Boe-Bot Gripper

Assembly Documentation and Applications

Version 2.0

PARALLAX INC

Warranty

Parallax, Inc. warrants its products against defects in materials and workmanship for a period of 90 days. If you discover a defect, Parallax will, at its option, repair, replace, or refund the purchase price. Simply call for a Return Merchandise Authorization (RMA) number, write the number on the outside of the box and send it back to Parallax. Please include your name, telephone number, shipping address, and a description of the problem. We 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 does not apply if the product has been altered or damaged.

Copyrights and Trademarks

ISBN 1-928982-18-2

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, and any costs or 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.

Internet Access

We maintain Internet systems for your use. These may be used to obtain software, communicate with members of Parallax, and communicate with other customers. Access information is shown below:

E-mail: stampsinclass@parallax.com
Web: http://www.parallax.com and http://www.stampsinclass.com

Internet BASIC Stamp Discussion List

The Parallax web site (www.parallax.com) has many application 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 students to ask educators questions, too.
- Parallax Educators – This focus group of 100 members consists exclusively of educators and those who contribute to the development of Stamps in Class curriculum. Parallax created this group to obtain feedback on our curriculum development 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, compilers (BASIC and C). Approximately 600 members.

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

Contents

Preface

In robotics, interacting with the environment is an important objective. Using the Gripper, the Boe-Bot can interact with and alter the world around it, gripping, moving, and repositioning objects. A Gripper falls into the category of robotics devices known as end effectors, devices at the end of a robot arm that are used to grasp or engage objects. Roboticists of all ages can enter the challenging realm of end effectors by building and programming the Boe-Bot Gripper.

Features

Aside from being a very high-quality product, the Gripper design was maximized to best utilize its BASIC Stamp control. There are several key design aspects:

- Single-servo control The design allows a single servo motor to open, close, raise and lower the Gripper. In most designs, the gripping motion requires one motor and raising requires another. With the Boe-Bot Gripper, a single servo motor does it all. This makes the Gripper easier to program and reduces the need for a second servo.

- Grab and hold without servo pulsing The Boe-Bot Gripper will hold onto objects without continuously pulsing the servo motor. It must only be pulsed to open or close the Gripper, after which cleverly designed springs keep the Gripper in place. This feature frees up processing time, allowing the Boe-Bot to concentrate on tasks other than gripping. It also means less power drain and longer battery life. Typically, servo-based Grippers require current consumption monitoring to know if the servo has closed on the object.

- CNC design with .001" tolerances Gripper designers specified a .001" tolerance for the aluminum parts. This means the Gripper should fit together really well and operate smoothly. Parts shouldn't bind on each other.

Application Projects

The Gripper can be used for autonomous and human-controlled projects. In the autonomous example, the Gripper should be able to roam and detect an object prior to pickup. In remote control mode under RF or using a Sony-programmable TV remote control the Gripper can move about and retrieve objects with a human user interface.

What's New in the Boe-Bot Gripper Documentation Version 2.0

Parallax is continually striving to improve its products. In the Summer of 2003, Parallax upgraded several components in the Boe-Bot kits. Since the Gripper is mounted on the Boe-Bot, these changes affected the Gripper Boe-Bot's functioning. Also, some Gripper components were upgraded, and finally, the PBASIC language itself was improved.

Changes which affect the Gripper's Functioning

- New Parallax Continuous Rotation wheel servos in Boe-Bot kit
- New standard servo for the Gripper servo in Gripper kit
- New version of the PBASIC language, PBASIC 2.5

Changes which do not affect the Gripper's Functioning

- Boe-Bot: New wheels, IR emitters, and whiskers
- General: New Board of Education (BOE) Rev C

Photographs

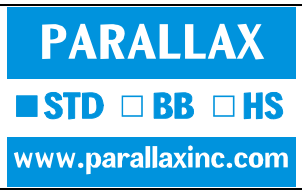
All photographs in this text are of the original hardware: wheels, servos, BOE, etc. No photographs were updated in this version of the *Boe-Bot Gripper Assembly Documentation and Applications.*

Program Listings

All program listings have been updated in this version of the *Boe-Bot Gripper Assembly Documentation and Applications*. **The new programs work with both the newer and older types of servos, in any combination.**

Servo Identification

Check the labeling on your servos. Use the table below to identify your Boe-Bot Gripper's servos.

Servo Type	Servo Label
Old Wheel Servo	
New Wheel Servo	
Old Gripper Servo	
New Gripper Servo	

Servo Configurations

Depending on when you obtained your Boe-Bot and your Gripper kit, you may have one of the following configurations:

- New Wheel Servos and New Gripper Servo
  If you bought your Boe-Bot and Gripper from Parallax recently (Summer 2003 or later)

- Old Wheel Servos and New Gripper Servo
  If putting a new Gripper on an older Bot-Bot

- Old Wheel Servos and Old Gripper Servo
  The original hardware configuration, matching the original Gripper Manual Version 1.0.

- New Wheel Servos and Old Gripper Servo
  This configuration is unlikely since the wheel servos and the Gripper servos were updated at the same time; however, the programs in this version of the manual will support this arrangement.

Effect of New Servos

When turning at full speed, the new Parallax Continuous Rotation servos are roughly 1.75 to 1.8 times as fast as the older wheel servos. This can lead to some pretty crazy Boe-Bot Gripper behavior, because programs that made the older servos travel a certain distance will cause the newer servos to travel almost twice as far. Fortunately, PBASIC 2.5 provides a mechanism which allows a single program to work for many hardware configurations.

Conditional Compilation

One of the many improvements in PBASIC 2.5 is the addition of conditional compilation directives. All the Boe-Bot Gripper programs use conditional compilation directives to specify which servo configuration is in use. The conditional compilation directives are shown below:

```
' Choose your type of wheel servo.  1 for newer, 0 for older.
#DEFINE NewWheelServo = 1                    ' Newer continuous
rotation servo
'#DEFINE NewWheelServo = 0                   ' Older pre-modified
servo

' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                     ' Newer standard servo
'#DEFINE NewGripServo = 0                    ' Older STD servo
```

The **NewWheelServo** and **NewGripServo** values represent which type of servos are in use. The value **1** represents the newer servos, and **0** the older type servos. In the code snippet above, **NewWheelServo** and **NewGripServo** are set to **1**. To change **NewWheelServo** to 0, simply comment out the first line, and uncomment the second line. (Any line starting with an apostrophe is "commented out", meaning it is not evaluated or executed.)

Once the **NewWheelServo** and **NewGripServo** values are set to match the hardware configuration, the programs use **#IF...#THEN...#ELSE** blocks to execute different lines of code depending on the values set

by the **#DEFINE** statements. For example, in the code segment below, **StepSize** will be set to 20 if using the new servos, or to 10 to if using the older servos. Thus, the same program will work for either type of servo.

```
#IF (NewGripServo = 1) #THEN                ' For newer Standard
gripper servo
  StepSize     CON     20                   ' Step size for gripper
servo
#ELSE                                       ' For older STD gripper
servo
  StepSize     CON     10                   ' Step size for gripper
servo
#ENDIF
```

Therefore, for all Gripper program listings, you must edit the conditional compilation section to correctly identify your servo configuration.

Experiment #1: Assembling the Gripper

Assembling the Gripper is a project in itself. It requires patience, attention to detail and some minor skills using hand tools. Because each step is reversible if you encounter problems, it should be easy to fix any mistakes along the way.

Parts Listing

Review the back of this text for a color page of all Gripper parts.

Stock Code	Description	Qty
Machined Parts		
720-28201	Main Frame	1
720-28202	Main Gripper Crank	2
720-28203	Gripper Support Crank	2
720-28204	Crank Axle (Brass Rod)	2
720-28205	Lever Arm	2
720-28206	Crank Bracket L	1
720-28207	Crank Bracket R	1
720-28208	Gripper Spoke Guide Rail (Delrin)	1
720-28209	Gripper Spoke (Square Bar)	1
720-28210	Gripper Support Gantry	1
720-28213	Servo Bracket	2
720-28214	Gripper Plate	2
720-28215	Metal 90 degree bracket for SSIR	2
720-28216	Spoke Axle	1
R/C Hardware		
725-00009	4.5" Control Rod	2
725-00010	2-56 Swivel Ball Links	2
725-00011	2-56 Metal Clevis	2
900-00005	Parallax Servo	1
805-00001	Servo Extension Cable 10"	3
Hardware		
720-28211	Axle pin #1	4
720-28212	Axle pin #2	2
713-00002	Stand-off 1.25"	4
713-00003	Stand-off 1.25" Hollow	2
713-00004	Stand-off 1/4" Male to Female	2

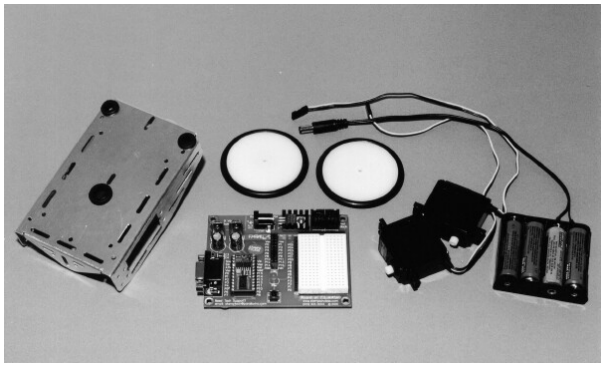
713-00005	Spacer .25" Nylon	3
710-00001	2-56 Philips Flat Head Screw 1/2"	1
711-00001	2-56 nut	1
710-00003	2-56 X 1/4" Philips Flat Head Screw	4
714-00001	4-40 X 1/8" Set Screw	7
710-00004	4-40 X 1 3/4" Screw	2
700-00028	4-40 X 1/4" Screw	12
700-00002	4-40 X 3/8" Screw	5
700-00016	4-40 X 3/8" Philips Flat Head Screw	1
700-00003	4-40 Nut	15
712-00002	4-40 Metal Washer	1
700-00015	Nylon Washer (Screw Size #4)	2
725-00012	Spring	2
700-00059	#4 Lock Washer	1
710-00006	4-40 x 1 / 2 Screw	4
Electronic Comp.		
550-28200	Gripper IR Sensor	1
550-10001	SSIR Sensor	2
451-00302	3-Pin Right Angle Header	3
800-00016	Jumper Wires	2
Miscellaneous		
122-28200	Gripper Manual	1
900-00012	Vinyl Foam .25" x 22" x .125" thick	1
700-00063	Allen Wrench .050	1
700-00064	Parallax Screwdriver	1

Mechanical Assembly Instructions

These instructions refer to different pieces of hardware from the Parts Listing. If you have trouble identifying the type of part referred to in these instructions, see the color back cover of this text which shows each part with a colored picture and Parallax stock code.

Begin by laying out all parts in the kit. Group identical parts together as shown on the back cover. This will make is easier to select the correct parts as you build the Gripper.

Step #1: Remove parts from existing Boe-Bot



Mounting the Gripper requires disassembly of the existing Boe-Bot. Carefully remove the following parts and set all mounting hardware (screws, nuts, etc) aside. You will need them when re-assembling the Boe-Bot.

