



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com


Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



Applied Robotics with the SumoBot

Student Guide

VERSION 1.0

PARALLAX 

WARRANTY

Parallax Inc. 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 Inc. 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 Inc. 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

This documentation is Copyright 2005 by Parallax Inc. By downloading or obtaining a printed copy of this documentation or software you agree that it is to be used exclusively with Parallax products. Any other uses are not permitted and may represent a violation of Parallax copyrights, legally punishable according to Federal copyright or intellectual property laws. Any duplication of this documentation for commercial uses is expressly prohibited by Parallax Inc. Duplication for educational use is permitted, subject to the following Conditions of Duplication: Parallax Inc. grants the user a conditional right to download, duplicate, and distribute this text without Parallax's permission. This right is based on the following conditions: the text, or any portion thereof, may not be duplicated for commercial use; it may be duplicated only for educational purposes when used solely in conjunction with Parallax products, and the user may recover from the student only the cost of duplication.

This text is available in printed format from Parallax Inc. Because we print the text in volume, the consumer price is often less than typical retail duplication charges.

BASIC Stamp, Stamps in Class, Board of Education, Boe-Bot SumoBot, SX-Key and Toddler are registered trademarks of Parallax, Inc. If you decide to use registered trademarks of Parallax Inc. on your web page or in printed material, you must state that "(registered trademark) is a registered trademark of Parallax Inc." upon the first appearance of the trademark name in each printed document or web page. HomeWork Board, Parallax, and the Parallax logo are trademarks of Parallax Inc. If you decide to use trademarks of Parallax Inc. on your web page or in printed material, you must state that "(trademark) is a trademark of Parallax Inc.", "upon the first appearance of the trademark name in each printed document or web page. Other brand and product names are trademarks or registered trademarks of their respective holders.

ISBN 1-928982-34-4

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 Inc. 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.

INTERNET DISCUSSION LISTS

We maintain active web-based discussion forums for people interested in Parallax products. These lists are accessible from www.parallax.com via the Support → Discussion Forums menu. These are the forums that we operate from our web site:

- [BASIC Stamps](#) – 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, subscribers 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](#) – Exclusively for 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 and obtain Teacher's Guides.
- [Translators](#) – 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.
- [Robotics](#) – Designed exclusively for Parallax robots, this forum is intended to be an open dialogue for robotics enthusiasts. Topics include assembly, source code, expansion, and manual updates. The Boe-Bot[®], Toddler[®], SumoBot[®], HexCrawler and QuadCrawler robots are discussed here.
- [SX Microcontrollers and SX-Key](#) – Discussion of programming the SX microcontroller with Parallax assembly language SX – Key[®] tools and 3rd party BASIC and C compilers.
- [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.

ERRATA

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.

Table of Contents

Preface	iii
Introduction.....	iii
Educator Resources.....	iv
The Stamps In Class Educational Series.....	iv
Foreign Translations.....	v
Special Contributors.....	vi
Chapter #1: Mechanical Adjustments	7
Before You Get Started.....	7
Small Adjustments can Make a Big Difference	7
ACTIVITY #1: Adjusting the Plow.....	8
ACTIVITY #2: Preventing Servo SlowDown.....	16
ACTIVITY #3: Friction Forces - Your SumoBot's Allies.....	22
Summary.....	38
Chapter #2: EEPROM Tricks and Program Tips	41
EEPROM and Program Management.....	41
ACTIVITY #1: A Closer Look at the EEPROM	42
ACTIVITY #2: Using and Reusing Variables.....	50
ACTIVITY #3: Program On/Off with Reset	55
ACTIVITY #4: Pushbutton, LED, and Speaker.....	58
ACTIVITY #5: Pushbutton Program Mode Selection	65
ACTIVITY #6: Integrating Programs.....	69
Summary.....	75
Chapter #3: Sensor Management	77
Sensors - Testing, Tuning, and Storing the Results.....	77
ACTIVITY #1: Testing and Tuning Infrared Object Detectors	78
ACTIVITY #2: A Closer Look at the QTI Line Sensors.....	95
ACTIVITY #3: Self Calibrating QTI Sensors.....	102
ACTIVITY #4: Reading the QTI Sensors More Quickly.....	107
ACTIVITY #5: Adding and Testing Sensors and Indicators	117
ACTIVITY #6: Testing All Sensors	122
ACTIVITY #7: Organizing Sensors with Flag Bits	129
ACTIVITY #8: Variable Management for Large Programs.....	132
Summary.....	140
Chapter #4: Navigation Tips	143
Sensor Flags and Navigation States.....	143
ACTIVITY #1: Servo Control with Lookup Commands.....	144
ACTIVITY #2: Setting Your Sights on the Opponent.....	155

