellen Sie alle Palmtop Kommunikationssoftware ab.

Wenn Sie eine BASIC Stamp und ein Board of Education benützen, überprüfen Sie auch folgendes:

- √ Versichern Sie sich, dass die BASIC Stamp richtig in die Fassung eingesteckt wurde, vergleichen Sie mit Figur 1-24 auf Seite 18.
- √ Wenn Sie ein Netzteil benützen, versichern Sie sich, dass es sowohl in der 230V-Steckdose als auch im Board of Education eingesteckt ist. Überprüfen Sie, ob das grüne Power-Licht am Board of Education leuchtet, wenn das Netzteil eingesteckt ist.
- √ Versichern Sie sich, dass die BASIC Stamp fest in der Fassung steckt. Schalten Sie zuerst den Strom aus, überprüfen Sie, ob die Markierungen und Pins richtig ausgerichtet sind und drücken Sie dann mit dem Daumen das Modul nach unten.

Schauen Sie auch nach, ob keine Pins unter dem Modul zusammengedrückt wurden, anstatt richtig in ihrer Vertiefungen auf dem Board of Education zu stecken.

Wenn ihr Identification Window der Abbildung unten ähnlich sieht, bedeutet dies, dass der BASIC Stamp Editor ihre BASIC Stamp auf keinem COM Port finden kann. Wenn dies ihr Problem ist, versuchen Sie folgendes:



Figur A-1
Identification Window

Beispiel: Keine BASIC Stamp an den COM Ports 1 und 2 gefunden.

- √ Schliessen Sie das Identification Window.
- √ Versichern Sie sich, dass das serielle Kabel richtig eingesteckt ist.
- √ Versuchen Sie den Run → Identify Test nochmals.
- √ Wenn Sie die Nummer des COM Port kennen, aber diese nicht im Identification Window erscheint, benützen Sie den Edit Port List Knopf um diesen COM Port hinzuzufügen, und dann versuchen Sie den Run → Identify Test nochmals.
- √ Wenn Sie mehr als einen COM port haben, versuchen Sie ihr Board of Education oder ihr BASIC Stamp HomeWork Board an einem anderen COM port anzuschliessen und versuchen Sie ob Run → Identify nun funktioniert.
- √ Wenn Sie einen zweiten Computer haben, versuchen Sie es auf diesem.
- √ Wenn keine dieser Lösungen hilft, gehen Sie zu www.parallax.com und folgen Sie dem Support Link (nur in englisch).

Wenn Sie die Fehlernachricht "No BASIC Stamp Found" (keine BASIC Stamp gefunden) bekommen, aber der *Run → Identify* Test ein "Yes" in beiden Spalten eines COM Ports anzeigt, müssen Sie möglicherweise die Einstellungen bei den FIFO Buffers ihres Computers ändern. Dies passiert Benutzern von Microsoft Windows® 98 und XP. Machen Sie eine Notiz, welcher der COM Ports die "Yes" Nachrichten hat, und versuchen Sie folgendes:

Windows® 98:

- √ Drücken Sie auf den Start-Knopf.
- √ Wählen Sie *Settings→ Control Panel → System → Device Manager → Ports (COM & LPT).*
- √ Wählen Sie den COM Port, der bei dem *Run → Identify* Test die "Yes" Nachrichten erhielt..
- √ Wählen Sie *Properties → Port Settings → Advanced*.
- √ Entfernen Sie die Markierung bei "Use FIFO Buffers" und drücken Sie dann auf *OK*.
- √ Drücken Sie *OK* um jedes Fenster zu schliessen und kehren Sie zum BASIC Stamp Editor zurück.
- √ Versuchen Sie nochmals, ein Programm zu downloaden.

Windows® 2000:

- √ Drücken Sie auf den *Start* Knopf.
- √ Wählen Sie *Settings → Control Panel → System → Hardware → Device Manager → Ports (COM & LPT).*
- √ Wählen Sie den COM Port, der bei dem *Run → Identify* Test die "Yes" Nachrichten erhielt.
- √ Wählen Sie → *Port Settings → Advanced*.
- √ Entfernen Sie die Markierung bei "Use FIFO Buffers" und drücken Sie dann auf *OK*.
- √ Drücken Sie *OK* um jedes Fenster zu schliessen und kehren Sie zum BASIC Stamp Editor zurück.
- √ Versuchen Sie nochmal ein Programm zu downloaden.

Windows® XP:

- √ Drücken Sie auf den *Start* Knopf.
- √ Wählen Sie *Control Panel → Printers and Other Hardware.*
- √ *In der "*See Also*"* box wählen Sie *System.*
- √ Wählen Sie *Hardware → Device Manager → Ports.*
- √ Geben Sie die COM Port Nummer ein, die beim *Run→ Identify* Test die "Yes" Nachrichten erhielt.
- √ Wählen Sie *Port Settings → Advanced.*
- √ Entfernen Sie die Markierung bei "Use FIFO Buffers" und drücken Sie dann auf *OK*.

- √ Drücken Sie auf *OK* um jedes Fenster zu schliessen und kehren Sie zum BASIC Stamp Editor zurück.
- √ Versuchen Sie nochmals ein Programm downzuloaden.

Windows® XP Pro:

- √ Drücken Sie auf den *Start* Knopf.
- √ Wählen Sie *Control Panel* → *System* → *Hardware* → *Device Manager* → *Ports(COM & LPT1)*.
- √ Wählen Sie die Kommunikationsportnummer, die beim *Run* → *Identify* Test die "Yes" Nachrichten erhielt.
- √ Wählen Sie *Properties* → *Port Settings* → *Advanced.*
- √ Entfernen Sie die Markierung bei “Use FIFO Buffers” und drücken Sie dann auf *OK*.
- √ Drücken Sie auf *OK* um jedes Fenster zu schliessen und kehren Sie zum BASIC Stamp Editor zurück.
- √ Versuchen Sie nochmals ein Programm downzuloaden.

Wenn keine dieser Lösungen funktioniert, können Sie bei www.parallax.com über den Support link weitere Hilfe erhalten (nur in englisch). Sie können auch ein eMail an support@parallax.com senden (ebenfalls in englisch).

Appendix B: BASIC Stamp and Carrier Board Components and Features

B

Die folgende Beschreibung der BASIC Stamp Module ist vor allem für den technisch Interessierten relevant. Sie ist daher im englischen Original belassen.

The BASIC STAMP® 2 Microcontroller Module

Figur B-1 shows a close-up of the BASIC Stamp® 2 microcontroller module. Its major components and their functions are indicated by labels.

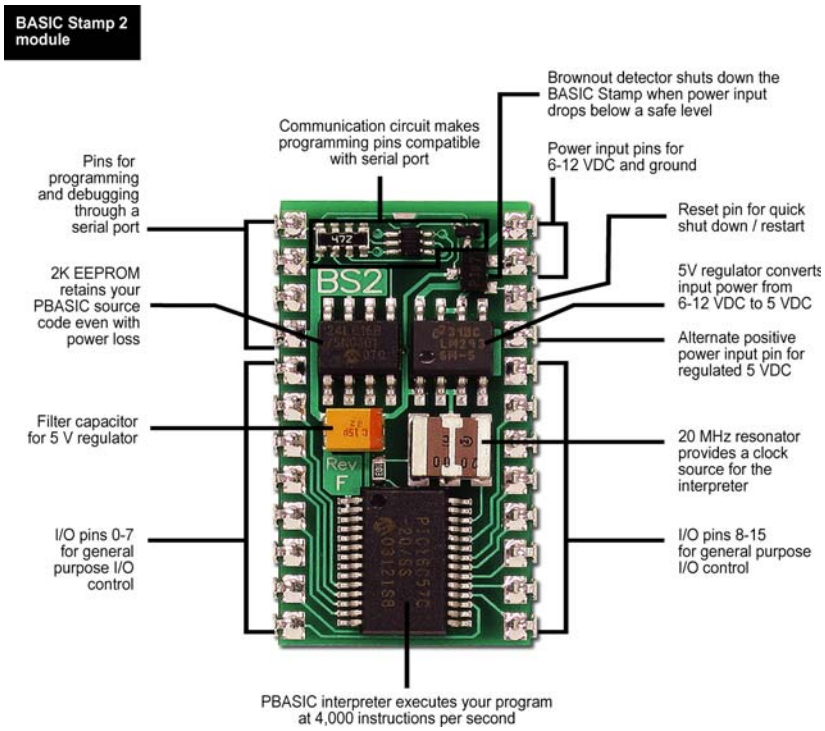


Figure B-1: BASIC Stamp® 2 Microcontroller Module Components and Their Functions

The Board of Education® Rev C Carrier Board

The Board of Education® Rev C carrier board for BASIC Stamp® 24-pin microcontroller modules is shown in Figure B-2. Its major components and their functions are indicated by labels.