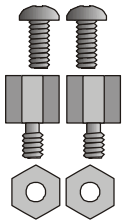
Remove:

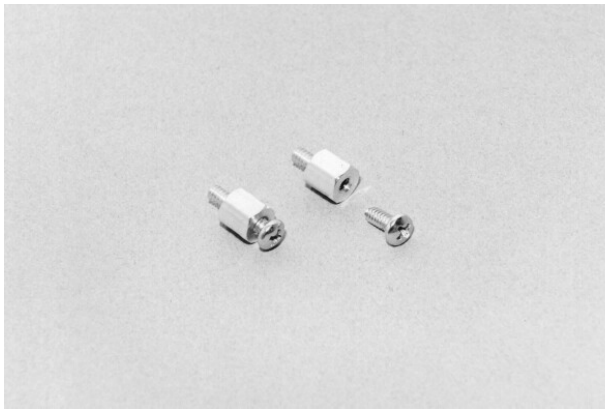
- Wheels
- Servos
- Board of Education
- Standoffs
- Battery Holder

Step #2: Mount spring standoffs to chassis

Parts Required:

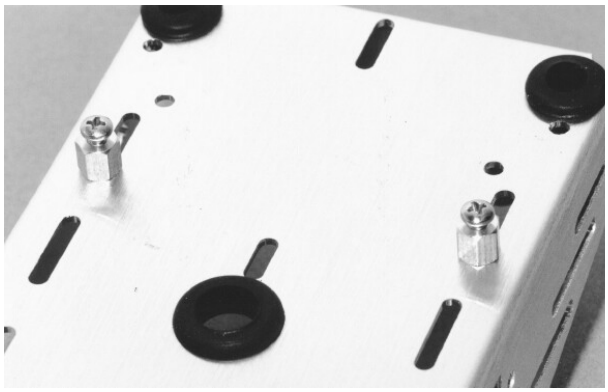
- (2) 4-40 x 1/4" screw
- (2) Stand-offs ¼" Male to Female
- (2) 4-40 nuts
- Boe-Bot Chassis



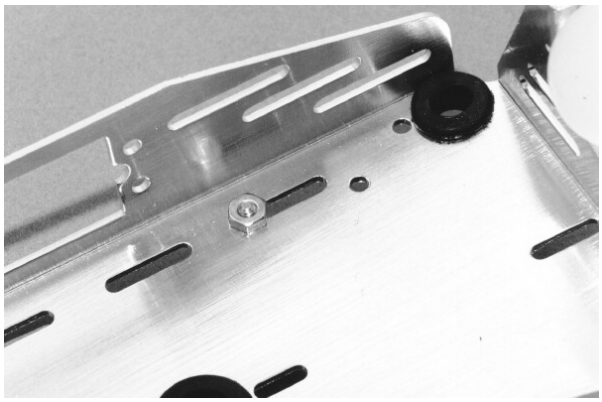


Screw the 4-40 x 1/4" screws into the top of the stand-offs. The screws will protrude. Later, they will be used to secure the Gripper's springs.

Install the standoffs into the slots in the Boe-Bot chassis as shown in the photo. Position them in the bottom of the slot, as shown.



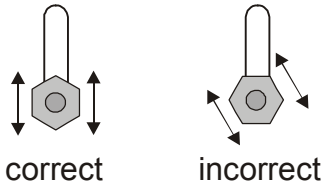
Secure using (2) 4-40 nuts.



Align the nut with the edges parallel to the slot, as shown in the "correct" illustration.

If the corner of the nut is sticking out, it will bump into the battery pack in Step #10.

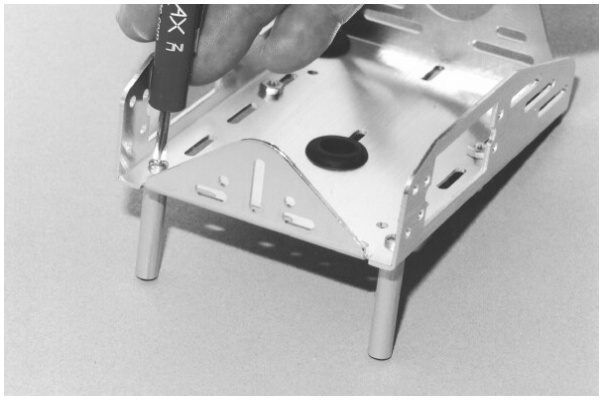
The easiest way to tighten is to hold the nut and turn the standoff.



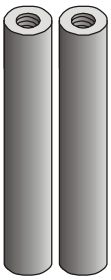
Step #3: Mount two standoffs to front of Boe-Bot

Parts Required:

(2) 4-40 x 1/4" screw



(2) Stand-offs 1.25"

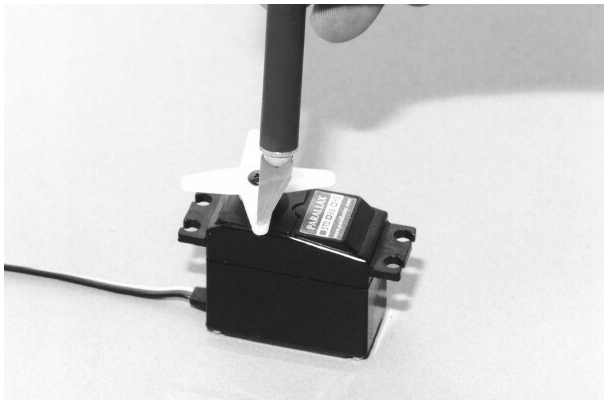


Use (2) 4-40 x 1/4" screws to attach the standoffs to the chassis as shown.

Step #4: Prepare servo horn

Parts Required:

- (1) Parallax servo
- (1) Hobby razor knife OR
- (1) 5/64" drill bit



One of the outermost holes in the servo horn must be made larger to accommodate a 2-56 screw.

The hole should be slightly smaller than the screw. This will ensure a snug fit by making threads as the screw goes in.

Using a hobby razor knife, enlarge the hole. Cut away material as shown in the photo, and then flip the horn over and cut material away from the other side, too. This will result in a good cylindrical hole.

Optionally, use a 5/64" drill bit to enlarge the hole.

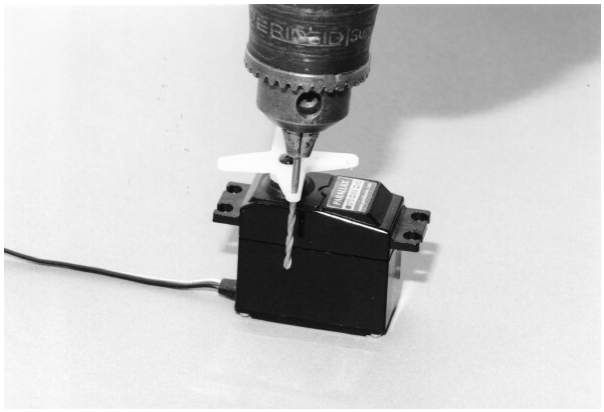
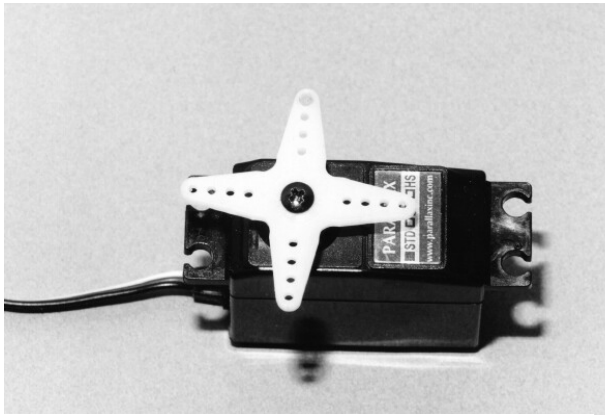
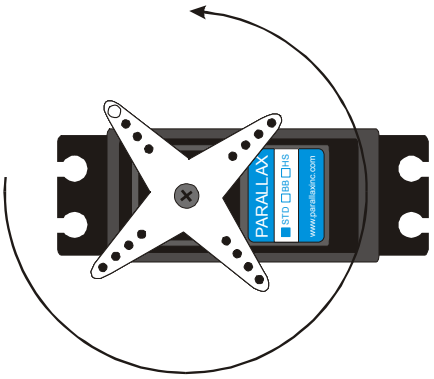


Photo shows servo horn after being cut/drilled out.



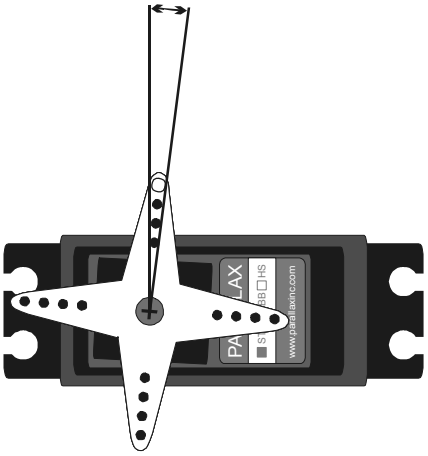
Step #5: Adjust position of servo horn

Hand rotate the servo horn counter-clockwise until it stops.

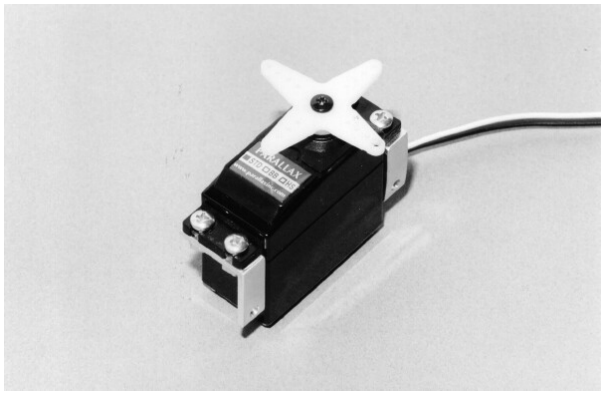


Remove the servo horn and remount as close to vertical as the splines will allow, with the large hole facing up as shown in the picture.

It works best to mount the servo a bit to the right of vertical, as shown, rather than a bit to the left.

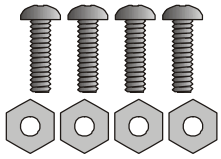


Step #6: Mount brackets on servo



Parts Required:

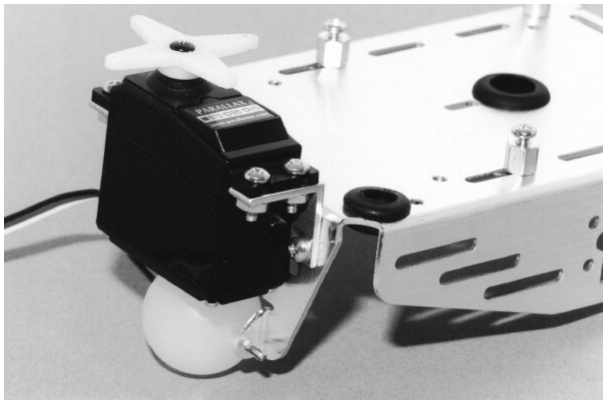
- (1) Parallax Servo
- (2) Servo Brackets
- (4) 4-40 x 3/8" screws
- (4) 4-40 nuts



Make sure the brackets are installed on the side of the servo shown in the photo.