ACTIVITY #3: Using Peripheral Vision.....	164
ACTIVITY #4: Introduction to State Machines and Diagrams.....	170
ACTIVITY #5: Search Pattern and Tawara Avoidance	176
ACTIVITY #6: Fully Functional Sumo Example Programs.....	186
Summary.....	203
Chapter #5: Debugging and Datalogging	207
Seeing what it Sees and Understanding what it Does	207
ACTIVITY #1: Using the LED to Signal an Event.....	208
ACTIVITY #2: Conditional Compiling.....	212
ACTIVITY #3: Debugging Problem Behaviors	216
ACTIVITY #4: Datalogging a Competition Round.....	233
Summary.....	253
Appendix A: System Requirements and Parts Listing.....	255
Index.....	259

Preface

INTRODUCTION

Robotics is currently enjoying ever increasing popularity with students. Especially when it involves a contest or competition, enthusiasm runs high as participants put everything they've got into their robots in hopes of winning top honors. With this in mind, Parallax developed the SumoBot[®] Robot Competition Kit and SumoBot Competition Ring as an inexpensive way for technology, programming, pre-engineering, and engineering classes to hold their own robotics competitions.

This textbook guides students through a variety of electronics, programming and physics activities as they prepare their SumoBot robots for SumoBot vs. SumoBot competition. Each of the principles presented are of general value to robotics students, applied in such a way as to add something to the SumoBot's competition performance.

Examples from electronics are mostly review from *What's a Microcontroller* and *Robotics with the Boe-Bot*, the entry-level texts to the Stamps in Class series, and include basics such as controlling LED indicators, speakers, and servos. Sensor basics include digital devices like pushbuttons and infrared receivers as well as analog devices like the QTI line sensors, which involve RC-decay measurements. More advanced electronic topics such as frequency response and thresholds for RC-decay measurements are also included.

Physics principles include introductions to time vs. distance at a constant velocity, force, mass, acceleration, coefficients of friction, and free body diagrams. While the physics experiments are optional, they can give students a new view to the direct benefit of experimentation to mechanical designs.

The programming topics in this book include many of the basics, such as looping, conditions, subroutines, saving variable space, using compiler directives, and adhering to coding conventions for the sake of debugging and reusable code. Some unique robotics and embedded systems topics are also included, such as sensor management, state machine design, and datalogging to capture real-time sensor events and navigation states for isolating robotic misbehaviors.

EDUCATOR RESOURCES

While the SumoBot Competition kit is designed for the classroom, it really provides an excellent starting point for the robotics enthusiast who wants to have a first taste of robot sumo wrestling. This book is written for ages 14 and up, and it contains lessons that can be useful additions to a variety of courses, including robotics, physics, technology, and pre-engineering.

Students as well as hobbyists working through this text are encouraged to use the public Stamps in Class forum to collaborate on the questions, exercises and projects at the end of each chapter. You can get there by going to forums.parallax.com, then click the Stamps in Class link.

As of this 1.0 revision, there are no answer keys or teachers guides available. Parallax does, however, have many resources and a support forum specially designed for instructors to use as a collaborative tool. Instructors should contact Parallax Inc. directly for more details.

THE STAMPS IN CLASS EDUCATIONAL SERIES

Applied Robotics with the SumoBot is considered an advanced text in the Stamps in Class educational series, and it is recommended that the student be familiar with the concepts introduced in *Robotics with the Boe-Bot*. All of the books listed are available for free download from www.parallax.com. The versions cited below were current at the time of this printing. Please check our web sites www.parallax.com or www.stampsinclass.com for the latest revisions; we continually strive to improve our educational program.

Stamps in Class Student Guides:

There are two entry-level text to choose from; either one is an appropriate gateway to the rest of the series.

“What’s a Microcontroller?”, Student Guide, Version 2.2, Parallax Inc., 2004
“Robotics with the Boe-Bot”, Student Guide, Version 2.2, Parallax Inc., 2004

For a well-rounded introduction to the design practices that go into modern devices and machinery, continue on with the following titles:

“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

Educational Project Kits:

Elements of Digital Logic, *Understanding Signals* and *Experiments with Renewable Energy* focus more closely on topics in electronics, while *StampWorks* provides a variety of projects that are useful to hobbyists, inventors and product designers interested in trying a variety of projects. *Advanced Robotics with the Toddler* further develops robotics skills with a bipedal walking robot.

“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., 2001
“Understanding Signals”, Student Guide, Version 1.0, Parallax Inc., 2003
“Advanced Robotics: with the Toddler”, Student Guide, Version 1.2, Parallax Inc., 2003

Reference

This book is an essential reference for all Stamps in Class Student Guides. It is packed with information on the BASIC Stamp series of microcontroller modules, our BASIC Stamp Editor, and our PBASIC programming languages.