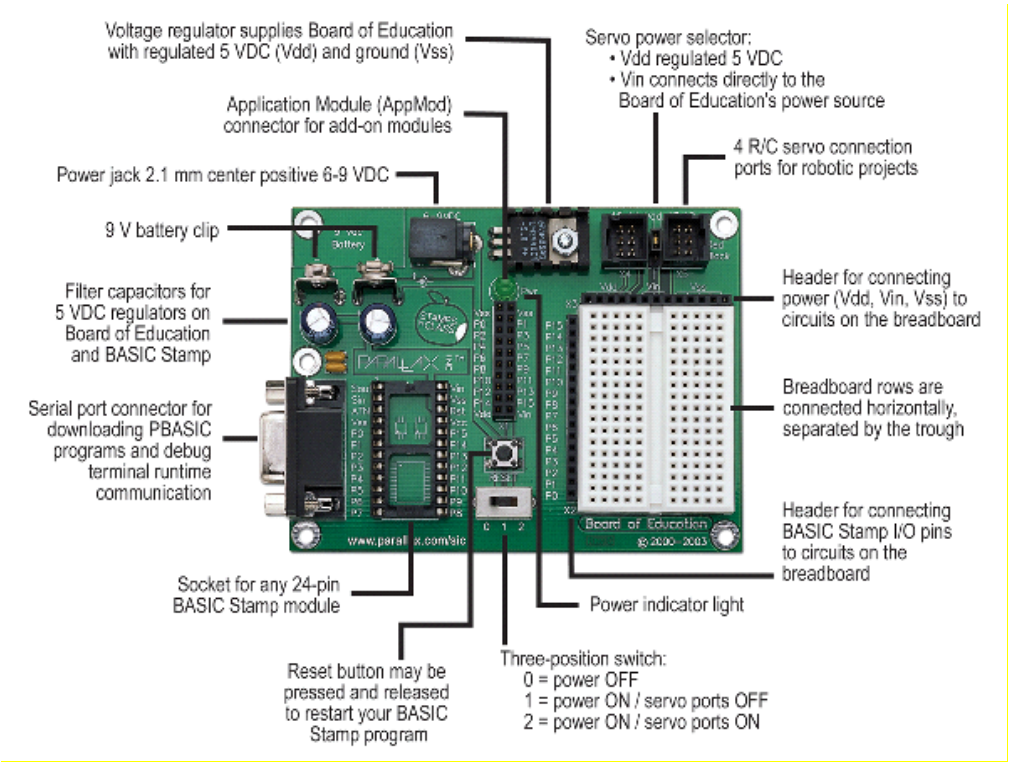


Figure B-2: Board of Education® Rev C Carrier Board

B

The BASIC Stamp® HomeWork Board™ Project Platform

The BASIC Stamp® HomeWork Board™ project platform is shown in Figure B-3. Its major components and their functions are indicated by labels.

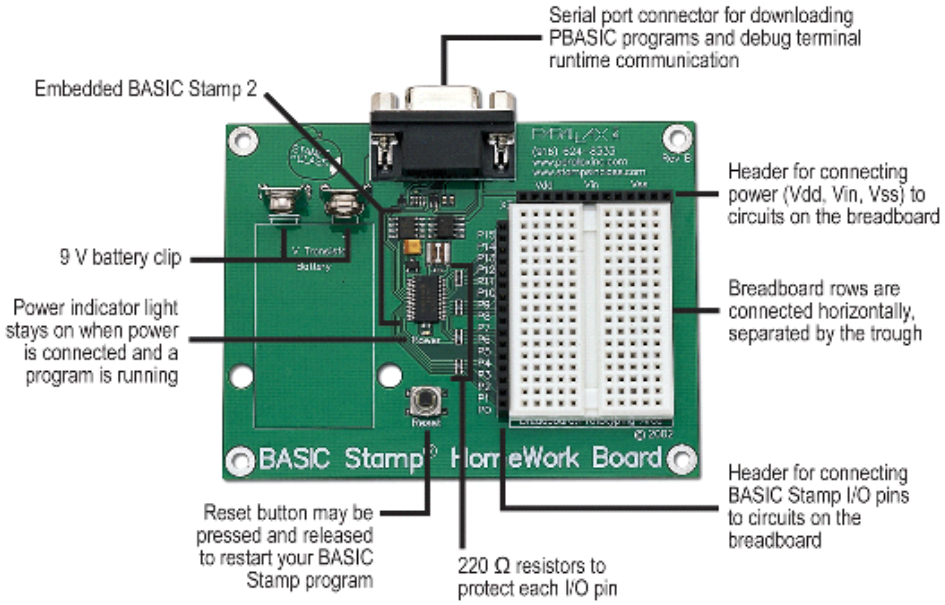


Figure B-3: BASIC Stamp® HomeWork Board™ Project Platform

The Board of Education® Rev B Carrier Board

Figure B-4 shows the Board of Education® Rev B carrier board for BASIC Stamp® 24-pin microcontroller modules. Its major components and their functions are indicated by labels.

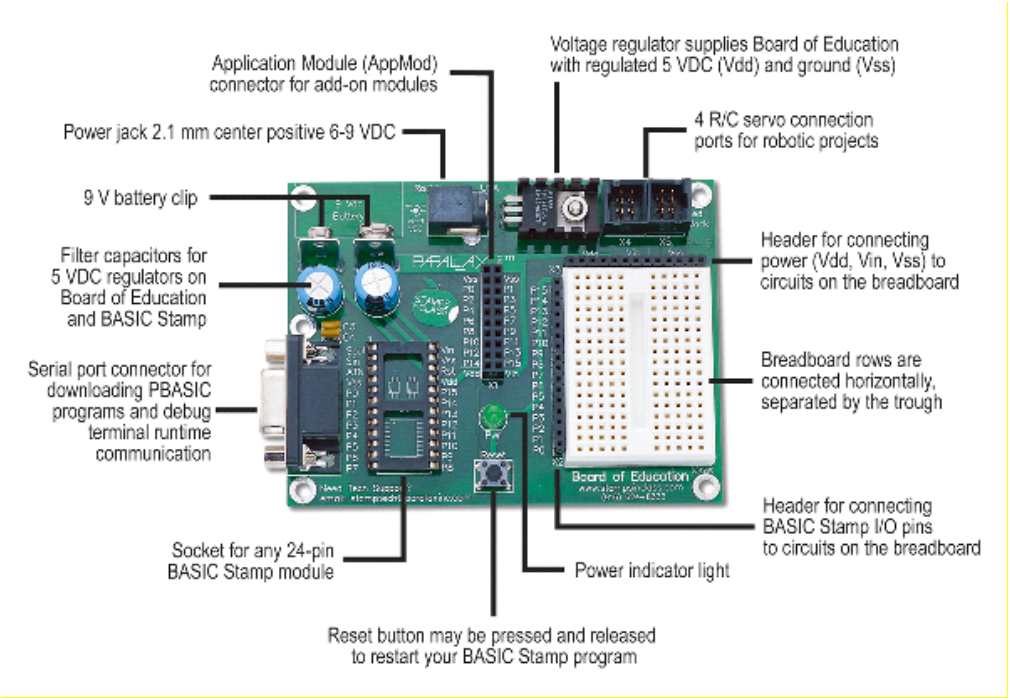


Figure B-4: Board of Education® Rev B Carrier Board

Anhang C: Widerstands - Farbcode

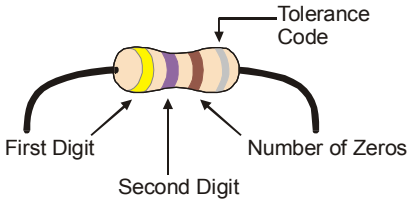
Widerstände, wie wir sie in diesem Handbuch verwenden, haben farbige Streifen zur Bezeichnung ihres Widerstandswertes. Für jeden Widerstandwert gibt es eine eigene Farbenkombination. Zum Beispiel ist der Farbencode für den 470 Ω Widerstand gelb-violett-braun.

Es ist möglich, dass noch ein vierter Streifen vorhanden ist, dieser gibt die Toleranz des Widerstandes an. Die Toleranz wird in Prozent angegeben und gibt an, wie weit der echte Widerstandwert vom angegeben entfernt sein kann. Der vierte Streifen kann golden (5%) oder silbern (10%) sein, oder gar nicht vorhanden (20%). Für die Aktivitäten mit dem BoeBot ist die Toleranz eines Widerstandes unwichtig, der Widerstandswert jedoch nicht.

Zu jedem Farbstreifen, der den Widerstandswert angibt gehört eine Zahl, diese finden Sie in Table C-1 gelistet. Figur C-1 zeigt, wie Sie mit jedem Farbstreifen und der Tabelle den Widerstandswert herausfinden.

Table C-1:
Widerstands Farbcode Werte

Wert	Farbe
0	Schwarz
1	Braun
2	Rot
3	Orange
4	Gelb
5	Grün
6	Blau
7	Violett
8	Grau
9	Weiss



Figur C-1
Widerstands-Farbcode

Hier folgt nun ein Beispiel, wie Sie mit Table C-1 und Figur C-1 den Widerstandswert berechnen, indem wir zeigen, dass gelb-violett-braun tatsächlich 470 Ω bedeutet:

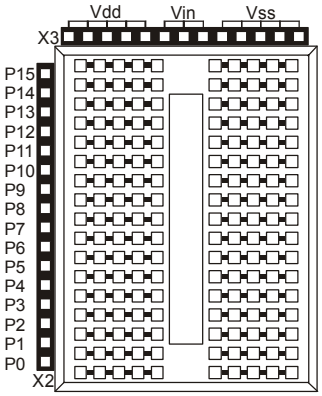
- Der erste Streifen ist gelb, dies bedeutet, dass die linkste Ziffer 4 ist.
- Der zweite Streifen ist violett, dies bedeutet die nächste Ziffer ist 7.
- Der dritte Streifen ist braun. Da braun einer 1 entspricht, bedeutet dies, dass noch eine Null hinzuzufügen ist.

Das Ergebnis: Gelb-violett-braun = 4 – 7 - 0.