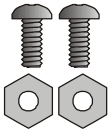
Place the 4-40 x 3/8" screws through the servo housing, and then place the brackets behind the housing. Secure with the 4-40 nuts.

Step #7: Mount servo to rear of chassis



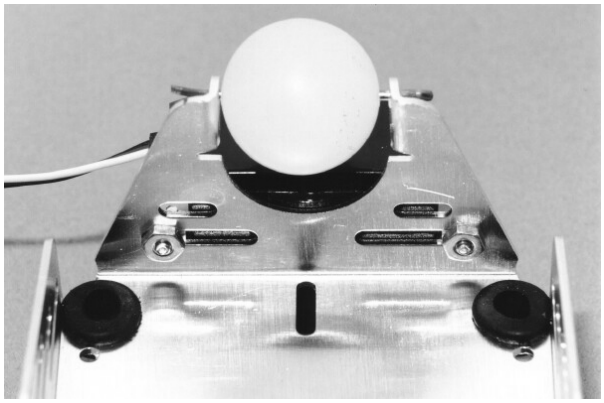
Parts Required:

- (2) 4-40 x 1/4" screws
- (2) 4-40 nuts



Align the holes in the brackets with the top slots in the chassis. Secure with (2) 4-40 x 1/4" screws and (2) 4-40 nuts. Check that the servo is mounted as shown in the photo, with the servo horn on the left.

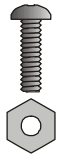
The screws will not protrude very much; they are just long enough to reach. This is necessary so they will not hit the battery pack when it is re-installed.



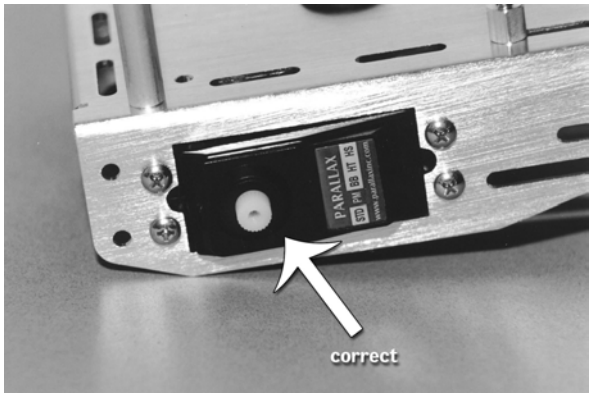
Step #8: Re-mount servos

Parts removed in Step #1[1]

- (8) 4-40 x 3/8" screw
- (8) 4-40 nuts or locknuts



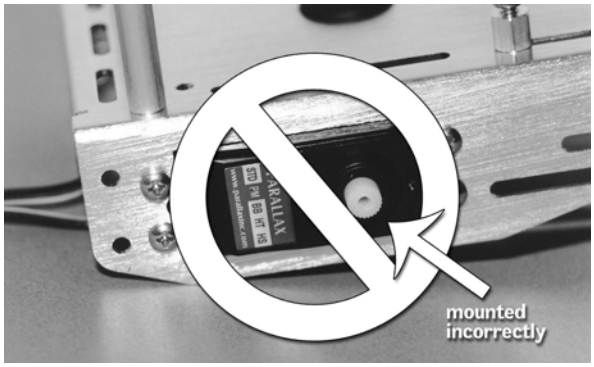
Using the 8 screws and nuts removed earlier, re-mount the wheels' servo motors to the Boe-Bot chassis. The output shaft should be closest to the front of the Boe-Bot – away from the tail wheel.



This will ensure the best balance when the Gripper is mounted and prevent the Boe-Bot from tipping.

[1] The parts listed came standard on Parallax Boe-Bots.

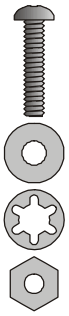
Servo incorrectly mounted.

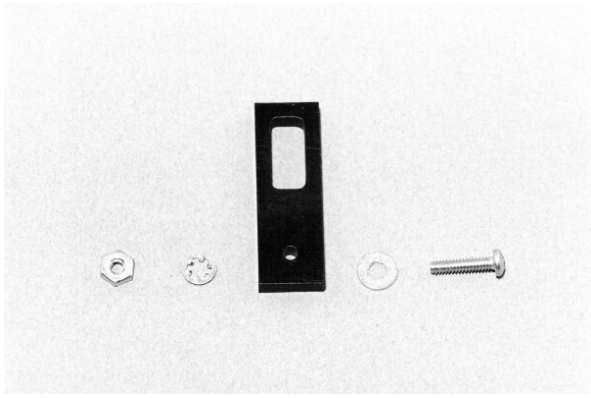


Step #9: Mount Gripper spoke guide rail to frame

Parts Required:

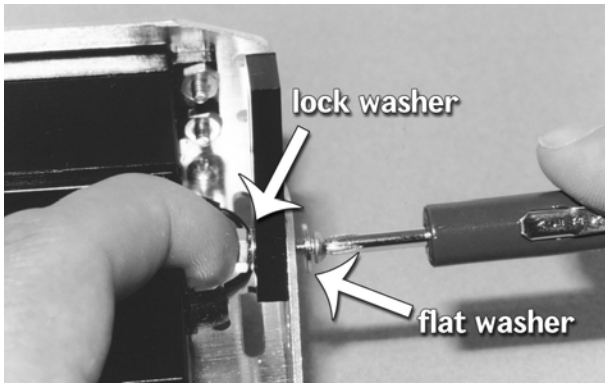
- (1) Gripper Spoke Guide Rail
- (1) 4-40 x ½" screw
- (1) 4-40 Flat Metal Washer
- (1) 4-40 Lock Washer (Star Type)
- (1) 4-40 Nut



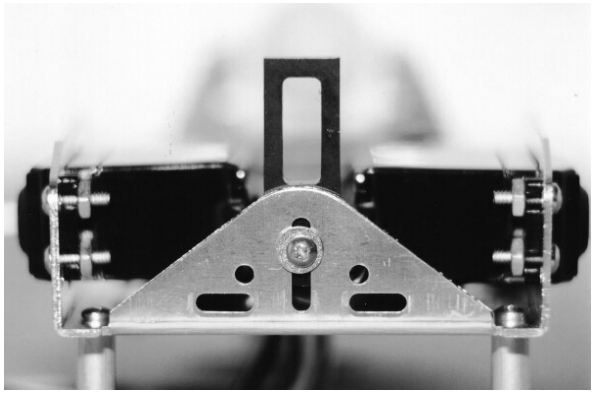


Place the flat metal washer onto the 4-40 screw. Insert the screw through the chassis, into the small hole in the Gripper spoke rail. Place the lock washer and 4-40 nut on the back.

The easiest way to assemble everything is to hold the lock washer and nut in place with your finger, then use the screwdriver to tighten.



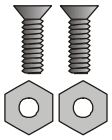
Adjust the Gripper spoke guide rail up or down until it is aligned as shown in the photo. The guide rail opening should be even with the metal chassis.



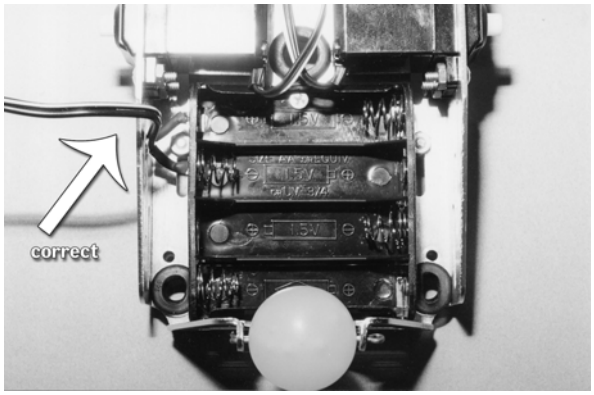
Step #10: Re-mount battery box

Parts removed in Step #1[2]

- (8) 4-40 x 3/8" flathead screw
- (2) 4-40 nuts or locknuts



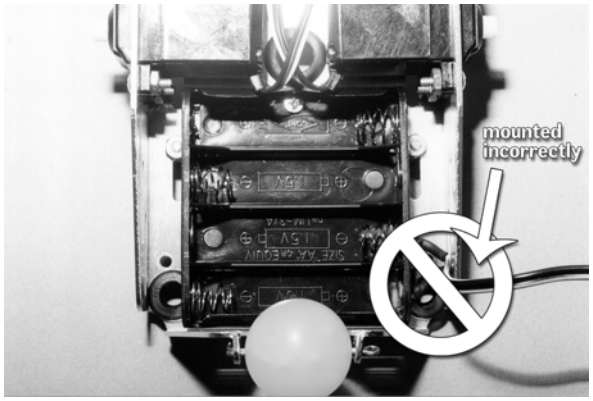
Orient the battery box so that the wires are closest to the wheel servos. In this photo, the wires are shown in the upper left.



Secure the battery box using the screws and nuts removed earlier.

Note that the nuts from the standoffs (installed in Step 2) should line up with the edges of the battery box.

Incorrect mounting.

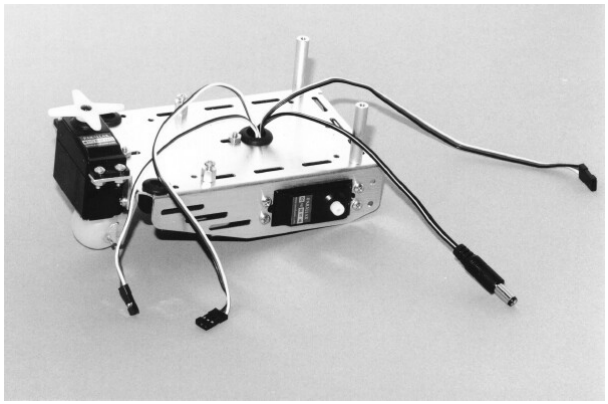


[2] The parts listed came standard on Parallax Boe-Bots.

Step #11: Route and Label Wires

Use cellophane tape to mark the servo connectors "Left", "Right", and "Gripper". This will be useful later when plugging in the electronics. The wheel servo that is visible in this photo is the "Right" servo.

Run the wires up through the hole in the chassis. It's easiest to run the battery pack wire first, as the connector is too large to fit otherwise.

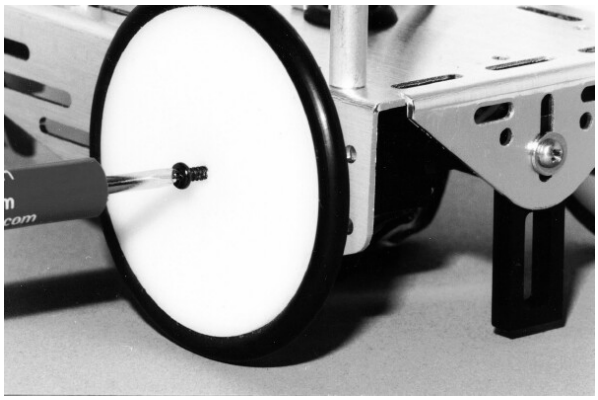


Step #12: Re-attach wheels

Parts removed in Step #1[3]

- (2) Small black screws

Re-attach the wheels using the small black screws removed earlier.