“BASIC Stamp Manual”, Version 2.2, Parallax Inc., 2005

FOREIGN TRANSLATIONS

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 which you may join. Go to www.yahogroups.com and search for “Parallax Translators.” This will ensure that you are kept current on our frequent text revisions.

SPECIAL CONTRIBUTORS

Parallax Inc. would like to recognize the Education Team members who made this book possible: Education and Project Manager Aristides Alvarez, Author and Engineer Andy Lindsay, Technical Illustrator Rich Allred, Graphic Designer Jen Jacobs, and Technical Editor Stephanie Lindsay. Special thanks also go to Ryan Clarke in Tech Support and Kris Magri in Education for their insightful and speedy review, and, as always, to Ken Gracey, the founder of Parallax Inc.'s Stamps in Class educational program.

Chapter #1: Mechanical Adjustments

BEFORE YOU GET STARTED

To complete the activities in this book, you will need to build, program and test two complete SumoBot robots by following the activities in the *SumoBot Manual*. Also, since *Applied Robotics with the SumoBot* is an advanced robotics text that builds upon the concepts introduced in *Robotics with the Boe-Bot*, familiarity with that material is recommended. *Robotics with the Boe-Bot* is available for download from www.parallax.com.

You will also need additional electronic components and a SumoBot Robot Competition Ring poster. The complete robot kits and these other items are all included in the SumoBot Robot Competition Kit. If you already own two SumoBot robots, the components and poster can also be purchased separately from www.parallax.com. A few common household items are also needed for some activities. Please see Appendix A for the full parts listings.

As you go through the activities in this text, you can type all of the program code directly into the BASIC Stamp Editor from the listings in this book, or you can download the listed programs from the SumoBot Robot Competition Kit product page at www.parallax.com.

SMALL ADJUSTMENTS CAN MAKE A BIG DIFFERENCE

This chapter introduces some of the mechanical adjustments you can make to your SumoBot robot to improve its performance against other SumoBots. They include plow adjustments, making sure your servos are running at full speed, and modifications you can make to improve your SumoBot's grip on the ring.

When it's SumoBot vs. SumoBot, something as simple as a small adjustment to the plow can make a big difference, as you will see in Activity #1. While motor speed doesn't make as much of a difference, it is another factor that can impact a SumoBot's likelihood of winning each round. Activity #2 will demonstrate how taking too much time to read sensors between delivering control pulses to the servos can slow your SumoBot down.

Friction is that force which prevents your SumoBot from sliding. More friction between the SumoBot's tire tread and the sumo ring means the SumoBot can push harder against

its opponent. The less friction, the more easily the SumoBot slips, which means it can no longer push as hard.

There are two ways to increase the friction between the tire tread and sumo ring. First, increase the SumoBot's weight, and second, find the best possible tread material. The interesting thing about tire tread materials is that they have to be paired with the material the sumo ring is made out of. While one material might work best in the SumoBot Competition Ring poster, a different material might work better on a painted wood surface. Activity #3 introduces experiments you can perform to quantify the increases in friction from both increasing the SumoBot's weight and changing the tread materials.

ACTIVITY #1: ADJUSTING THE PLOW

One of the keys to increasing your SumoBot's chances of winning a match against another Parallax SumoBot is adjusting the plow so that it's more likely to pass under the opponent's plow. For example, the "winning" SumoBot in Figure 1-1 has the mechanical advantage, with its opponent off balance. In this activity, you will repeatedly test and adjust your SumoBots' plows while looking for the setting that will give one of your SumoBots the best chances in the sumo ring.

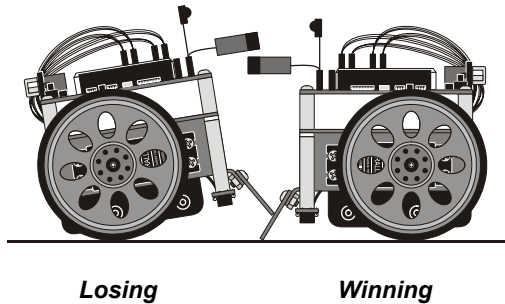


Figure 1-1
SumoBots Wrestling
*The one on the right
has the advantage*

Parts Required

- (2) Fully assembled and tested Parallax SumoBot robots
- (1) SumoBot Competition Ring poster, or other sumo ring
- Clear tape (not included)
- Black felt-tip marker (not included)

Setting up the SumoBot Competition Ring Poster

The SumoBot robot and SumoBot Robot Competition Ring poster are for indoor use only. For best results, follow these setup instructions:

- √ Unfold the SumoBot Competition Ring poster, and re-fold it the opposite way so the creases will lie flat, then unfold it again.
- √ Find a location with these characteristics:
 - Indoors, and well away from direct or indirect sunlight
 - Fluorescent or indirect incandescent lighting
 - Hard, flat, smooth surface such as a large table or floor; not on carpet
 - Surface any color other than white, and preferably not super-shiny, or the infrared detectors may see it
 - No walls or other objects within 1 meter of the outside of the ring
- √ Place the ring on this surface, and secure the corners and edges with clear tape to make it lie flat.
- √ If, from frequent folding and unfolding, the creases start to appear white, touch them up with a black felt-tip marker.
- √ If possible, place some heavy books on top of the creases for a couple of days to flatten them out.

Initial Adjustments

The plow is held to the chassis by two screws shown in Figure 1-2. Each screw passes through a slot in the plow. Adjusting the plow is a simple matter of loosening the two screws, changing the plow's position, then retightening the screws. Each slot only has a few millimeters of wiggle room. Even so, the slight adjustments you can make to the plow's height and tilt can make a big difference in performance.

- √ Start by adjusting each plow so that it is flush to the sumo ring's surface along its whole length, as shown in Figure 1-2.

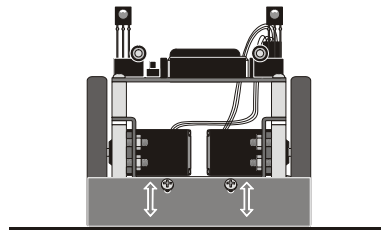


Figure 1-2
Adjusting the Plow

Testing Plow Adjustments

A simple program to make each SumoBot go forward should be used to test the plows. This eliminates the possibility of IR object detectors interfering with the SumoBot's forward motion. For example, if one of the SumoBot's IR detectors briefly misses its opponent, the SumoBot will hesitate and might not be going full speed when the two SumoBots collide. It's true that this will also happen during a match, but during practice it's best to test only one variable at a time, in this case, the plow adjustment.



Example Source Code Available

Remember, you can save yourself some typing and debugging! The example BASIC Stamp programs printed in this text are available for free download as .bs2 source code from the *Applied Robotics with the SumoBot* product page at www.parallax.com. The various modifications and "Your Turn" programs are not provided.

- √ Label both your SumoBots so that you can distinguish them. If A and B aren't interesting enough labels, some searching on the Internet will yield names of sumo legends as well as present stars.
- √ Enter Forward100Pulses.bs2 into the BASIC Stamp Editor (listed on the next page).
- √ Load the program into both SumoBots.
- √ Place both SumoBots facing each other as shown in Figure 1-3.

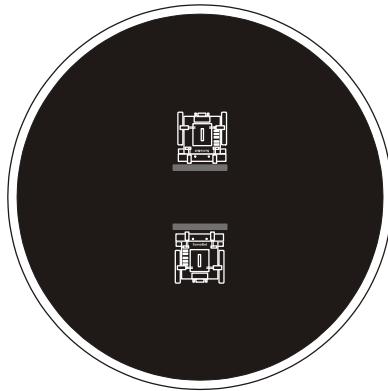


Figure 1-3
Facing Off on the
Shikiri Lines

*Place the SumoBots
so that they are
directly facing each
other before pressing
and releasing Reset.*

- √ Press and release both SumoBot Reset buttons at the same time.
- √ Make notes on which SumoBot appeared to be winning the match and why.

- √ Repeat five to ten times to be sure which SumoBot's plow adjustment has the advantage.
- √ Adjust the plow of the SumoBot that appeared to lose more often, and repeat the test.
- √ When you are confident that one of your SumoBots has a winning plow adjustment, try lots of different adjustments on the other SumoBot to find out if there is any better adjustment that can make it the winner.
- √ When you are satisfied with your winning SumoBot's plow, leave its adjustment as-is, and tune the other SumoBot's plow until its chances of winning/losing are close to even.