Anhang D: Arbeiten mit Experimentierplatinen

Werfen Sie einen Blick auf ihr Education oder HomeWork Board. Die weissen Vierecke mit den vielen Löchern oder Vertiefungen, heissen Steckplatinen . Diese Steckplatine zusammen mit den schwarzen Streifen voll Buchsen an beiden Seiten heisst Experimentierfeld (siehe Figur D-1).

Die Beispiel-Schaltkreise in diesem Text werden aufgebaut, indem verschiedene Komponenten wie Widerstände, LEDs, Lautsprecher und Sensoren in diese kleinen Vertiefungen – Buchsen – eingesteckt werden. Die Komponenten werden durch diese Steckplatinen-Buchsen miteinander verbunden. Der Schaltkreis wird mit Elektrizität von den Stromanschlüssen versorgt, diese sind die schwarzen Buchsen an der Oberseite, angeschrieben mit Vdd, Vin, und Vss. Die schwarzen Buchsen an der linken Seite sind mit P0, P1, … bis P15 angeschrieben. Mit diesen Buchsen verbinden Sie ihren Schaltkreis mit den Input/Output-Anschlüssen der BASIC Stamp.



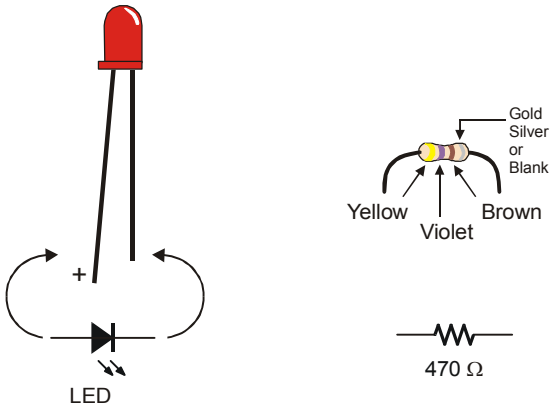
Figur D-1
Prototyping Area

Stromanschlüsse (schwarze Buchsen an der Oberseite), I/O Pin Anschlüsse (schwarze Buchsen an der Seite) und Steckplatine (weisse Buchsen)

Die Steckplatine hat 17 Reihen mit Buchsen, die von einer Mulde in zwei Spalten geteilt werden. Diese Mulde teilt jede der 17 Zeilen in zwei Zeilen mit fünf Buchsen. In jeder Zeile sind die fünf Buchsen ist im Innern der Steckplatine elektrisch miteinander verbunden. Sie können mit diesen Zeilen verschiedene Komponenten verbinden wie in einem Schaltschema. Wenn Sie zwei Drähte in zwei Buchsen der gleichen Fünfer- Zeile stecken, sind diese elektrisch miteinander verbunden.

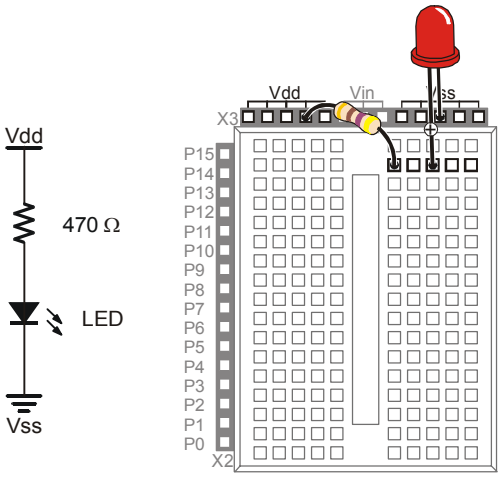
Ein Schaltschema ist ein Plan, der zeigt, wie die Komponenten miteinander verbunden werden sollen. Für jeden Komponententyp gibt es ein eigenes Symbol. Diese Komponentensymbole sind mit Linien verbunden um eine elektrische Verbindung anzuzeigen. Wenn zwei Schaltkreissymbole mit Linien miteinander verbunden sind, bedeutet diese Linie, dass dazwischen eine elektrische Verbindung besteht. Ausserdem können Linien gebraucht werden, um eine Komponente mit einer Spannungsversorgung zu verbinden. Vdd, Vin und Vss haben alle ein Symbol. Vss entspricht dem negativen Ende (0 V) der Batterieversorgung des Board of Education oder des BASIC Stamp HomeWork Board. Vin ist die positive Eingangsspannung (ca. +9V) und Vdd entspricht +5 Volt.

Nun folgt ein Beispiel, welches mit einem Schema die Bauelemente aus Figur D-2 verbindet. Für jedes dieser Bauelemente, ist die Zeichnung gleich über dem Symbol.



Figur D-2
Bauteilzeichnung und Schema - Symbols

LED(links) und
470 Ω Widerstand (rechts)



Figur D-3
Beispiel für Schema und Verdrahtungsdiagramm

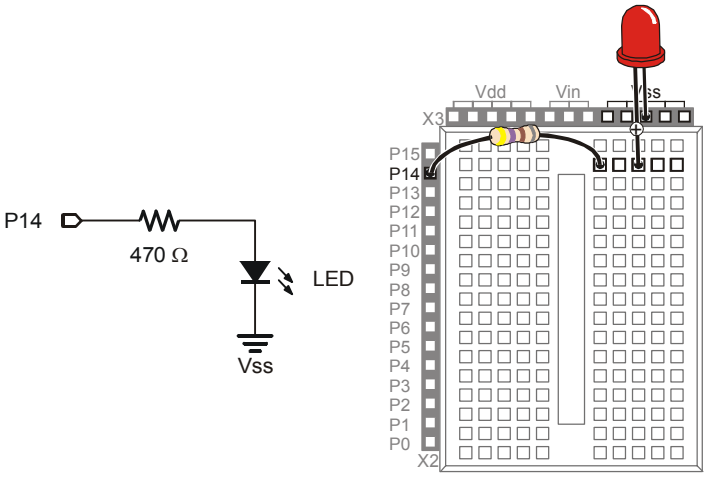
Schema (links) and Verdrahtungsdiagramm (rechts)

D

In Figur D-3 sehen Sie ein Beispiel für ein Schaltschema auf der rechten Seite, und die Zeichnung eines Schaltkreises, der mit dem Schema gebaut werden kann. Beachten Sie besonders, dass in dem Schema die Zick-Zack-Linie die einen Widerstand anzeigt, mit dem Symbol für Vdd verbunden ist. In der Zeichnung ist eine der zwei Adern des Widerstands in eine der Buchsen eingesteckt, die mit Vdd angeschrieben sind. Im Schema ist das andere Ende des Widerstandssymbols mit einer Linie mit dem +Ende des LED Symbols verbunden. Erinnern Sie sich, diese Linie zeigt an, dass die zwei Komponenten elektrisch verbunden sind. In der Zeichnung wird dies erreicht, indem der zweite Anschlussdraht des Widerstandes in die gleiche Fünfer-Reihe eingesteckt wird, wie das +Ende der LED. Dies stellt eine elektrische Verbindungen der beiden Anschlüsse her. Im Schema ist das andere Ende der LED mit dem Vss Symbol verbunden. In der Zeichnung ist der andere Anschluss der LED in eine Buchse eingesteckt, die mit Vss angeschrieben ist.

In Figur D-4 sehen Sie ein zweites Beispiel für ein Schema und ein Diagramm. Dieses Schema zeigt P14 mit einem Ende des Widerstandes verbunden, während das andere Ende mit dem +Ende der LED verbunden ist, und das –Ende der LED mit Vss verbunden ist. Der einzige Unterschied ist eine Verbindung. Der Anschluss des Widerstandes, der mit Vdd verbunden war, ist nun mit dem I/O Anschluss P14 der BASIC Stamp verbunden. Das Schema mag sehr anders aussehen, dies jedoch in erster Linie, weil der Widerstand horizontal anstatt vertikal eingezeichnet ist. Was jedoch die Verbindungen angeht, gibt es nur einen Unterschied: P14 statt Vdd. Das Verdrahtungsdiagramm zeigt,

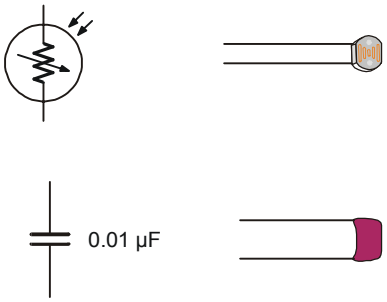
wie man mit diesem Unterschied umgeht, indem der Anschluss des Widerstands, der zuvor in Vdd eingesteckt war nun in P14 eingesteckt wird.



Figur D-4
Beispiel für Schema und Verdrahtungsdiagramm

Schema (links) and Verdrahtungsdiagramm (rechts)

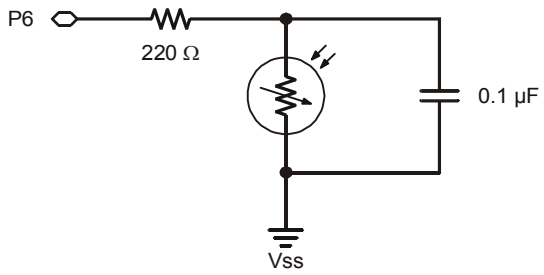
Hier ist ein komplexeres Beispiel, welches zwei weitere Bestandteile enthält, einen Fotowiderstand und einen Kondensator. Die Schaltsymbole und Zeichnungen sehen Sie in Figur D-5..