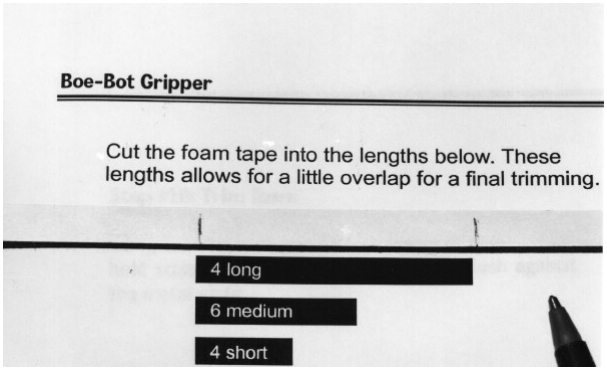
[3] The parts listed came standard on Parallax Boe-Bots.

Step #13: Install foam tape on Gripper plates

Parts Required:

- (2) Gripper Plates
- Foam Tape
- Scissors

Use the template below to mark the foam tape for cutting. You can align this document along the strip of foam, and mark the paper backing with a pen.



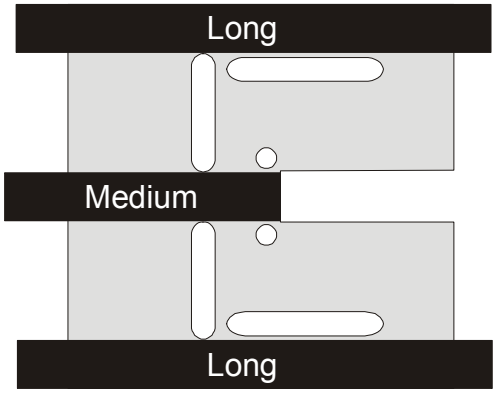
Cut the foam tape into the lengths on the right. These lengths allow for a little overlap for a final trimming.

4 long

6 medium

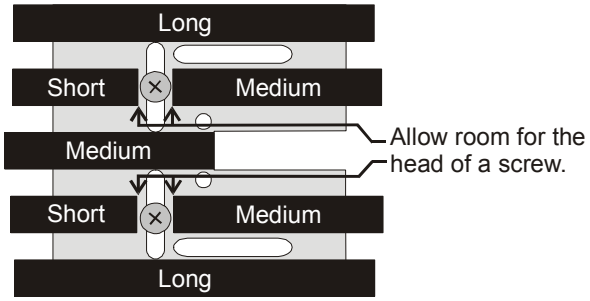
4 short

It's easier to center the foam tape if you stick down these pieces first.



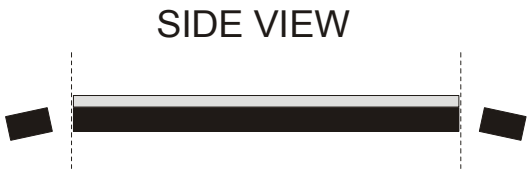
Attach the remaining foam tape.

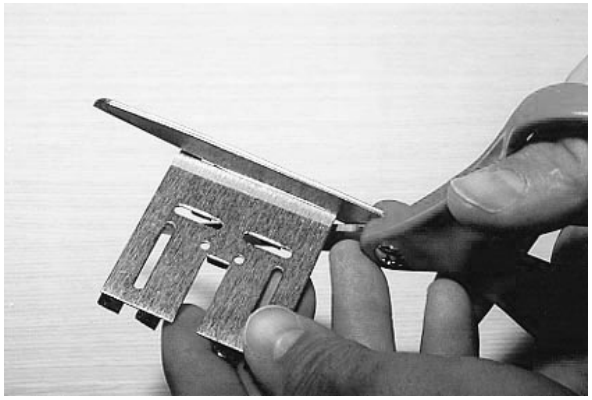
Allow room for the head of a screw



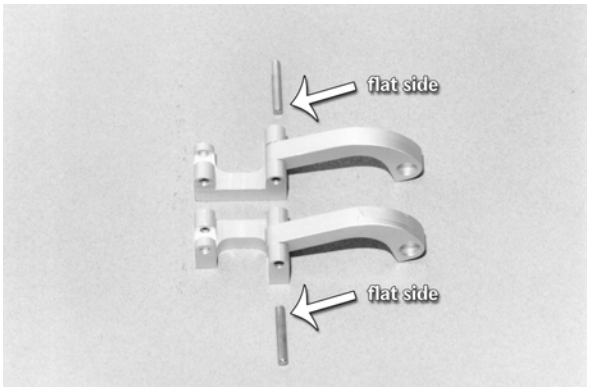
Step #14: Trim foam

Using scissors, trim excess foam. For a clean cut, hold scissors at a 90 degree angle and flush against the metal plate.



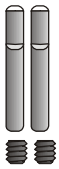


Step #15: Mount main Gripper crank to crank bracket



Parts Required:

- (2) Main Gripper Crank
- (2) Crank Brackets, L & R
- (2) Axle Pin #2

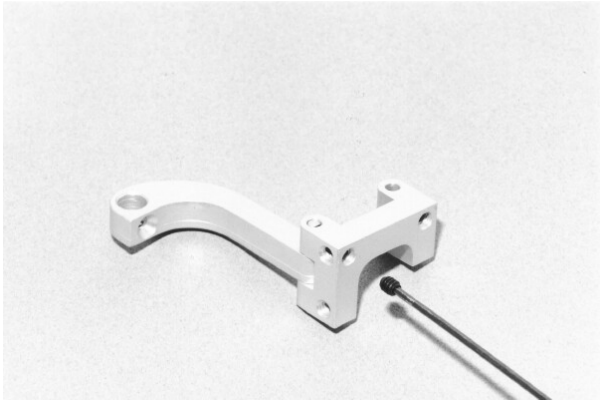


- (2) 4-40 x 1/8" Set Screw
- (1) 0.050 Allen wrench

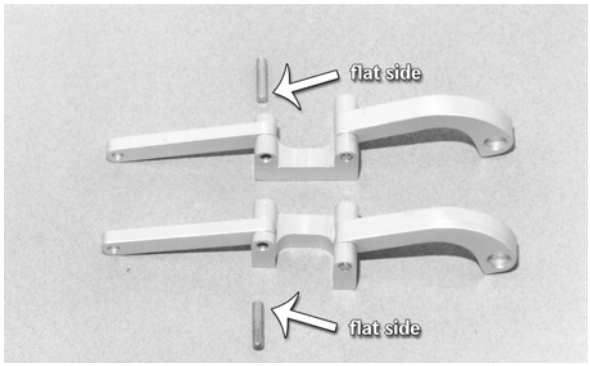
In this step you will build two parts, left and right, which are mirror images of each other.

Place the main Gripper crank into the crank bracket as shown in the photo. Insert axle pins into holes.

Ensure that the flat sides of the axle pins are facing the set screw holes. Using the Allen wrench, place the set screws in the holes and tighten until the screw contacts the pins. You may have to repeatedly back off the set screw and advance it a little bit at a time, because the set screw may be a bit tight.

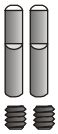


Step #16: Mount Gripper support crank to crank bracket.



Parts Required:

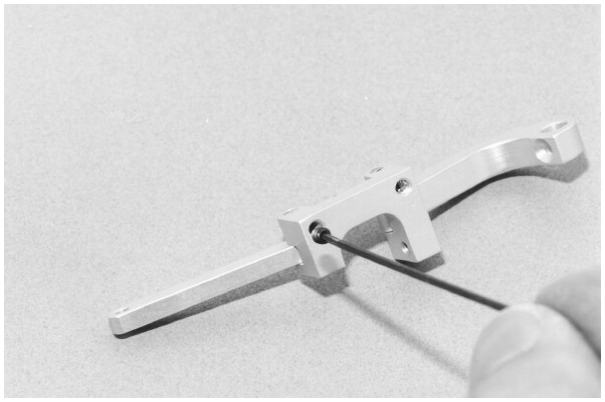
- (2) Gripper Support Crank
- (2) Axle Pin #1



- (2) 4-40 x 1/8" Set Screw
- (1) 0.050 Allen wrench

Place the Gripper support crank into the remaining slot in the crank bracket. Then insert the axle pins into the holes.

Again, make sure that the flat sides of the axle pins are facing the set screw holes.



Using the Allen wrench, place the set screws in the holes and tighten up against the axle pins.

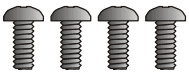
Remember, you may have to repeatedly back off the set screw and advance it a little bit at a time, because the set screw may be a bit tight.

Both the main Gripper crank and the Gripper support crank should move smoothly and freely, without tightness or binding. If not, lubricate with a few drops of light machine oil.

Step #17: Mount crank assembly to Gripper plates

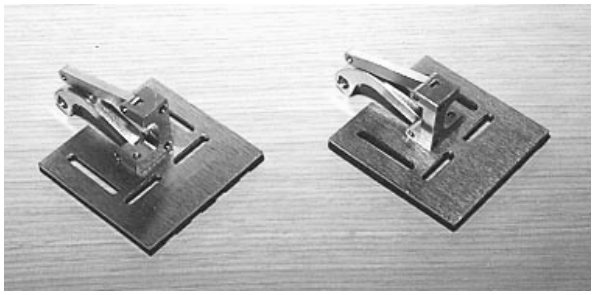
Parts Required:

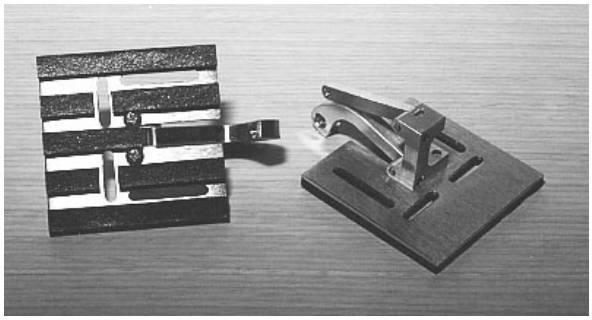
- (2) Gripper Plates
- (4) 4-40 x ¼ " screws



Place the crank assembly onto the Gripper plate, as shown in the photo. Insert the 4-40 screws through the plate into the crank assembly and tighten.

Note: The Gripper plates are identical, so it does not matter which assembly goes on which plate.