Example Program: Forward100Pulses.bs2

```
' -----[ Program Description ]-----
' Applied Robotics with the SumoBot - Forward100Pulses.bs2
' SumoBot goes forward 100 pulses after Reset button is pressed and released.
' To repeat the forward motion, press/release the Reset button twice.
'
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]-----
ServoLeft      PIN    13          ' Left servo connected to P13
ServoRight     PIN    12          ' Right servo connected to P12

' -----[ EEPROM Data ]-----
RunStatus      DATA    0          ' Start with program not running

' -----[ Variables ]-----
temp           VAR      Byte       ' Temporary variable
counter        VAR      Byte       ' FOR...NEXT loop counter

' -----[ Initialization ]-----
DEBUG CLS          ' Clear the Debug Terminal
Reset_Button:

  READ RunStatus, temp          ' read current status
  temp = ~temp                 ' invert status
  WRITE RunStatus, temp        ' save for next reset
  IF (temp > 0) THEN           ' 0 -> End, 1 -> Main routine
    DEBUG "Press/release Reset..."
  END
ENDIF
```



```

DEBUG "Main program running...", CR           ' Display program status
' -----[ Main Routine ]-----
FOR counter = 1 TO 100                         ' Deliver 100 forward pulses
  PULSOUT ServoLeft, 850
  PULSOUT ServoRight, 650
  PAUSE 20
NEXT
DEBUG "Done!", CR,                             ' User instructions
  "Press/release reset", CR,
  "twice to restart...", CR
END

```

Understanding Forward100Pulses.bs2

When you click the BASIC Stamp Editor's Run button, the program downloads to the SumoBot's BASIC Stamp. The `Reset_Button` routine in the Initialization displays the message "Press/release Reset...", then it ends the program. When you press and release the Reset button on the SumoBot board, the same `Reset_Button` routine displays the message "Main program running..." and moves on to the Main Routine and the servos start turning. If you leave the SumoBot connected to its serial cable and press/release the Reset button a third time, you will again see the "Press/release Reset button" prompt. Repeat a fourth time, and the servos will run for a couple seconds.

The reason the `Reset_Button` routine is able to perform this function is because it manipulates values stored in the SumoBot's EEPROM program memory. The portion of this memory that is not used to store the program can be used to store values. While the BASIC Stamp's RAM memory is erased whenever the power is turned off or the Reset button is pressed and released, the EEPROM memory retains the values stored in it. That's why the same program runs after your turn the SumoBot's power off, then back on.

The ability to retrieve values stored in EEPROM, change them, and store them back into EEPROM is what allows the `Reset_Button` routine to track whether you've pressed and released the Reset button an odd or even number of times. The mechanics of exactly how the `Reset_Button` routine does this is covered in more detail in Chapter 2, Activity #2. For now, just keep in mind that the `Reset_Button` routine allows the program to continue to the Main Routine when you have pressed and released the SumoBot's Reset button an odd number of times. That means, the first, third, fifth,... time you press and release the Reset button, the program will continue to the Main Routine, and the servos will turn for about 2 1/2 seconds. Whenever you have pressed/released the Reset button

an even number of times, including zero, the `Reset_Button` routine just displays the message prompting you to press the Reset button, then it ends the program ends.

If you have already completed *Robotics with the Boe-Bot*, the forward motion code in `Forward100Pulses.bs2`'s Main Routine should be very familiar. Although Chapter 4, Activity #1 features a quick review of the servo control principles that were introduced in *Robotics with the Boe-Bot*, the information box below lists a few activities you can try to get up to speed.

Understanding How Pulses Control Servos

If you have not already worked through *Robotics with the Boe-Bot v2.2*, download it from www.parallax.com, and try the following chapters and activities:

Chapter 2, Activity #6

Chapter 3, Activity #4

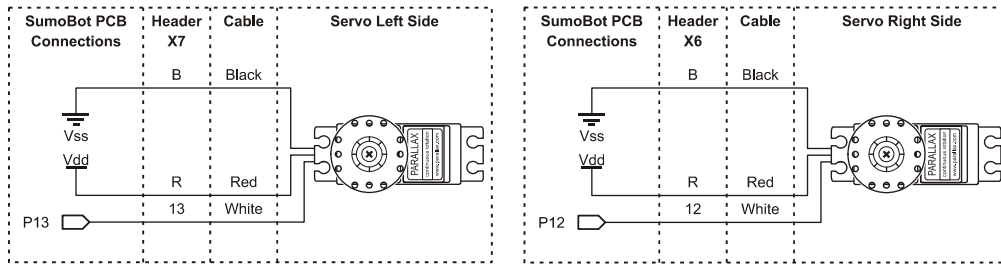
Chapter 4, Activity #1 to Activity #6

The servos are connected to the same I/O pins, so the example programs will run correctly in your SumoBot. The only part that will not work is the piezospeaker, which is connected to P4 in *Robotics with the Boe-Bot*. If your piezospeaker circuit is connected to P1, simply update every instance of `FREQOUT 4, 2000, 3000` to `FREQOUT 1, 2000, 3000`.