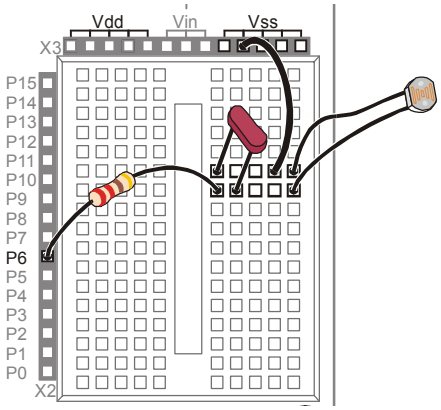
Figur D-5
Bauteilzeichnung und Schema-Symbole

Fotowiderstand (oben) und unpolarer Kondensator (unten)

Da das Schema aus Figur D-6 einen 220 Ω Widerstand verlangt, ist der erste Schritt, in der Tabelle nachzusehen, welches der Farbcode für einen 220 Ω Widerstand ist: Rot, Rot, Braun. Dieser Widerstand ist im Schema mit P6 verbunden, was bedeutet, dass der Anschluss des Widerstands in der Buchse, welche mit P6 angeschrieben ist, eingesteckt wird (Figur D-7). In diesem Schema ist das andere Ende des Widerstands nicht mit einem, sondern mit zwei weiteren Komponenten verbunden. Ein Ende des Fotowiderstands und des Kondensators teilen sich diese Verbindung. In der Steckplatine ist der andere Anschluss des Widerstands in eine der Fünfer-Zeilen eingesteckt. In derselben Zeile sind auch der Fotowiderstand und der Kondensator eingesteckt. Im Schema sind die anderen Enden des Fotowiderstands und des Kondensators mit Vss verbunden. Hier ist ein Trick, der ihnen hilft, wenn Sie Schaltkreis auf eine Steckplatine bauen. Sie können mit einem Kabel eine ganze Zeile der Steckplatine mit einer anderen Zeile, oder sogar mit einem I/O Anschluss oder einem Stromanschluss wie Vdd oder Vdd verbinden. Dann wurden die Enden des Kondensators und des Fotowiderstands in die gleiche Zeile eingesteckt um den Schaltkreis zu schliessen.

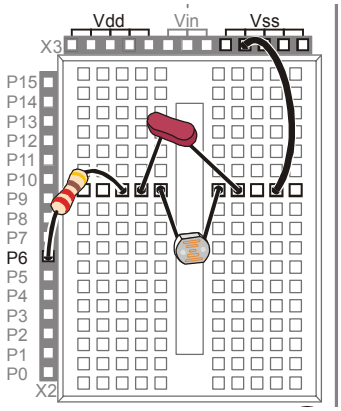


Figur D-6
Schema mit Widerstand, Fotowiderstand, und Kondensator



Figur D-7
Verdrahtungsdiagramm mit Widerstand, Fotowiderstand, und Kondensator

Vergessen Sie nicht, dass die Verkabelungsdiagramme die hier als Umsetzung der Schaltschemata gezeigt werden, nicht die einzig möglichen Lösungen darstellen. In Figur D-8 sehen Sie eine andere Lösung zu dem eben diskutierten Schema. Folgen Sie den Verbindungen und überzeugen Sie sich selbst, dass sie mit dem Schema übereinstimmen.



Figur D-8
Verdrahtungsdiagramm mit Widerstand, Fotowiderstand, und Kondensator

Beachten Sie die andere Anordnung der Bauteile.

D

Appendix E: Boe-Bot Parts Lists

To complete the activities in this text, you will need a complete Boe-Bot and the electronic components necessary to build the example circuits. There are several options for ordering these items from Parallax, which are described on the following pages.

All of the information in this appendix was current at the time of printing. Parallax may make part substitutions at our discretion, out of necessity or to upgrade the quality of our products. For the latest information and free downloads about your Boe-Bot and the Robotics with the Boe-Bot Student Guide, check their individual product pages at www.parallax.com.

Boe-Bot Robot Kit (also known as the Boe-Bot Full Kit)

Aside from a PC with a serial port and a few common household items, the Boe-Bot Robot Kit contains all the parts and documentation you'll need to complete the experiments in this text.

Table E-1: Boe-Bot Full Kit (#28132)
Parts and quantities subject to change without notice

Parallax Stock Code	Description	Quantity
BS2-IC	BASIC Stamp 2 microcontroller module	1
27000	Parallax CD with software and documentation	1
27218	BASIC Stamp Programming Manual Version 2.0	1
28124	Robotics with the Boe-Bot Parts Kit	1
28125	Robotics with the Boe-Bot Student Guide Version 2.0	1
28150	Board of Education Rev C	1
700-00064	Parallax Screwdriver	1
800-00003	Serial cable	1

All of these items may also be ordered separately, using the individual part numbers. You can contact the Parallax Sales Team toll free at 1-888-512-1024 or order online at www.parallax.com. For technical questions or assistance call our Technical Support team at 1-888-99-STAMP.

Robotics with the Boe Bot Parts Kit

If you already have a Board of Education and BASIC Stamp, you may purchase the Robotics with the Boe-Bot Parts kit, with or without the printed text *Robotics with the Boe-Bot Student Guide v2.0.*

Table E-2: Robotics with the Boe-Bot Parts & Text, #28154
Robotics with the Boe-Bot Parts only, #28124
Parts and quantities subject to change without notice

Parallax Stock Code	Description	Quantity
150-01020	1 kΩ resistor	2
150-01030	10 kΩ resistor	2
150-02020	2 kΩ resistor	2
150-02210	220 Ω resistor	8
150-04710	470 Ω resistor	4
150-04720	4.7 kΩ resistor	2
200-01031	0.01 µF capacitor	2
200-01040	0.1 µF capacitor	2
350-00003	Infrared LED	2
350-00006	Red LED	2
350-00009	Photoresistors (EG&G Vactec VT935G group B)	2
350-00014	Infrared receiver (Panasonic PNA4602M or equivalent)	2
350-90000	LED standoff for infrared LED	2
350-00001	LED light shield for infrared LED	2
451-00303	3-Pin Header	2
700-00056	Whisker wire	2
700-00015	#4 screw-size nylon washer	2
710-00007	7/8” 4-40 pan-head screw, Phillips	2
713-00007	½” Spacer, aluminum, #4 round	2
800-00016	Jumper wires (bag of 10)	2
900-00001	Piezospeaker	1
28133	Boe-Bot Hardware Pack	1

Building a Boe-Bot with a HomeWork Board

If you already have a BASIC Stamp HomeWork Board that you wish to use with a Boe Bot, you will need the Robotics with the Boe-Bot Parts kit **and these additional items:**

(2) 3-pin male/male headers, #451-00303
(1) Tinned-lead battery pack, #753-00001

Boe-Bot Hardware Pack

All of the Boe-Bot hardware parts can be purchased individually, as found in our on-line Robot Component Shop if you find you need a replacement part. Please note that the Hardware Pack is not sold as a unit separately from the Boe-Bot Robot (Full) Kit or the Boe-Bot Parts Kit.

Table E-3: Boe-Bot Hardware Pack (#28133)
Parts and quantities subject to change without notice

Parallax Stock Code	Description	Quantity
700-00002	4-40 x 3/8”machine screw, Phillips	8
700-00003	Hex nut, 4-40 zinc plated	10
700-00009	1" polyethylene ball, pre-drilled	1
700-00016	4-40 x 3/8” flathead machine screw, Phillips	2
700-00022	Boe-Bot aluminum chassis	1
700-00023	1/16" x 1.5” long cotter pin	1
700-00025	13/32" rubber grommet	2
700-00028	4-40 x 1/ 4” machine screw, Phillips	8
700-00038	Battery holder with cable and barrel plug	1
700-00060	Standoff, threaded aluminum, round 4-40	4
721-00001	Parallax plastic wheel	2
721-00002	Rubber band tire	4
900-00008	Parallax Continuous Rotation Servo	2

Board of Education Kits

Almost all of the titles in the Stamps in Class curriculum feature different hardware and component packages that depend on the BASIC Stamp and Board of Education as a core. The Board of Education can be purchased separately or in its own kit, as listed in Table E-4 below.

Table E-4: Board of Education – Full Kit (#28102)
Parts and quantities subject to change without notice

Parallax Stock Code	Description	Quantity
28150	Board of Education Rev C	1
800-00016	Jumper wires – pack of 10	1
BS2-IC	BASIC Stamp 2 microcontroller module	1
750-00008	300 mA 9 VDC power supply	1
800-00003	Serial cable	1

Anhang F: Abgleich der Fotowiderstände

In diesem Anhang, überprüfen Sie, ob die Photowiderstände auf die gleiche Umgebungshelligkeit ähnlich reagieren. Wenn Sie unterschiedliche Messungen bei der gleichen Umgebungshelligkeit erhalten, können Sie ihr Programm anpassen, so dass die Messungen der Photowiderstände unterschiedlich gewichtet werden. Die Messungen werden dann für ähnliche Umgebungshelligkeit ähnlich sein und ihr Boe-Bot erkennt verschiedene Lichtstärken besser. Diese Technik hilft dann ihrem Boe-Bot, dunkle Räume zu verlassen, sogar mit nicht aufeinander abgestimmten Fotowiderstands-Schaltkreisen.