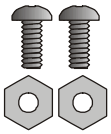


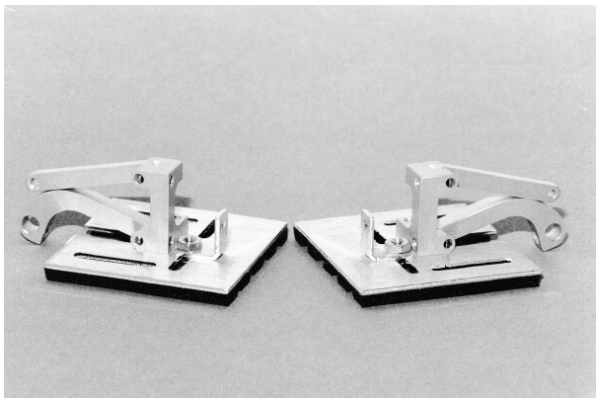


Step #18: Mount brackets to Gripper plate

Parts Required:

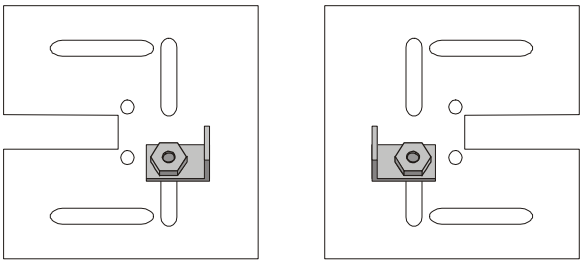
- (2) Metal 90 degree bracket for SSIR
- (2) 4-40 x ¼" screws
- (2) 4-40 nuts



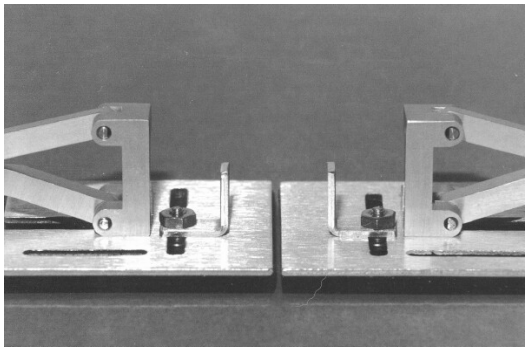


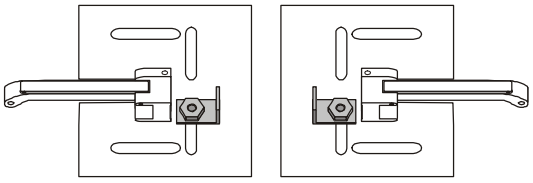
Position the 90 degree bracket over the slots shown.

Insert the screw from the foam side of the Gripper and secure with the 4-40 nut.



Line the edge of the 90 degree bracket up with the edge of the crank bracket





Step #19: Mount SSIR sensors to brackets

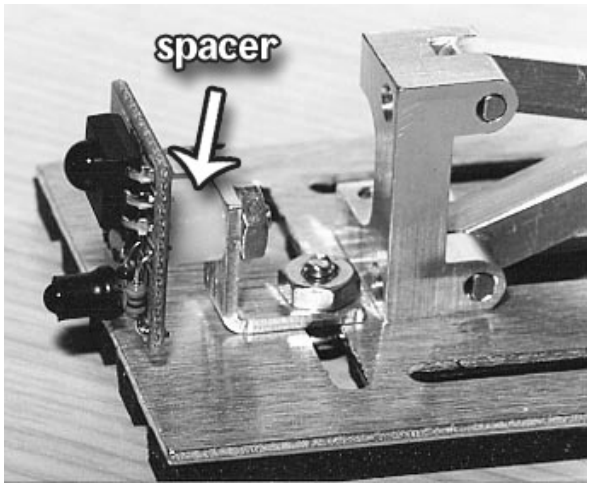
Parts Required:

- (2) SSIR sensor
- (2) 4-40 x 1/2" screw

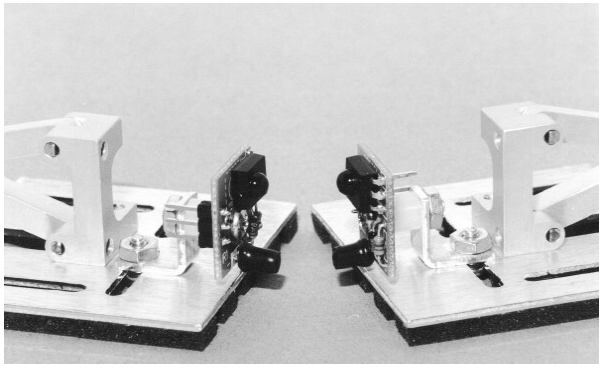
- (2) Spacer .25" Nylon

- (2) 4-40 nuts

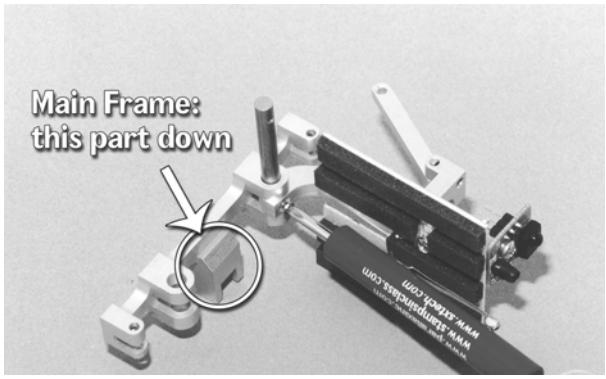
Put the screw through the component side of the SSIR sensor, and then slide the nylon spacer on the back. Secure sensor to L-bracket using 4-40 nut.



Repeat with the other sensor. Orient the sensors as shown in the photo.



Step #20: Attach left Gripper assembly to main frame



Parts Required:

- (1) Main frame
- (1) Left Gripper assembly
- (1) Crank axle (Brass Rod)
- (1) #2 x ¼" Philips flat head screw
- (1) Axle Pin #1

- (1) 4-40 x 1/8" set screws
- (1) Allen Wrench .050"

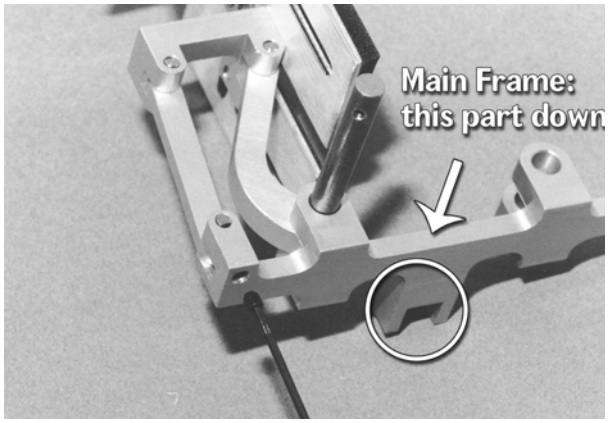
The right and left Gripper assemblies are not identical. To identify which is the left assembly, place a Gripper assembly into the main frame as shown in the photo and try to fit both crank arms into their places in the main frame. If one of the arms hits the frame, this is the incorrect Gripper assembly.

Note that the holes drilled in the crank axle (brass rod) are not both the same distance from the end of the rod. Orient the rods so that the hole closest to the end is up.

Slide the brass rod into the main frame, through the main Gripper crank. Secure with the #2 x ¼" Philips flat head screw.

Place the Gripper support crank into its slot in main frame.

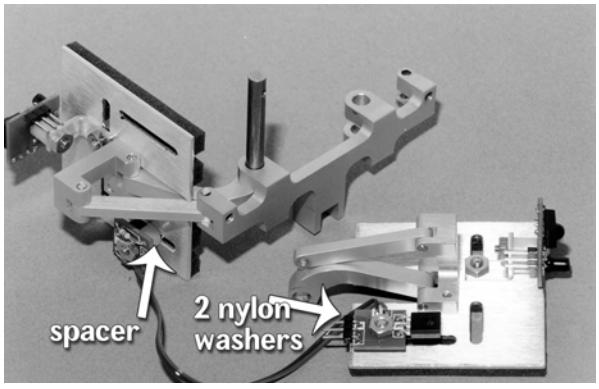
Insert the axle pin with the flat side up. Secure with the 4-40 x 1/8"set screw, making sure the set screw contacts the flat side of the axle pin.



Step #21: Attach Gripper IR Sensor

Parts Required:

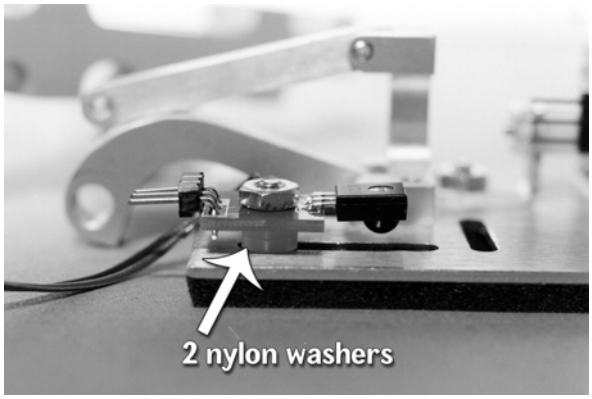
- (1) Gripper IR Sensor
- (1) 4-40 x 3/8" Screw
- (2) Nylon Washer (Screw Size #4)
- (1) 4-40 x ½" Screw
- (1) Spacer .25" Nylon
-
- (2) 4-40 Nut



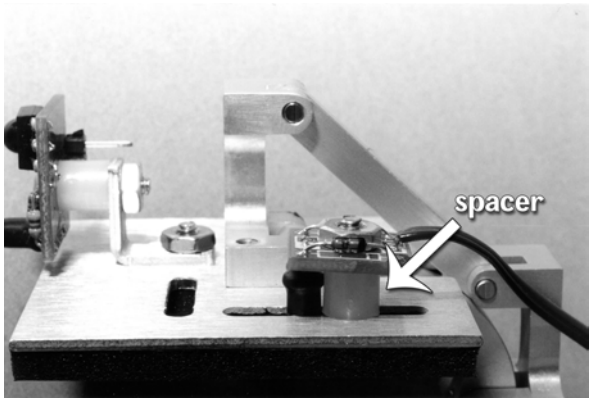
Attach the IR detector to the right Gripper assembly as shown in the photo. The right Gripper is the one that is not yet attached to the main frame.

Place the 4-40 x 3/8" screw through the right Gripper plate from the foam side. Put two nylon washers (Screw Size #4) on the back of the screw, followed by the sensor, as shown in the photo. Secure with the 4-40 nut.

Position the sensor such that the detector is visible through the slot.



Attach the IR emitter to the left Gripper plate.
Place the 4-40 x 1/2" screw through the left Gripper plate from the foam side. Put the .25" nylon spacer on the back of the screw, followed by the sensor, as shown in the photo. Secure with the 4-40 nut.