Figure 1-4 shows the servo connections you made in the *SumoBot* text. The left servo connects to header X7 on the SumoBot board. The plug at the end of the servo's cable plugs into X7 so that the black wire connects to the B pin, the red wire connects to the R pin, and the white signal line connects to the pin labeled 13. Traces on the SumoBot printed circuit board in turn connect the header pin labeled 13 to BASIC Stamp I/O pin P13. The also connect the pin labeled R to Vdd, which is the board's regulated 5 V power supply, and the pin labeled B to Vss, which is the board's ground or 0 V connection. The right servo connects to header X6. The difference with X6 is that it connects that servo's white signal line to BASIC Stamp I/O pin P12 instead of P13.

Figure 1-4 SumoBot Servo Connections



The instructions that make the SumoBot move forward starts with these `PIN` declarations:

```
ServoLeft    PIN    13
ServoRight   PIN    12
```

The SumoBot's left servo is connected to P13, so I/O pin P13 is given the name `ServoLeft`. Likewise, P12 is connected to the right servo, so it's named `ServoRight`.

In order for the program to apply 100 pulses, a `counter` variable is declared:

```
counter      VAR    Byte
```

This `FOR...NEXT` loop delivers 100 forward pulses to the SumoBot's servos. According to *Robotics with the Boe-Bot*, this loop delivers 40.65 pulses per second, so the SumoBot will roll forward for $100 \div 40.65 = 2.46$ seconds.

```
FOR counter = 1 TO 100
  PULSOUT ServoLeft, 850
  PULSOUT ServoRight, 650
  PAUSE 20
NEXT
```

Your Turn - Does Angle of Approach Matter?

By experimenting with different angles of approach, you might (or might not) find an even more "winning combination". Figure 1-5 shows examples of two different approaches. The edge of a SumoBot's plow collides with the flat of the other's (left). The SumoBot is using a curved approach (right). They aren't necessarily better approaches, but they are worth investigating for the sake of better understanding the relative merits and drawbacks of each.

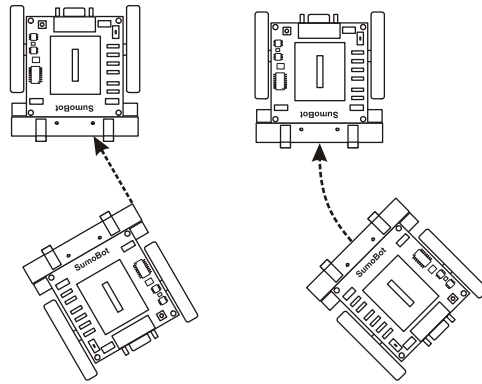


Figure 1-5
Other Collision Paths

One SumoBot can approach at an angle, or even with a curved path.



Do not adjust your plows.

The strategies presented in this book will use the head-on approach. You can modify your programs (and the plows) to optimize for different approach angles, but wait until after Chapter 5.

- ✓ Test a variety of approach angles and plow intersection points with the same full-speed-forward settings.

To test curved approaches, you will need make one of the servos turn slower than the other. You can do this by modifying the code in the Main Routine. Simply reduce one of the `PULSOUT` command's *Duration* arguments closer to 750. If you want it to curve right, change `PULSOUT ServoRight, 650` to `PULSOUT ServoRight, 720`. For a tighter turn, try `PULSOUT ServoRight, 730`. `PULSOUT ServoRight, 735` will make the turn tighter still. For a wider turn, try `PULSOUT ServoRight, 715`, or even `PULSOUT ServoRight, 710`.

You can repeat this for left turns. First, restore the right servo to `PULSOUT ServoRight, 650`. Then, change the left servo's control signal to `PULSOUT ServoLeft, 780`. The same adjustment pattern applies for the left servo. For tighter turns, adjust the `PULSOUT` commands *Duration* argument closer to 750, and for wider turns adjust it closer to 850.

- ✓ Experiment with a curved approach with one SumoBot and a straight approach with the other.
- ✓ Also experiment with curved vs. curved.



A notebook for your observations - keep notes on the various results you observe for developing wrestling strategies.

ACTIVITY #2: PREVENTING SERVO SLOWDOWN

A SumoBot that executes certain maneuvers more quickly will have an edge over a slower opponent. One of the things that can slow your SumoBot down is trying to read too many sensors between servo pulses. This activity examines how much time your SumoBot can actually take between servo pulses before it starts to slow down. Later activities will introduce ways to reduce the time it takes to read certain sensors.