RC-Schaltkreise mit Fotowiderständen können aus vielen Gründen verschiedene Messungen für die gleiche Lichtstärke ergeben: Der angegebene Wert des Kondensators ist 0.01 µF, aber der echte Wert kann davon abweichen. Viele der gewöhnlichen Keramik-Kondensatoren haben eine Toleranz von +80%/-20%. Dies bedeutet, dass der eigentliche Wert bis zu 80% grösser oder 20% kleiner sein kann, als die angegebenen 0.01 µF. Dies bedeutet, dass die gemessene Abfallzeit also zwischen 80% grösser und 20% kleiner sein kann. Die Photowiderstände selbst verhalten sich unterschiedlich, wenn Sie aus verschiedenen Produktionsserien stammen, oder die Sensorflächen verschmiert oder abgesplittert sind.

Test für gut gepaarte Fotowiderstands-Schaltkreise

Das folgende Beispielprogramm zeigt die Abklingzeit beider Fotowiderstände im Debug Fenster. Dies vereinfacht eine Beurteilung der Differenz zwischen zwei Messungen bei gleichem Lichtpegel.

Um die besten Ergebnisse zu erhalten vermeiden Sie direktes Sonnenlicht: Im allgemeinen erhalten Sie bei gleichmässiger Beleuchtung bessere Resultate mit den Fotowiderständen des Boe-Bot. Ziehen Sie die Vorhänge um direktes Sonnenlicht zu vermeiden. Räume mit mehreren verteilten Lichtquellen, wie Leuchtstoffröhren oder Deckenlampen, funktionieren gut.

Beispielprogramm: TestPhotoresistors.bs2

√ Erfassen, speichern und starten Sie: TestPhotoresistors.bs2.

- √ Werfen Sie mit einem Papier einen Schatten auf die Photowiderstände des Boe-Bots. Erzeugen Sie soviel Schatten, dass die Messungen zwischen 20 und 100 liegen.
- √ Notieren Sie die Ergebnisse beider Messungen in der ersten Zeile von Table **F-1**.
- √ Werfen Sie mit ihrer Hand gleichviel Schatten auf beide Photowiderstände. Idealerweise sollten die Messungen zwischen 200 und 400 liegen.
- √ Notieren Sie die Ergebnisse beider Messungen in der zweite Zeile von Table F-1.

Table F-1: RC-Zeitmessungen in schwachem und Umgebungslicht

Duration Werte		Beschreibung
timeLeft	**timeRight**	
		Fotowiderstände bei gleichmässigem Umgebungslicht
		Fotowiderstände bei gleichmässig schwachem Licht

```
' Robotics with the Boe-Bot - TestPhotoresistors.bs2
' Test Boe-Bot photoresistor circuits.

' {$STAMP BS2}                                 ' Stamp directive.
' {$PBASIC 2.5}                                ' PBASIC directive.

timeLeft      VAR     Word                    ' Variable declarations.
timeRight     VAR     Word

DEBUG "PHOTORESISTOR VALUES", CR,             ' Initialization.
      "timeLeft  timeRight", CR,
      "--------  ---------"

DO                                            ' Main routine.

  HIGH 6                                      ' Left RC time measurement.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                      ' Right RC time measurement.
  PAUSE 3
  RCTIME 3,1,timeRight

  DEBUG CRSRXY, 0, 3,                         ' Display measurements.
        DEC5 timeLeft,
        "     ",
        DEC5 timeRight

  PAUSE 200
```

```
LOOP
```

Kalibrierung mit Linearer Approximation

Der Fotowiderstand wird oft als nichtlineares Bauteil bezeichnet. Mit anderen Worten, wenn er bei einer Helligkeit einen Messwert liefert, bedeutet das nicht, dass er in fünfmal hellerem Licht auch einen fünfmal grösseren Messwert liefert. Die Mathematik dahinter ist komplizierter und enthält Logarithmen. Wenn allerdings nur ein kleiner Ausschnitt aus dem gesamten Messbereich des Sensors verwendet wird, kann der Sensor wie ein lineares Bauteil behandelt werden. Sie können ein paar Test-Messungen machen, und dann ausrechnen welche Ergebnisse der Sensor liefern müsste, wenn die übrigen Messergebnisse im Messbereich mit einer Geraden verbunden werden könnten. Diese Methode nennt man die Lineare Approximation .

Sie können ferner bei linearen Bauteilen auch die Differenz zwischen zwei Geraden als eine Gerade zeichnen. Wenn Sie also einen linearen Sensor haben, das grössere Werte für normales und schwaches Licht liefert als der andere, so können Sie die Lineare Approximation anwenden, damit beide Sensoren ungefähr gleiche Werte für gleiche Lichtstärken liefern.

Jeden Messwert des einen Sensors (bezeichnen wir sie als x) können wir mit einem Proportionalitätsfaktor (m) multiplizieren und dann zu einer Konstanten (b) addieren, um ungefähr den Wert zu erhalten, den der andere Sensor geliefert hätte (y).

$$y = mx + b$$

Hier ist ein Beispiel dafür, wie Sie die Werte von *m* und *b* bekommen, um den linken Fotowiderstandschaltkreis auf den rechten abzustimmen. Zuerst nehmen Sie X_1 und X_2 für die Werte des linken Fotowiderstandes und Y_1 and Y_2 für die Werte des rechten. Table F-2 zeigt ein paar Beispielwerte für nicht zusammenpassende Photowiderstände. Sie werden andere Werte erhalten.

Table F-2: RC-Zeitmessungen in schwachem und Umgebungslicht

Duration Werte		**Beschreibung**
timeLeft	**timeRight**	
$X_1 = 36$	$Y_1 = 56$	Fotowiderstände bei gleichmässigem Umgebungslicht
$X_2 = 152$	$Y_2 = 215$	Fotowiderstände bei gleichmässig schwachem Licht

Nun, lösen Sie nach *m* und *b* auf mit zwei Gleichungen mit zwei Unbekannten. Einer der einfacheren Lösungswege ist, indem Sie zwei Gleichungen $y = mx + b$ notieren, eine mit den Werten von X_1 und Y_1 und die andere mit den Werten von X_2 und Y_2. Subtrahieren Sie danach die eine von der anderen, um *b* zu eliminieren. Lösen Sie sie danach nach *m* auf.

$$\begin{aligned} & y_2 = mx_2 + b \\ -(& y_1 = mx_1 + b) \\ & \text{-----------} \\ & (y_2 - y_1) = m(x_2 - x_1) \end{aligned}$$

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

Wenn Sie einmal *m* haben, können Sie *m* in eine der beiden Gleichungen einsetzen und nach *b* auflösen.

$$\begin{aligned} y_2 &= mx_2 + b \\ b &= y_2 - mx_2 \end{aligned}$$

Die Gleichungen für *b* und *m* sind also:

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad \text{and} \quad b = y_2 - mx_2$$

Wenn wir nun die Beispielwerte aus Table F-2 in die Gleichungen einsetzen, bekommen wir den Proportionalitätsfaktor (*m*) und die Verschiebungskonstante (*b*) des linken Fotowiderstandes. Zuerst berechnen wir m:

$$\begin{aligned} m &= \frac{(y_2 - y_1)}{(x_2 - x_1)} \\ m &= \frac{(215 - 56)}{(152 - 36)} \\ m &= \frac{159}{116} = 1.37 \end{aligned}$$

Berechnen Sie danach *b* mit m, y_2, und x_2:

$$b = 215 - (1.37 \times 152)$$
$$b = 6.76$$
$$b \approx 7$$

Nun wissen wir, wie wir die **`timeLeft`** Variable korrigieren müssen, damit sie innerhalb dieses kleinen Lichtstärken-Bereichs ähnliche Messwerte liefert wie die **`timeRight`** Variable:

$$y = mx + b$$
$$y = 1.37x + 7$$
$$timeLeft_{(adjusted)} = 1.37 \times timeLeft + 7$$

Eine Lineare Gleichung in PBASIC

In den meisten Programmiersprachen für PCs könnte diese Gleichung genau so eingegeben werden. Die BASIC Stamp ist jedoch ein sehr kleiner Prozessor verglichen mit einem PC. Deshalb braucht es einen zusätzlichen Schritt um mit einem Bruch zu multiplizieren. Dazu müssen Sie den */ Operator benützen (er heisst "Star-Slash" Operator). Für die **`timeLeft`** Gleichung kann der PBASIC Code wie folgt aufgebaut werden, um die **`timeLeft`** Variable anzupassen:

```
timeLeft = (timeLeft */ 351) + 7
```

Der angepasste **`timeLeft`** Wert nach dieser Code-Zeile ist 1.37 mal der alte **`timeLeft`** Wert plus 7.

Wieso wurde 1.37 aus 351? Der */ Operator funktioniert wie folgt: Er multipliziert ihren Bruch mit 256 und platziert ihn rechts des */ Operator. Da 1.37 X 256 = 350.72 ≈ 351, erscheint der Wert 351 auf die rechte Seite des */ Operator.

You can find out more about the */ operator in the BASIC Stamp Editor by clicking Help and selecting Index. Type in */ in the field labeled “Type in keyword to find”. You can also look up */ in the Binary operators section of the *BASIC Stamp Manual*.

Your Turn – Balance Your Photoresistors with m and b
Sie sind dran – Gleichen Sie die Fotowiderstände mit m und b ab.