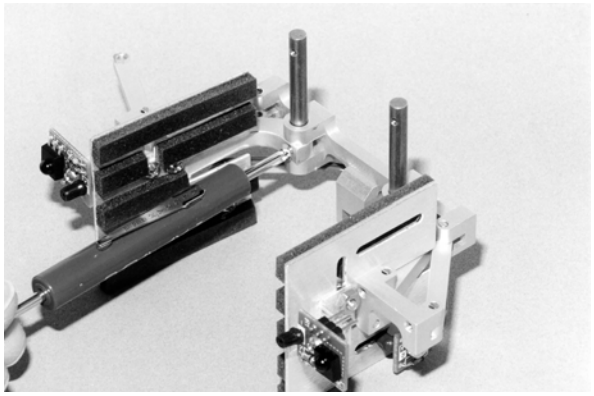


Step #22: Attach Right Gripper Assembly

Parts Required:

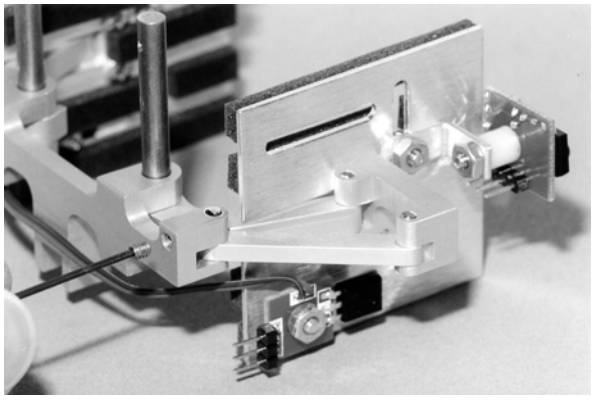
- (1) Crank axle (Brass Rod)
- (1) #2 x ¼" Philips flat head screw
- (1) Axle Pin #1

- (1) 4-40 x 1/8" set screw
- (1) Allen Wrench .050"



Place the right Gripper assembly into the main frame. Slide the crank axle into the main frame, through the main Gripper crank. Secure with the #2 x ¼" Philips flat head screw.

Place the Gripper support crank into its slot in main frame. Insert the axle with the flat side up. Secure with 4-40 x 1/8"set screw, making sure the set screw contacts the flat side of the axle pin.



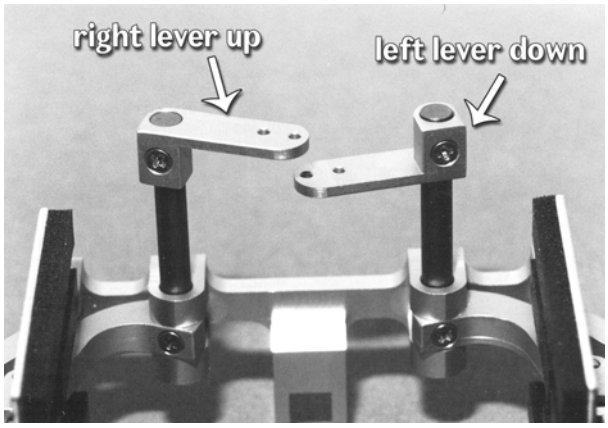
Step #23: Mount lever arms to crank axle

Parts Required:

- (2) Lever arms
- (2) 2-56 x ¼" Philips flat head screw 

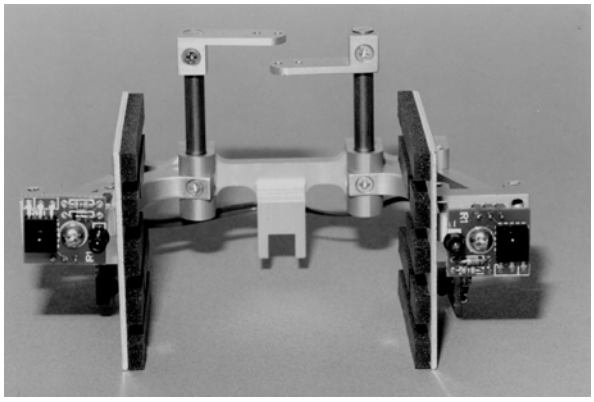
Place the lever arms onto the crank axles (brass rods) as shown in the photo. Note that the lever arms are flipped over in respect to each other. That is, one has the lever arm at the top of the crank axle, and the other is upside down.

Note: The lever arms are identical, so it does not matter which arm goes on which crank axle.



Secure the lever arms to the crank axle using the 2-56 x ¼" Philips flat head screws.

This finishes the Gripper assembly. The Gripper should open and close smoothly without any tightness or binding. If not, lubricate with a few drops of light machine oil.

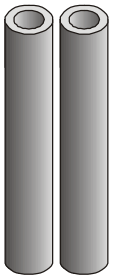


Step #24: Mount support gantry to hollow standoffs to rear BOE-Bot standoffs

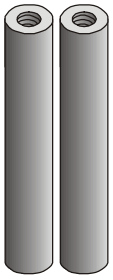
Parts Required:

- (1) Gripper Support Gantry
- (2) 4-40 x 1 ¾ screw

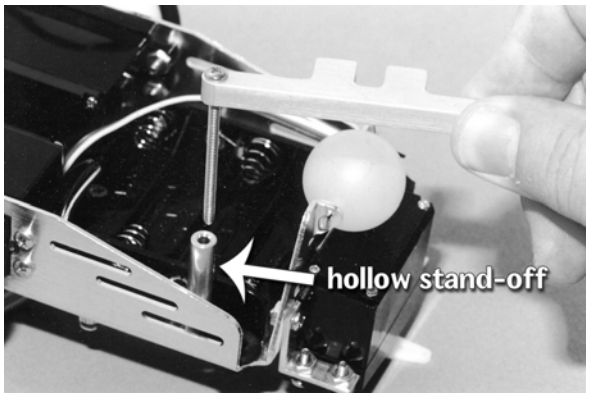
- (2) Stand-off 1.25" Hollow



- (2) Stand-off 1.25"

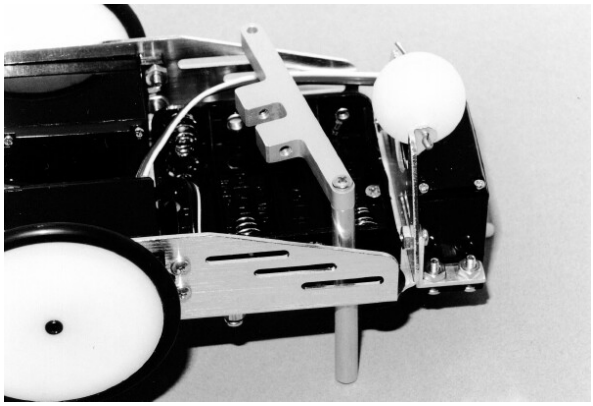


Put the long 4-40 x 1 ¼ screw through the Gripper support gantry and the hollow standoff, through the hole in the Boe-Bot chassis.

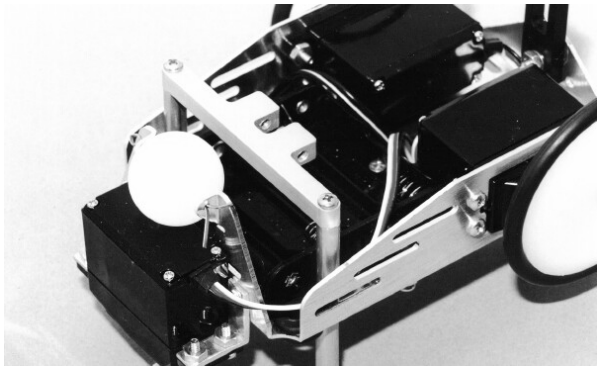


Screw the threaded standoff onto the protruding 4-40 x 1 ¼ screw. Do not tighten.

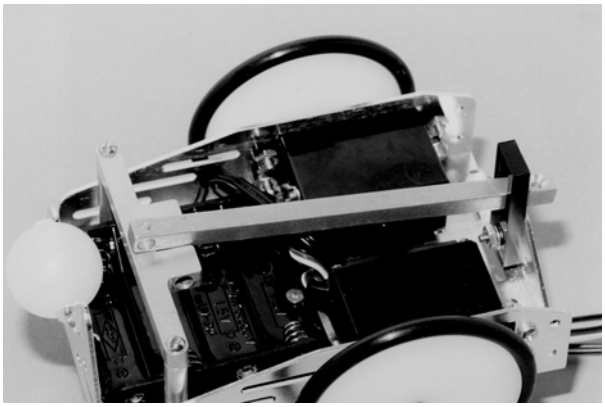
Check that the Gripper support gantry is facing as shown in the photograph.



Install the second set of standoffs. You may have to angle the support gantry to get them in place. When all standoffs are installed, tighten the screws.



Step #25: Attach Gripper spoke to support gantry with set screws

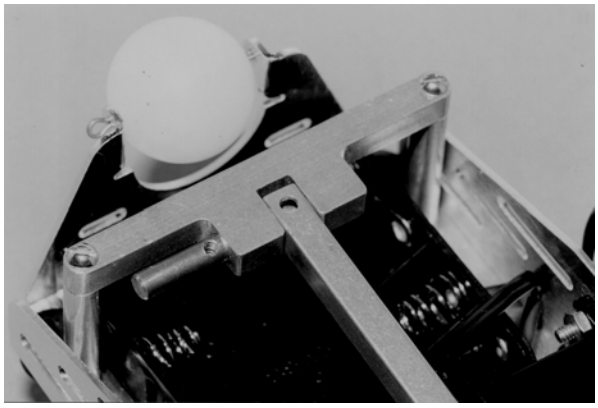


Parts Required:

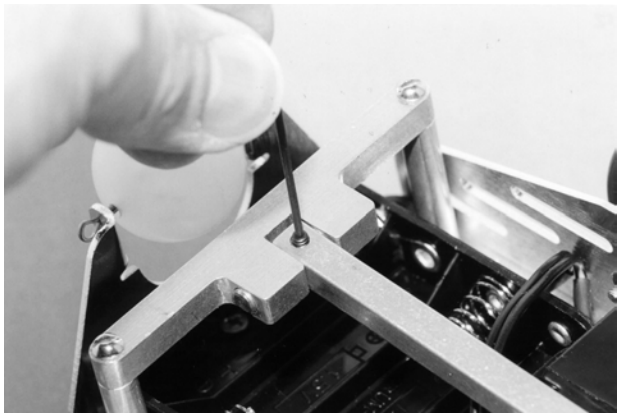
- (1) Gripper Spoke
- (1) Spoke axle
- (1) 4-40 x 1/8" set screw 
- Allen Wrench .050"

Slide the Gripper spoke through the Gripper spoke rail (black Delrin piece) and into the notch in the Gripper support gantry.

Align the hole in the spoke axle with the hole in the Gripper spoke. Slide the spoke axle through the Gripper support gantry and Gripper spoke.

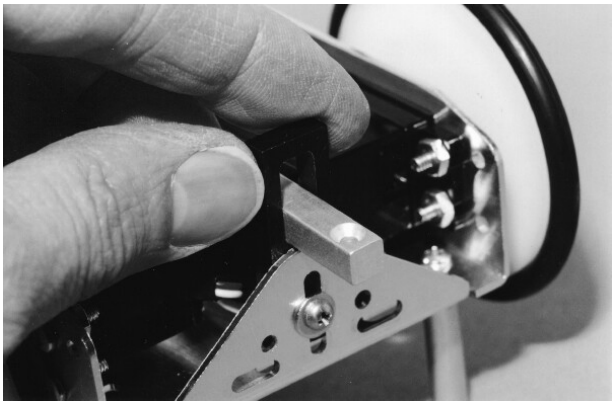


Secure with the 4-40 x 1/8" set screw. It will thread all the way into the spoke axle.



Step #26: Adjust Gripper spoke rail

The Gripper spoke should pivot up and down freely, but may bind against the Gripper spoke guide rail. If so, loosen the screw and adjust the guide rail side to side, then retighten.