Maximizing speed and strength

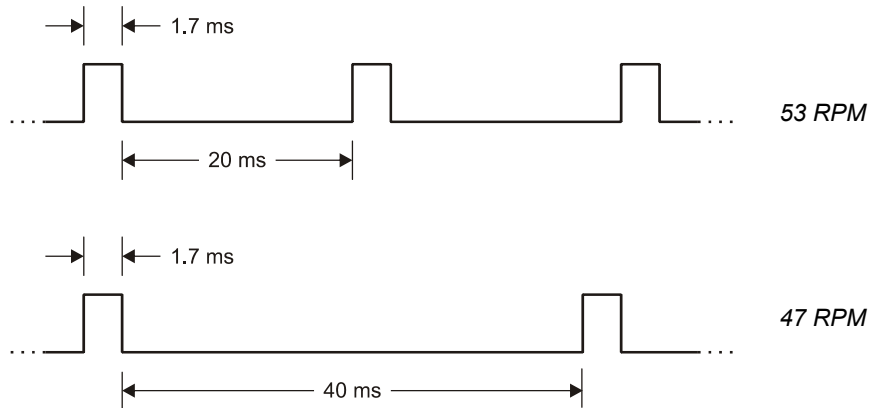
In robotics clubs, competitors often rely on new servo gear sets or DC motors along with hobby RC batteries to optimize their competition robot for speed and brute force.

Parts Required

- (2) Parallax SumoBot robots
- (8) New 1.5 volt AA batteries - same brand and type

Top Speed vs. Low Time

The high time of both pulse trains in Figure 1-6 control servo speed and direction. Because both pulse trains have high times of 1.7 ms, either pulse train will make a servo turn full speed counterclockwise. The problem is that full speed for a servo with 40 ms between pulses isn't quite as fast as the full speed for a servo with 20 ms between pulses. For example, the servo might turn 53 RPM with 20 ms pauses between pulses, but only 47 RPM with 40 ms.

Figure 1-6 Top Speed vs. Low Time

As mentioned earlier, taking too much time between pulses to check sensors can cause the servos to slow down. IR object detectors don't take a very big bite out of the low time between servo pulses. Each one only takes a couple milliseconds to read. QTI line detectors, on the other hand, can take up to 20 milliseconds each. The time it takes a QTI to complete its measurement depends on ambient light and how reflective the surface is. The problem is, if both QTIs take 20 ms to read, that pushes the low time into the 40 ms range, which means the servos will slow down, which may put your SumoBot at a disadvantage.

In this activity, you will determine just how much time you can take between servo pulses before the servos start to slow down. This will be an important consideration in the sensor management chapter. One of the goals of sensor management will be to figure out how to read as many sensors as possible between each servo pulse without exceeding the time limit. By making a note of the maximum low time before servo slowdown in this activity, you will have a key piece of information for the sensor management chapter.

Testing Speed vs. Low Time - How Much Does it Matter?

A SumoBot race is a good way to examine the speed difference a longer low time can make. Simply program both SumoBots to travel forward at full speed, with different low times. Both programs should deliver pulses in an infinite loop. Each program should also make use of the same initialization routine from Activity #1 so that you can use the Reset button to start and stop the race.

With a couple modifications to Forward100Pulses.bs2 from Activity #1, you'll be ready to go.

- √ Save Forward100Pulses.bs2 as ForwardLowTimeTest.bs2.
- √ Add a **LowTime** constant declaration:

```
LowTime      CON      20
```

- √ Change the **FOR...NEXT** loop in the Main Routine to a **DO...LOOP**, and substitute the **LowTime** constant for the 20 in the **PAUSE** command's *Duration* argument:

```
DO
  PULSOUT ServoLeft, 850
  PULSOUT ServoRight, 650
  PAUSE LowTime
LOOP
```

It's important to use new batteries in both SumoBots. It's also best to swap programs and re-test to make sure that one SumoBot doesn't happen to be slower than the other. This can be especially common in the classroom, where servos may have been subject to differing levels of wear and tear over time.

Example Program: ForwardLowTimeTest.bs2

- √ Load fresh alkaline batteries into both SumoBots.
- √ Enter and download ForwardLowTimeTest.bs2 to SumoBot A.
- √ Change the **LowTime CON** directive from 20 to 40.
- √ Download the modified program into SumoBot B.
- √ Set them on a flat surface for the race.
- √ Press/release both Reset buttons at the same time to start the race.
- √ Follow the SumoBots for 3 seconds, then press/release the Reset buttons again to end the race.
- √ Measure the distance each SumoBot traveled, and make a note of it.
- √ Divide the distance by 3 to calculate each SumoBot's speed in distance per second.
- √ Swap the programs so that SumoBot B now has the program with 20 ms pauses and SumoBot A has the program with 40 ms pauses.
- √ Repeat the race and measurement.