- √ Schreiben Sie in Table F-1 ihre erste **`timeLeft`** Eingabe mit X1 und die zweite **`timeLeft`** Eingabe mit X2 an.
- √ Schreiben Sie die erste **`timeRight`** Eingabe mit Y1 und die zweite **`timeRight`** Eingabe mit Y2 an.
- √ Benützen Sie diese Gleichungen und die Werte für X1, X2, Y1 und Y2 um m und b nach folgender Formel zu berechnen:
- √

$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad \textbf{and} \quad b = y_2 - mx_2$$

- √ Um m zu berechnen, benützen Sie den */ Operator indem Sie m mit 256 multiplizieren.
- √ Ersetzen Sie ihre m und b Werte in der folgenden Codezeile von BalancePhotoresistors.bs2:

```
timeLeft = (timeLeft */ 351) + 7
```

- √ Erfassen, speichern, und starten Sie ihre Version von BalancePhotoresistors.bs2.
- √ Sorgen Sie, dass beide Fotowiderstände mit derselben Lichtstärke beleuchtet werden.
- √ Kontrollieren Sie, ob die “after” Werte vergleichbar sind und prüfen Sie auf Differenzen in den “before” Werten.
- √ Wählen Sie für beide Fotowiderstände eine andere Lichtstärke.
- √ Überprüfen Sie die “after” Werte nochmals auf Vergleichbarkeit.

```
' Robotics with the Boe-Bot - BalancePhotoresistors.bs2
' Test adjustments to Boe-Bot photoresistor circuits.

' {$STAMP BS2}                                 ' Stamp directive.
' {$PBASIC 2.5}                                ' PBASIC directive.

timeLeft      VAR     Word                    ' Variable declarations.
timeRight     VAR     Word

DEBUG "PHOTORESISTOR VALUES", CR,             ' Initialization.
      "timeLeft  timeRight", CR,
      "--------  ---------"

DO                                            ' Main routine.
```

```
  HIGH 6                                        ' Left RC time measurement.
  PAUSE 3
  RCTIME 6,1,timeLeft

  HIGH 3                                        ' Right RC time measurement.
  PAUSE 3
  RCTIME 3,1,timeRight


  DEBUG CRSRXY, 0, 3,                           ' Display measurements.
        DEC5 timeLeft,
        "     ",
        DEC5 timeRight,
        "   Before"

  timeLeft = (timeLeft */ 351) + 7

  DEBUG CRSRXY, 0, 5,                           ' Display measurements.
        DEC5 timeLeft,
        "     ",
        DEC5 timeRight,
        "   After"

  PAUSE 200

LOOP
```

Anhang G: Tuning der IR Distanzerkennung

Wie man die richtigen Werte für den Frequenz-Sweep findet

Für das Feintuning der Distanz-Erkennung des Boe-Bot muss man herausfinden, welche Frequenz für jede Distanzzone und für jedes IR-Paar am zuverlässigsten funktioniert.

Hinweis: In diesem Anhang wird eine Methode zur Bestimmung der besten Frequenz für eine bestimmte Distanz gezeigt, bei welcher Rechenblätter eingesetzt werden. Dafür braucht es Zeit und Geduld. Deshalb sollten Sie dies nur tun, wenn ihre Distanz-Sensoren deutliche Abweichungen zeigen. Die Frequenz-Sweep Daten werden gesammelt und dazu verwendet, die verlässlichsten Werte zu suchen, um bestimmte Distanzen zu erkennen.

- √ Drehen Sie die IR LEDs und die Detektoren so, dass sie gerade nach vorne zielen.
- √ Stellen Sie den Boe-Bot vor eine Wand mit einem Blatt weissem Papier als IR Ziel.
- √ Platzieren Sie den Boe-Bot so, dass seine IR LEDs 2.5 cm vom Papierziel entfernt sind. Versichern Sie sich, dass der Boe-Bot gerade vor dem Papierziel steht. Die IR LEDs und Detektoren sollten direkt auf das Papier zielen.

IR Feintuning-Programm

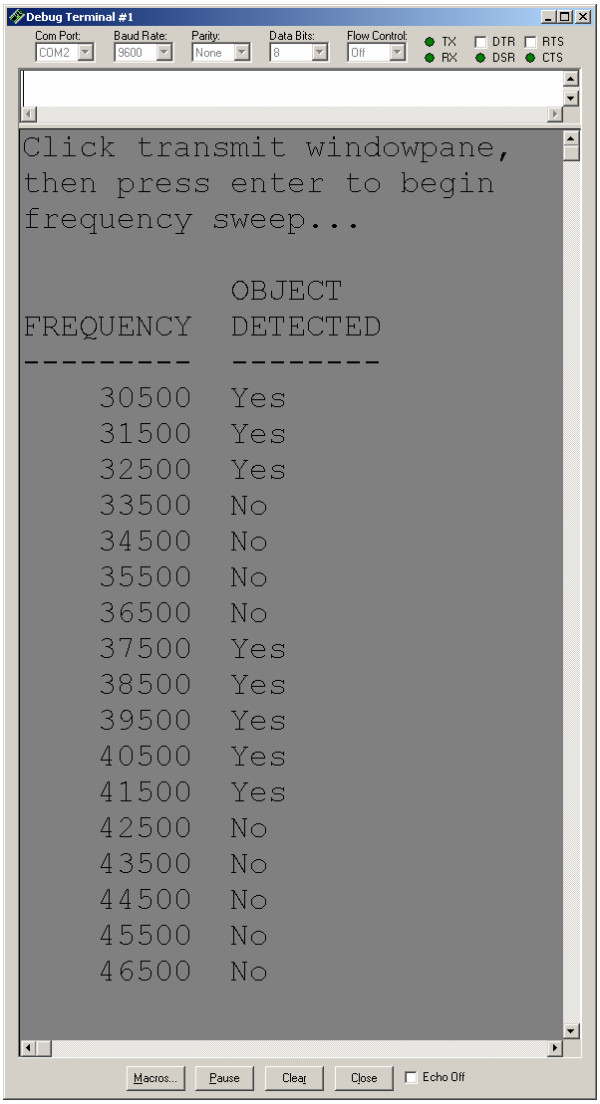
FrequencySweep.bs2 führt einen Frequenz-Sweep der IR Detektoren durch und zeigt die Daten an. Obwohl die Techniken ähnlich sind wie bei anderen Programmen, hat es eine neuartige Eigenschaft. Die BASIC Stamp wartet, bis Sie Enter gedrückt haben.

- √ Geben Sie FrequencySweep.bs2 ein und lassen Sie es laufen, aber lassen Sie das serielle Kabel des Boe-Bot eingesteckt.

```
' -----[ Title ]-------------------------------------------------------------
' Robotics with the Boe-Bot - FrequencySweep.bs2
' Test IR LED/detector response to frequency sweep.

' {$STAMP BS2}                              ' Stamp directive.
' {$PBASIC 2.5}                             ' PBASIC directive.

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

crsrPosRow     VAR     Byte
irFrequency    VAR     Word
irDetect       VAR     Bit
```

```
distance       VAR     Nib
dummy          VAR     crsrPosRow

' -----[ Initialization ]----------------------------------------------------

DEBUG CLS,
      "Click transmit windowpane,", CR,
      "then press enter to begin", CR,
      "frequency sweep...", CR, CR,

      "           OBJECT", CR,
      "FREQUENCY  DETECTED", CR,
      "---------  --------", CR

' -----[ Main Routine ]-------------------------------------------------------

DO

  DEBUGIN dummy

  crsrPosRow = 6

  FOR irFrequency = 30500 TO 46500 STEP 1000

    crsrPosRow = crsrPosRow + 1

    FREQOUT 8,1, irFrequency
    irDetect = IN9

    DEBUG CRSRXY, 4, crsrPosRow, DEC5 irFrequency
    DEBUG CRSRXY, 11, crsrPosRow

    IF (irDetect = 0) THEN DEBUG "Yes" ELSE DEBUG "No "

    PAUSE 100

  NEXT

LOOP
```

- √ Klicken Sie das obere der beiden Fenster aus Figur G-1 an.
- √ Drücken Sie Enter. Die Frequenz und Anwort-Daten werden wie im Bild gezeigt erscheinen.



Figur G-1
Debug Fenster der Frequenz-Daten

Die BASIC Stamp wurde so programmiert, dass sie im Debug Terminal ein “Yes” zeigt, wenn ein Objekt gefunden wurde, und ein “No” wenn nicht. Figur G-1 zeigt, dass die Spanne des guten Antwortverhaltens des linken Sensors zwischen 36500 und 42500 liegt.

- √ Verändern Sie die **FOR...NEXT** Schleife im Programm FrequencySweep.bs2 so, dass es in 250er Schritten steigt und die obere und untere Limite beider Detektoren enthält. Basierend auf den Daten Figur G-1 werden die Start-, Ende- und Schritt-Werte der **FOR...NEXT** Schlaufe wie folgt modifiziert:

  ```
  FOR irFrequency = 36500 to 42500 STEP 250
  ```

- √ Lassen Sie ihr verändertes FrequencySweep.bs2 laufen und drücken Sie Enter.
- √ Kopieren Sie die Daten für links und rechts in ein separates Rechenblatt.
- √ Drücken Sie Enter und nehmen Sie das nächste Datenset auf.
- √ Wiederholen Sie diesen Prozess noch drei mal. Am Ende werden Sie fünf Datensets für diese eine Frequenz in verschiedenen Rechenblättern haben.
- √ Rücken Sie ihren Boe-Bot 2.5 cm zurück. Nun sind IR Detektoren 5 cm vom Papier entfernt.
- √ Schreiben Sie nochmals fünf Datensets für diese Distanz auf.
- √ Rücken Sie den Boe-Bot jedes Mal weitere 2.5 cm zurück und schreiben Sie jedes Mal wieder fünf Datensets auf.
- √ Wenn der Boe-Bot 20 cm zurückgerückt ist wird der Frequenz-Sweep meistens, wenn nicht sogar immer ”No” anzeigen. Wenn der Frequenz-Sweep immer ”No” anzeigt, bedeutet dies, dass der Boe-Bot mit keiner Frequenz des Sweeps auf diese Distanz irgendein Objekt erkennen kann.