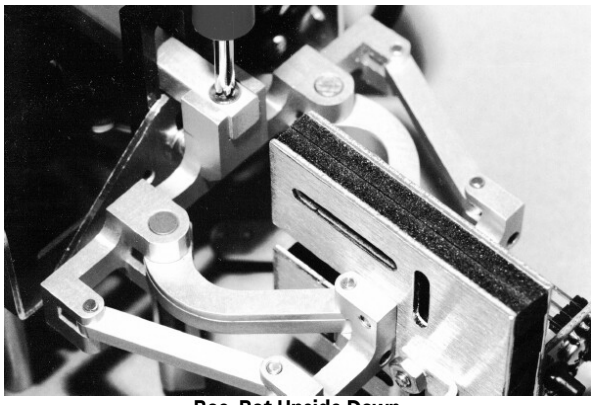
Step #27: Attach Gripper

Parts Required:

- (1) Gripper Assembly
- (1) 4-40 x 3/8" Phillips Flat Head screw

Place the Gripper assembly onto the Gripper spoke. Route the wire above the spoke, between the main frame and the Boe-Bot chassis.

Turn the Boe-Bot upside down and install the 4-40 x 3/8" Phillips flat head screw.



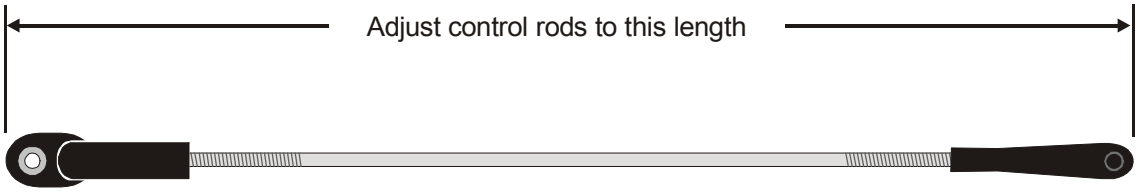
Boe-Bot Upside Down

Step #28: Prepare control rods
Parts Required:

- (2) 4.5" control rod
- (2) 2-56 swivel ball links
- (2) 2-56 metal clevis

Screw the ball link on one end of the control rod and the clevis on the other. You may need to grip the control rod with a pair of pliers and twist the ends on. If so, place the pliers on the smooth part of the rod, not the threads.

Adjust the overall length of the control rod to the length shown in the template below by screwing the clevis and ball links in or out.



Step #29: Bolt control rods to servo horn

Parts Required:

(2) Control rods
(1) 2-56 Phillips Flat head Screw ½"

(1) 2-56 nut

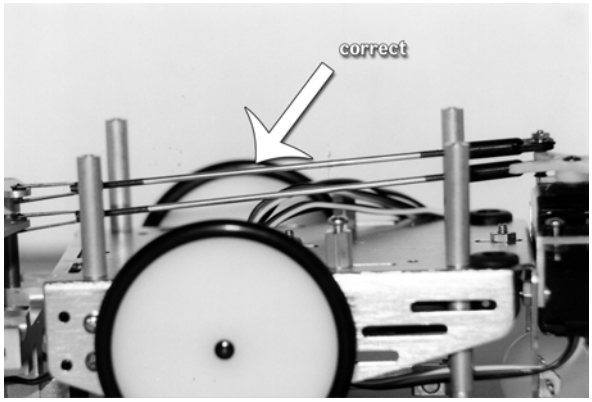
Rotate the servo horn fully clockwise so that the hole is accessible. Place the 2-56 Phillips flat head screw through one control rod's ball link and screw it into the servo horn from below as shown. The screw will

cut threads into the soft plastic as you go.

Rotate the servo horn back, and then place the second control rod on top of the servo horn. Secure with the 2-56 nut.

Important:
Make sure you have mounted one control rod beneath the servo horn, and one above, as shown in the photo.

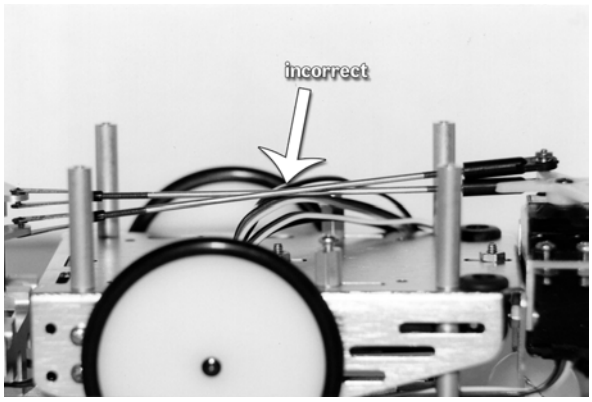
Step #30: Attach control rods to Gripper



The clevis ends of the control rods attach to the Gripper's lever arms. There are two holes in each lever arm. Insert the clevis pins into the holes in the ***tip*** of the lever arms. To insert the pin, pry the metal clevis apart with a flat-blade screwdriver.

The bottom control rod attaches to the lever arm that is lower, and the upper control rod into the lever arm that is higher.

Incorrect control rod routing!!



Step #31: Fine Tune Control Rods

Fine tune the length of the control rods so the two Gripper plates come together evenly.

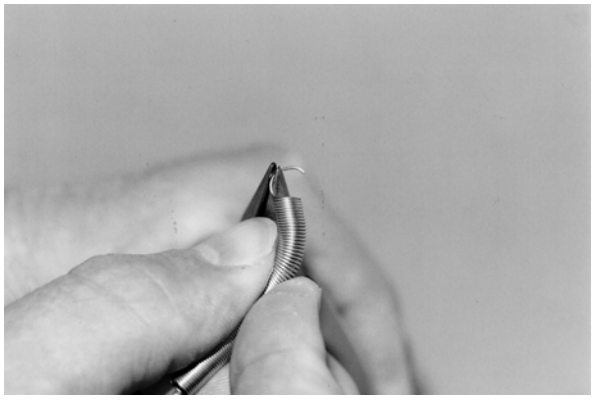
Using a flat-blade screwdriver, pry the metal clevis off the lever arm, screw it in or out a few turns as needed, then replace and try again. Repeat until the plates line up evenly.

Step #32: Attach springs to Gripper

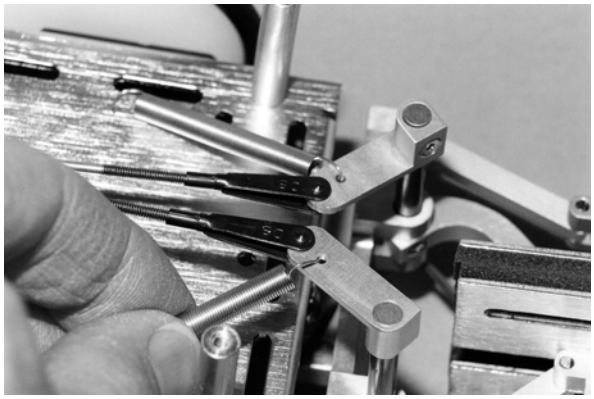
Parts Required:

- (2) Springs

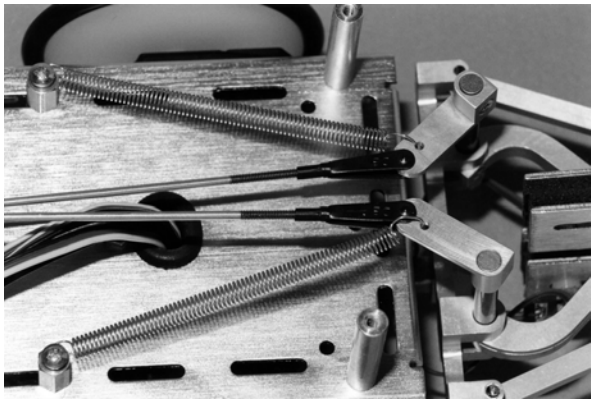
Use needle-nose pliers to slightly open the loop in one end of each spring.



Hook one end of the springs into the holes in the Gripper's lever arm.



Stretching the springs, hook the other end over the screw protruding from the standoff on the Boe-Bot chassis.



Chapter #2: Tuning and Troubleshooting the Gripper Hardware

The Gripper is now assembled and mounted. It is time to fine-tune it for smooth opening and closing operation.

Turn the servo by hand to check the operation of the Gripper. It should open and close fully as the servo is turned through its full range of motion. When the Gripper is fully closed, the edges of the Gripper plate should align evenly along all edges. As the Gripper opens, the entire Gripper assembly should lower, and as it closes, the entire assembly should raise.

Hardware Troubleshooter

Possible problems are shown below, with solutions appearing in the following table.

Problem:	**Possible Cause:**								
	1	2	3	4	5	6	7	8	9
Gripper binds while opening and closing		X	X	X	X	X			
Gripper doesn't fully close		X	X	X	X	X			
Gripper doesn't fully open	X	X	X	X	X	X			
Gripper doesn't fully lower							X	X	
Gripper doesn't fully raise							X	X	
Gripper plates don't line up					X				
Boe-Bot tips toward the front									X

Solutions

1. Foam sticking together

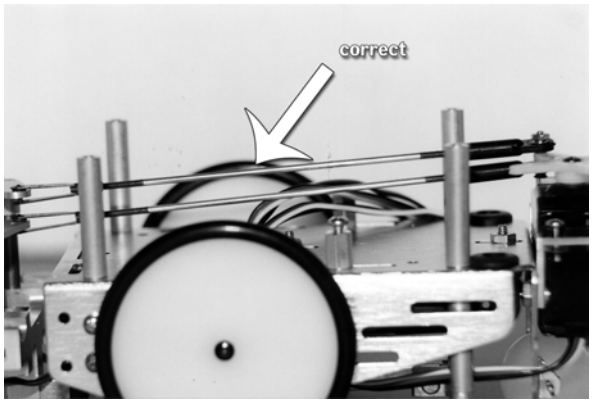
The foam strips on the Gripper may be sticking to each other. Use rubbing alcohol to swab off the foam, and place a small piece of paper between the Gripper plates when the Gripper is not used. This will help absorb some of the stickiness from the foam strips.

2. Control rod incorrectly mounted on servo horn
   See Step #29

The control rods should be placed on the servo horn such that one is beneath the horn, and one is above.

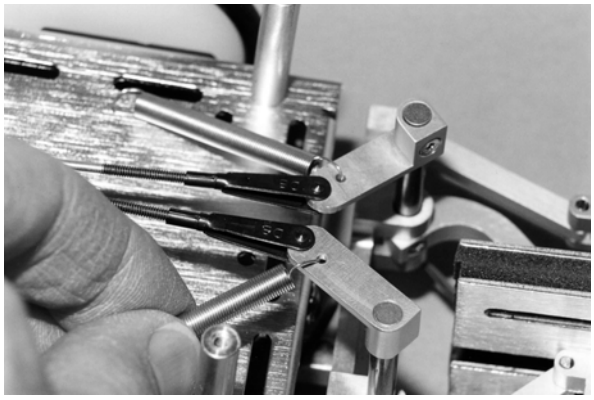
3. **Control rods improperly connected to lever arms See Step #30**

   The bottom control rod should be in the lever arm that is lower, and the upper control rod in the lever arm that is higher.