- √ Compare the results of the two trials and determine which program gives the SumoBot better performance.

```
' -----[ Program Description ]-----
' Applied Robotics with the SumoBot - ForwardLowTimeTest.bs2
' SumoBot goes forward indefinitely. Use the Reset button to start and stop
' the forward motion.
'
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----[ I/O Definitions ]-----
ServoLeft      PIN    13          ' Left servo connected to P13
ServoRight     PIN    12          ' Right servo connected to P12

' -----[ EEPROM Data ]-----
RunStatus      DATA   0          ' Start with program not running

' -----[ Constants ]-----
LowTime        CON    20          ' Try 1 SumoBot with 20 ms pulses
' Try the other with 40 ms pulses

' -----[ Variables ]-----
temp           VAR     Byte       ' Temporary variable
counter        VAR     Byte       ' FOR...NEXT loop counter

' -----[ Initialization ]-----
DEBUG CLS          ' Clear the Debug Terminal

READ RunStatus, temp ' read current status
temp = ~temp       ' invert status
WRITE RunStatus, temp ' save for next reset
IF (temp > 0) THEN ' 0 -> End, 1 -> Main routine
  DEBUG "Press/release Reset..."
  END
ENDIF

DEBUG "Main program running...", CR ' Display program status

' -----[ Main Routine ]-----
DO ' Forward pulses indefinitely
  PULSOUT ServoLeft, 850
  PULSOUT ServoRight, 650
  PAUSE LowTime
```



```

LOOP
DEBUG "Press/release reset", CR,           ' User instructions
      "twice to restart...", CR
END

```

Your Turn - More Pulses, Less Distance

Chapter 4, Activity #3 in *Robotics with the Boe-Bot* demonstrates how the amount of time a servo turns translates to distance traveled. When there's less time between each pulse, the program will have to send the servos more pulses to make them turn for the same amount of time. This can make a huge difference in certain maneuvers, especially distance and turns. Let's take a closer look at turns. If the low time between pulses is cut in half, it means you have to deliver around twice as many pulses to execute the same maneuver.

- ✓ Save ForwardLowTimeTest.bs2 as ForwardLowTimeTestYourTurn.bs2
- ✓ Set the **LowTime CON** directive to 40.
- ✓ Replace the **DO...LOOP** in the Main Routine with this:

```

FOR counter = 1 to 15           ' Deliver 15 left turn pulses
  PULSOUT ServoLeft, 850
  PULSOUT ServoRight, 850
  PAUSE LowTime
NEXT

```

- ✓ Download the modified program to SumoBot A.
- ✓ Set the **LowTime CON** directive to 20.
- ✓ Modify the Main Routine again, this time doubling the number in the **FOR...NEXT** loop's **EndValue**:

```

FOR counter = 1 to 30         ' Deliver 30 left turn pulses
  PULSOUT ServoLeft, 850
  PULSOUT ServoRight, 850
  PAUSE LowTime
NEXT

```

- ✓ Download the modified program to SumoBot B.

The distance traveled will not be exactly the same because the servos don't turn as fast with 40 ms pauses. This means that you'll probably have to use a little less than twice as many pulses with 20 ms pauses to match the distance of the program with 40 ms pulses.

- √ Tune the `EndValue` in the `FOR...NEXT` loop with the 20 ms pauses to get as close as possible to the distance traveled with 40 ms pauses.

How Much Time Can the SumoBot Take between Pulses?

`ForwardLowTimeTest.bs2` can also be used to determine the maximum *PAUSE Duration* between pulses before servo slowdown sets in. You can do this by measuring a servo's speed with a 20 ms low time, then again with a 21 ms low time, then 22 ms, and so on.

- √ Slip a jumper wire under one of the rubber band tires as shown in Figure 1-7. This will serve as a marker for counting wheel revolutions.

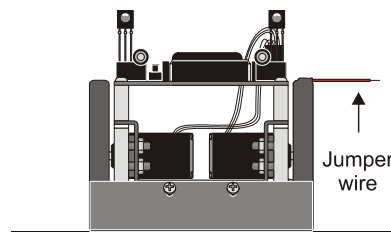


Figure 1-7
Jumper Wire Marker
under Tire Tread

- √ Start over with an unmodified version of `ForwardLowTimeTest.bs2`.
- √ Begin with the `LowTime` constant set to 20, and run the program.
- √ Press/release the Reset button to start the servos, and start counting revolutions.
- √ Press/release the Reset button 10 seconds later, and make a note of how many revolutions the wheel turned.
- √ Multiply this value by 6 to calculate the servo's rotational speed in RPM. For example, if the servo turned 8.75 revolutions in 10 seconds, the servo is turning at 52.5 RPM:

$$\frac{8.75 \text{ revolutions}}{10 \text{ seconds}} \times \frac{60 \text{ seconds}}{\text{minute}} = 52.5 \frac{\text{revolutions}}{\text{minute}} = 52.5 \text{ RPM}$$

- √ Change the `LowTime` constant to 21.