Bei sorgfältiger Durchsicht der Rechenblätter und durch fortlaufende Elimination können Sie die optimale Frequenz für jedes IR Paar für jede Zone bestimmen. Anpassungen an bis zu 8 Zonen kann man ohne eine Restrukturierung der Navigationsroutine des Boe-Bots vornehmen. Wenn Sie 15 Zonen unterscheiden wollten, würde das 30 Ein-Millisekunden **FREQOUT** Befehle benötigen. Dies passt nicht zwischen die Impulse der Servos. Eine Möglichkeit wäre, je 15 Messungen bei jedem zweiten Impuls zu machen.

Hier wurde gezeigt, wie Sie die besten Frequenzen für den linken Sensor bestimmen können, Denken Sie daran, dass Sie den Prozess für den rechten Sensor wiederholen müssen. In diesem Beispiel werden sechs Zonen unterschieden (null bis fünf).

- √ Beginnen Sie damit, die mit dem grössten Abstand aufgenommenen Datenpunkte durchzusehen. Dabei gibt es wohl kaum ein Datenset, bei dem alle ein “Yes” bei der selben Frequenz haben. Prüfen Sie die Daten bei 2.5 cm näher am Papier. Sie werden voraussichtlich ein Set sehen, dass vier oder fünf “Yes”

bei einer bestimmten Frequenz hat. Benützen Sie diese Frequenz als verlässliche Trennung zwischen Zone Null und Zone 1.

√ Bei jeder der fünf verbleibenden Distanzen suchen Sie eine Frequenz bei der die Werte gerade stabil wurden.

Ein Beispiel: Nehmen wir an, bei 15 cm Abstand haben drei verschieden Frequenzen fünf ”Yes” . Wenn Sie auf die 17.5 cm Marke zurückschauen, waren zwei dieser Frequenzen stabil, aber eine nicht. Nehmen Sie die Frequenz die bei 17.5 cm nicht stabil, aber bei 15 cm schon stabil war als ihre beste Frequenz für diese Distanz. Dieses Beispiel hat gezeigt, welche Frequenzen benutzt werden können, um die Zonen 5 und 4 und die Zonen 4 und 3 zu trennen. Wiederholen Sie diesen Prozess für die weitere Zonenunterteilung.

Sie sind dran

√ Wenn Sie das Feintuning mit fünf Messungen erfolgreich abgeschlossen haben, und Sie noch Zeit haben, versuchen Sie es mit acht Messungen. Speichern Sie auf jeden Fall ihre Daten für beide Methoden.

Anhang H: Boe-Bot Navigationswettbewerb

Wenn Sie einen Wettkampf für selbstgesteuerte Robots planen, können Sie auf die folgenden Regeln zurückgreifen, die uns die Seattle Robotics Society freundlicherweise zur Verfügung gestellt hat.

WETTBEWERB 1: BODENTURNEN FÜR ROBOTS

Zweck

Der Wettbewerb im Bodenturnen soll den Robot-Erfindern Gelegenheit geben, ihre Robots oder andere Apparate vorzuführen.

Regeln

Die Regeln für diesen Wettbewerb sind ziemlich einfach: Eine 3 x 3 Meter grosse Fläche wird bestimmt, vorzugsweise mit einer stabilen Begrenzung. Jeder Teilnehmer hat maximal 5 Minuten Zeit um vorzuführen, was sein Robot alles kann. Der Vorführende kann die verschiedenen Eigenschaften und Fähigkeiten dabei kommentieren. Wie immer sind Robots, die das Wettkampfgelände beschädigen könnten oder eine Gefahr für die Zuschauer darstellen, nicht zugelassen. Die Robots müssen nicht notwendigerweise selbststeuernd sein, aber es wird gefördert. Die Bewertung erfolgt durch das Publikum, entweder durch Klatschen (das lauteste nach Einschätzung des Wettkampfleiters) oder durch sonst ein Abstimmverfahren.

WETTBEWERB 1: LINIENFOLGER

Ziel

Ziel ist, einen selbststeuernden Robot zu bauen, der im Bereich “A” (auf Position “S”) startet, vollständig der Linie folgend zum Bereich “B” fährt, und dann zum Bereich “A” zurückkehrt (auf Position “F”). Der Robot, der dafür am wenigsten Zeit braucht (inklusive Boni) gewinnt. Der Robot muss dabei die Bereiche “B” und “C” besuchen. Die genaue Anordnung der Strecke bleibt bis zum Wettkampftag geheim. Sie weist aber die oben beschriebenen drei Bereiche auf.

Getestete Fähigkeiten

Getestet wird die Fähigkeit, eine Navigationshilfe (die Linie) zu erkennen und zur Zielerreichung zu benutzen.

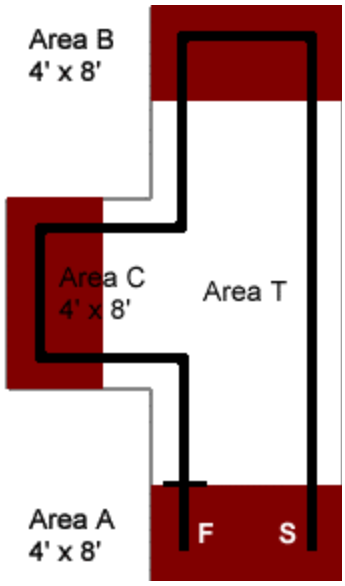
Zeitlimit

Der Parcours muss in maximal vier Minuten absolviert werden.

Beispielparcours

Alle Massangaben im Beispielparcours sind als Richtwerte zu verstehen. Eine durchgezogene Linie trennt Bereich “A” vom Bereich “T” bei der Position “F”. Hier endet der Parcours. Die Linie ist schwarz, ca. 6 cm breit und hat einen Abstand von rund 50 cm von den Wänden. Alle Kurven haben einen Radius von wenigstens 30 cm und maximal 1 m. Die Wände sind etwa 10 cm hoch und begrenzen die Teststrecke. Der Boden ist weiss und besteht aus Papier oder Tyvek®. Tyvek von Dupont ist ein unverwüstliches PE-Vlies, das auch für Versandtaschen, Schutzkleidung und im Hausbau eingesetzt wird.

Die Positionen "S" und "F" dienen nur der Illustration und sind keine genau festgelegten Punkte. Der Wettbewerbsteilnehmer kann für den Start seinen Robot im Bereich “A” in beliebiger Position und Ausrichtung aufstellen. Der Robot muss vollständig innerhalb des Bereiches “A” stehen. Im echten Parcours sind die Flächen "A," "B" and "C" ***nicht*** rot eingefärbt.



Figur H-1
Beispiel Wettkampf-Parcours

Punkte

Das Punktetotal für jeden Wettbewerbsteilnehmer wird berechnet aus der benötigten Fahrzeit (in Sekunden), abzüglich 10% für jede erfüllte Aufgabe. Der Teilnehmer mit der tiefsten Punktezahl gewinnt.

Table H-1: Punkte Linienfolger

Aufgabe erfüllt	Zeitgutschrift
Stoppt im Bereich A nach dem Besuch in Bereich B und C	10%
Hat keine Wand berührt	10%
Startet auf Befehl	10%

("Startet auf Befehl" bedeutet, der Robot startet auf einen externen, berührungslosen Befehl, wie z.B. einem optischen oder akustischen Signal.)

WETTBEWERB 2: LABYRINTH ABFAHREN

Zweck

Das grosse Labyrinth testet die Navigationsfähigkeiten eines selbststeuernden Robots. Die Punktewertung bevorzugt Robots, die entweder brutal schnell sind, oder die sich den Weg im ersten Durchlauf merken können. Das Ziel für den Robot, der an den Eingang des Labyrinths gesetzt wird, ist es, den Weg durch das Labyrinth zu finden und den Ausgang in der kürzestmöglichen Zeit zu finden

Technische Kenndaten

Das Labyrinth wird aus handelsüblichem 20mm Sperrholz hergestellt. Die Wände sind ca. 75 cm hoch und mit Hochglanzlack in den Grundfarben lackiert. Die Wände sind in einem 75cm-Raster montiert. Wegen der Wanddicke und Ungenauigkeiten in der Konstruktion können die Durchgänge bis zu 65 cm schmal werden. Das Labyrinth kann insgesamt - abhängig vom verfügbaren Platz - bis zu 6 x 6 m gross sein.

Das Labyrinth wird auf einem Teppich- Turnhallen- oder Betonboden (je nach Veranstaltungsort) aufgestellt. Es wird überdacht, so dass die teilnehmenden Robots nicht regenfest sein müssen. Allerdings sind die Robots dem Wind, Temperaturänderungen und wechselnden Lichtverhältnissen ausgesetzt. Die Wettkampfstrecke ist als echtes zweidimensionales Labyrinth (nicht als Irrgarten) konstruiert: Es gibt einen einzigen richtigen Weg vom Start bis zum Ziel, und es gibt keine Inseln im Labyrinth. Sowohl Eingang wie Ausgang befinden sich aussen. Ein solches echtes Labyrinth kann erfolgreich gelöst werden, indem man im Kontakt mit entweder der linken oder der rechten Wand bleibt. Das Wettkampf-Labyrinth ist sorgfältig so konstruiert, dass es weder von Vor- noch von Nachteil ist, ob man sich an der linken oder rechten Wand orientiert.

Einschränkungen der Robots

Die wichtigste Einschränkung für den Robot ist, dass er unabhängig sein muss: Einmal gestartet, sind keine weiteren Einflussnahmen des Eigentümers oder Bedieners mehr erlaubt, bis der Robot den Ausgang verlässt oder hoffnungslos festsitzt. Offensichtlich muss der Robot ferner klein genug sein, dass er zwischen die Wände des Labyrinths passt. Der Robot darf die Wände berühren, darf aber die Wände nicht zu seinem Vorteil verschieben – keine Bulldozer! Die Wettbewerbsleitung kann Robots disqualifizieren, die übermässig scheinende Verschiebungen der Wände vornehmen. Der Robot darf weder die Wände noch den Boden des Labyrinths beschädigen. Jede Form der

Energieversorgung ist zulässig, solange die örtlichen Vorschriften für den Betrieb weder Gehörschutz nach andere Restriktionen vorschreiben.

Bewertung

Jeder Robot muss das Labyrinth drei mal durchqueren. Der Robot mit der kürzesten Einzelzeit ist der Sieger.. Die Maximalzeit pro Durchlauf beträgt 10 Minuten. Wenn ein Robot in dieser Zeit die Aufgabe nicht bewältigen kann, wird der Lauf abgebrochen und die Zeit mit 10 Minuten verbucht. Wenn kein Robot den Ausweg aus dem Labyrinth findet, wird jener Robot Sieger, der nach Einschätzung der Wettkampfleitung am weitesten gekommen ist.

Logistik

Jeder Robot hat einen Durchlauf, solange bis alle Robots einen Versuch hatten. Dann startet jeder Robot in derselben Reihenfolge zum zweiten Versuch, schliesslich zum dritten. Der Wettkampfleiter kann nach Ermessen Änderungen vornehmen, wenn ein Teilnehmer wegen technischer Probleme seinen Versuch verschieben muss. Ein Robot darf sich an die Strecke von früheren Durchläufen erinnern, um seine Zeit zu verbessern. Das heisst er kann das Labyrinth beim ersten Durchlauf kartographierein, und diese Informationen bei späteren Versuchen verwenden, solange der Robot dies selbst macht. Es ist nicht zulässig, den Robot manuell auf die Anordnung des Labyrinths zu “konfigurieren”, weder hardware- noch softwaremässig.

Index

Go to the head of the class!

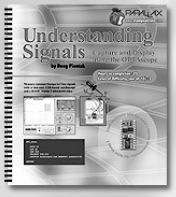
Explore microcontroller programming and interfacing with any of our popular Stamps in Class Curricula.

What's a Microcontroller? introduces BASIC Stamp programming, circuit building, and electronics with fun, multi-sensory experiments. This is the gateway text to Stamps in Class.

Robotics with the Boe-Bot mounts a Board of Education onto a robot chassis. Learn programming as you teach your Boe-Bot to navigate autonomously with multiple sensor circuits.

Elements of Digital Logic will immerse students into a sight, sound, and hands-on logic environment. This curriculum utilizes both hardware and software simultaneously to teach logic.

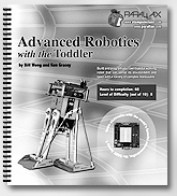
Applied Sensors (formerly *Earth Measurements*) covers program structuring, sensor calibration, EEPROM data logging, conductivity of water, closed-loop feedback, and debugging in an earth science format.



The *Understanding Signals* kit shows students how to view and analyze signals with the OPTAscope. A perfect companion kit, the example circuits are drawn from other Stamps in Class texts.

Process Control introduces aspects of industrial system automation with sensors, mechanical and digital switches, with on-off, differential gap, and PID control methods.

Basic Analog and Digital explores the principles of interfacing analog devices to the digital BASIC Stamp. Measure and control real-world conditions with A-to-D and D-to-A conversions.



Advanced Robotics with the Toddler will teach you how to build and program a high-quality machined two-servo bipedal walking robot controlled by an embedded BASIC Stamp 2.

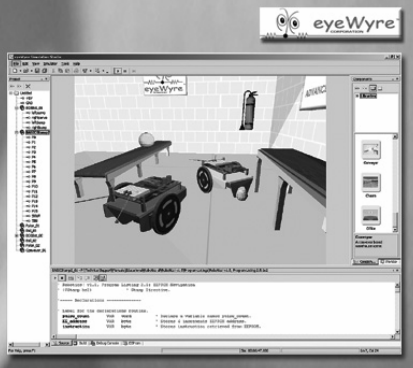
PARALLAX www.stampsinclass.com

eyeWyre Boe-Bot Simulator

Using the eyeWyre Simulation Studio, you can now write your BASIC Stamp code, debug it one step at a time in virtual hardware, download it to your favorite Parallax robot and watch it run! A real-world physical environment provides an assortment of props, sensors and terrain. It's simple. Drag a Boe-Bot into the environment, load it with your code and watch it run! Adding objects such as pylons, balls or changing scenery is simple. Following light, avoiding objects with whiskers or infrared works just like it does in the real world.

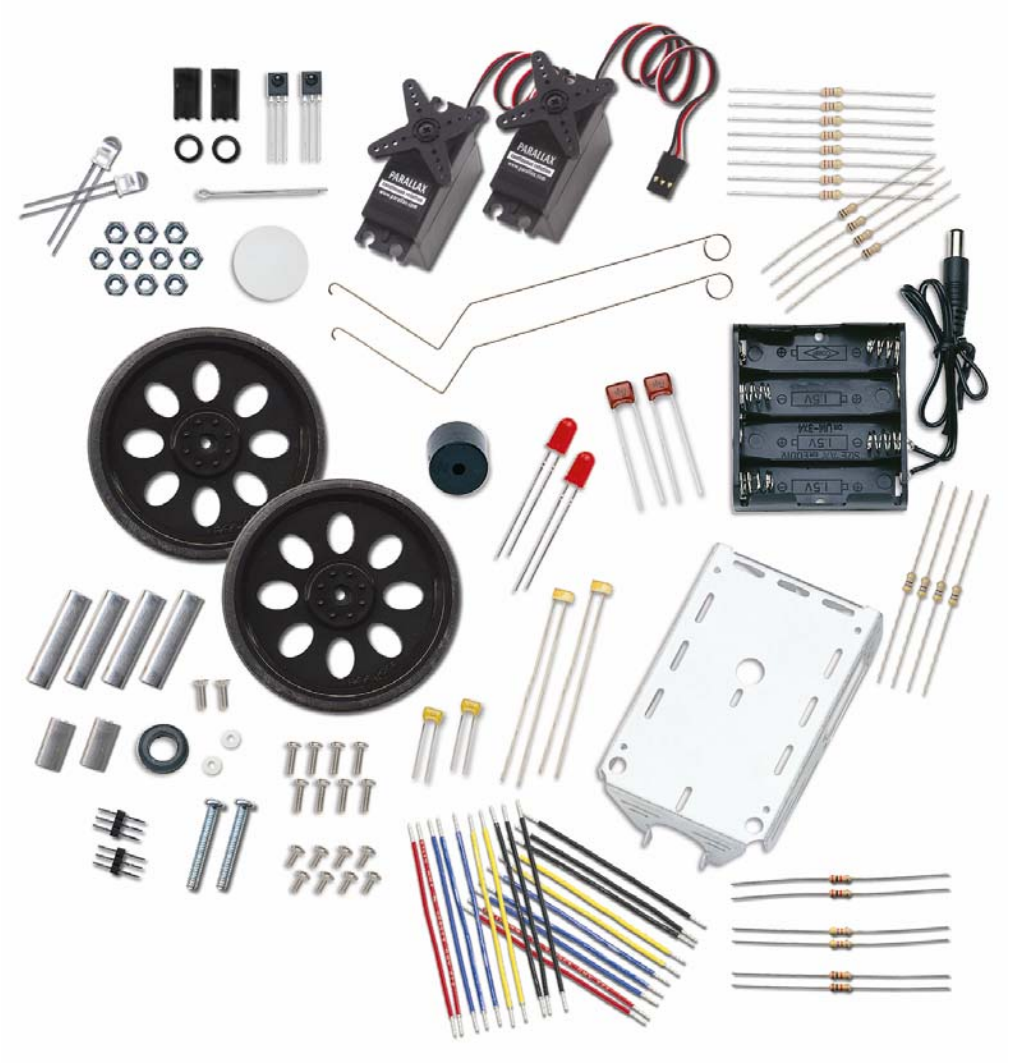
eyeWyre Simulation Studio Advance Features

- Interactive, real-time, 3D simulation
- Complete BASIC Stamp 2 series simulation support
- Fully interactive physics enabled environment
- Expandable component library using the add-on architecture, eyeON™
- Complete built-in script language allows you to customize your simulations
- Simulation of the Parallax Boe-Bot, SumoBot, and the Toddler *(each software package sold separately)*



Visit our website to download eyeWyre Videos and an eyeWyre Boe-Bot ScreenSaver.

PARALLAX
www.parallax.com



Teile und Mengen in den verschiedenen Boe-Bot® Robot Kits können ohne Vorankündigung geändert werden. Die Teile können sich von den im Bild gezeigten unterscheiden. Bitte kontaktieren Sie stampsinclsss@parallax.com wenn Sie Fragen zu Ihrem Kit haben. (englisch)