4. **Spring in wrong hole of lever arm See Step #32**

   There are two holes in the Gripper lever arm. The control rod attaches to the hole at the tip of the lever arm. The spring attaches to the other hole.

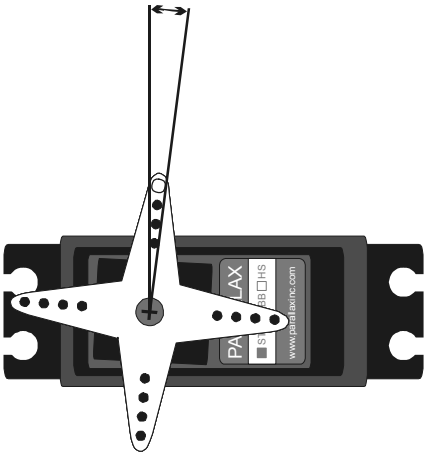


5. **Length of control rods need adjusting See Steps #28 and #31**

   You may have to fine-tune the length of the control rods. Using a flat-blade screwdriver, pry the metal clevis off the lever arm, screw it in or out a few turns as needed, then replace and try again. Repeat until Gripper opens or closes fully.

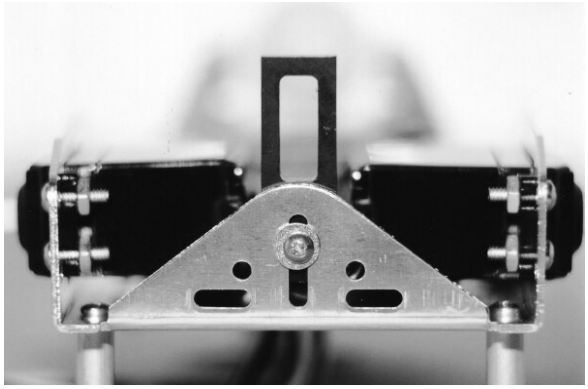
6. **Servo horn improperly placed on servo**
   See Step #5

   The servo horn may have been mounted too far off the vertical, or the position may need to be adjusted slightly.



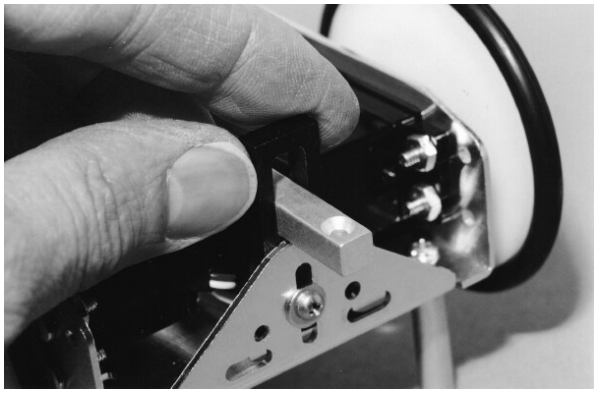
7. **Gripper spoke guide rail too far up or down**
   See Step #9

   The Gripper spoke guide rail opening should be even with the metal chassis.



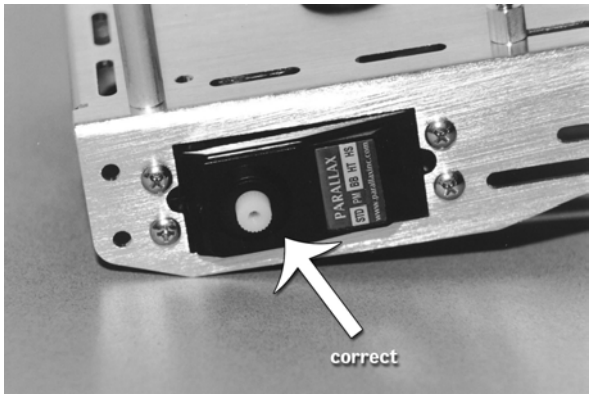
8. **Gripper spoke guide rail needs adjusting left or right See Step #26**

   The Gripper spoke may be hitting the guide rail, preventing it from moving up and down. Adjust the Gripper spoke rail left or right.



9. **Wheels servos installed backwards See Step #8**

   The output shafts should be closest to the front of the Boe-Bot – away from the tail wheel. This will ensure the best balance when the Gripper is mounted, preventing the Boe-Bot from tipping.



Electronics Assembly Instructions

Step #1: Re-attach Board of Education.

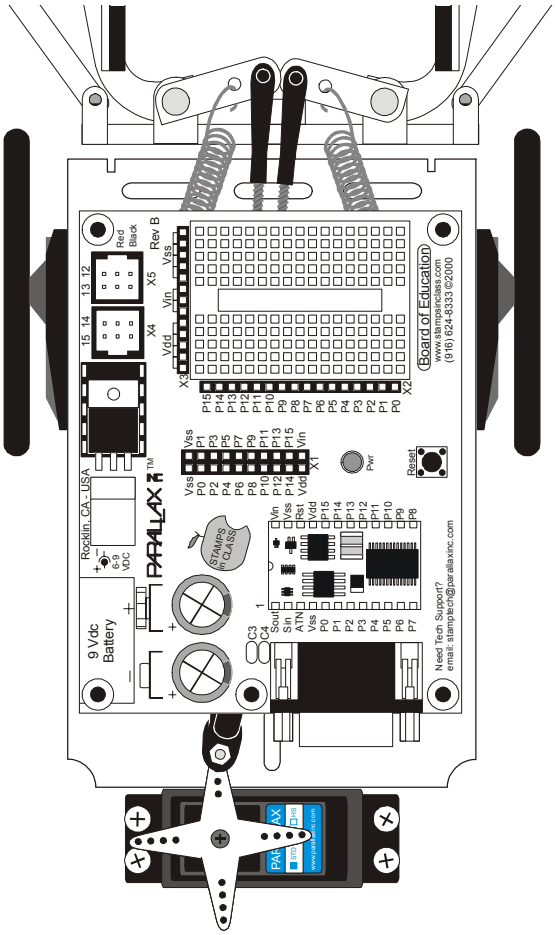
Parts removed in Step #1[4]

- (4) 4-40 x 1/4" screw

Route all the servo wires underneath the springs and toward the upper left standoff. Wrap the wires once around the standoff to take up some extra slack.

Route the battery cable toward the back of the Boe-Bot.

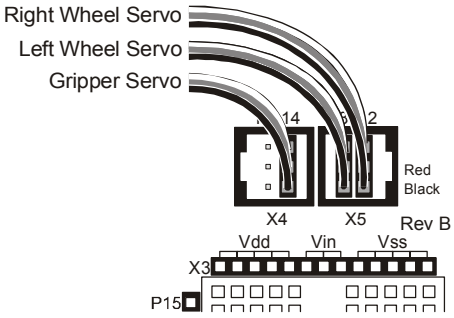
Re-attach the Board of Education to the Boe-Bot using the four 4-40 x ¼" screws removed earlier. The white breadboard goes toward the front of the Boe-Bot, over the wheels.



[4] The parts listed came standard on Parallax Boe-Bots.

Step #2: Connect Servos to Board of Education

Plug the right servo into servo port 12, the left servo into servo port 13, and the Gripper servo into servo port 14. Follow the color code as printed on the Boe-Board, with the black wires closest to the breadboard.



Step #3: Wire breadboard

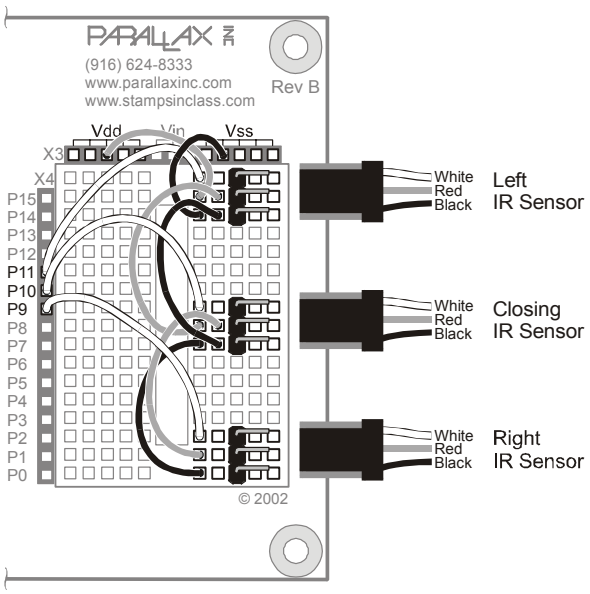
Parts Needed:

(3) Right angle 3-pin headers
(9) Jumper Wires

Insert the 3-pin headers as shown, with the long ends into the breadboard.

Wire the 3-pin headers as shown. Each header has three connections -- to Vdd, to Vss, and to a BASIC Stamp pin.

Header:	To BASIC Stamp Pin:
Left IR Sensor	P11
Closing IR Sensor	P10
Right IR Sensor	P9



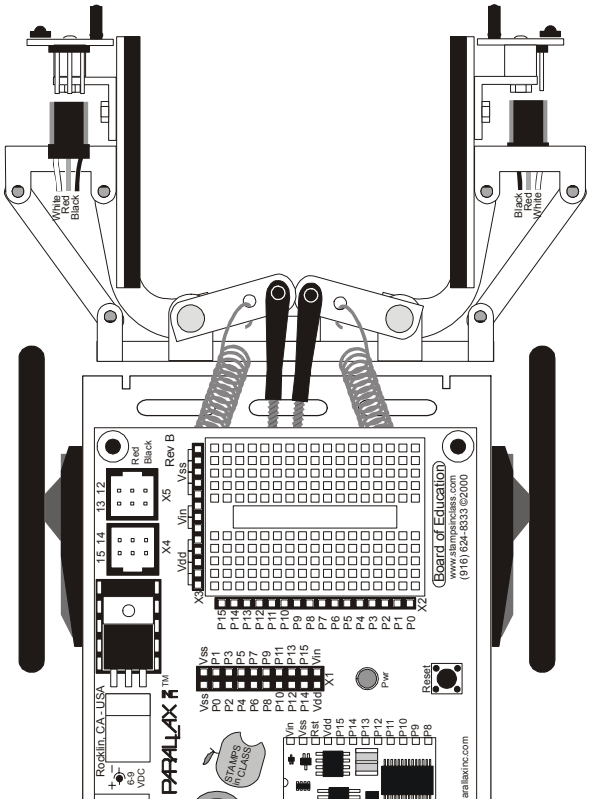
Step #4: Connect sensors to breadboard

Parts Needed:

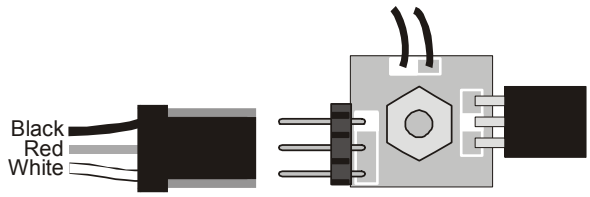
(3) 10" Servo Extension Cables

Plug servo extension cables into the left and right IR sensors, following the color code shown in the drawing.

Note that the black wire is towards the Gripper for both the left and right sensors.

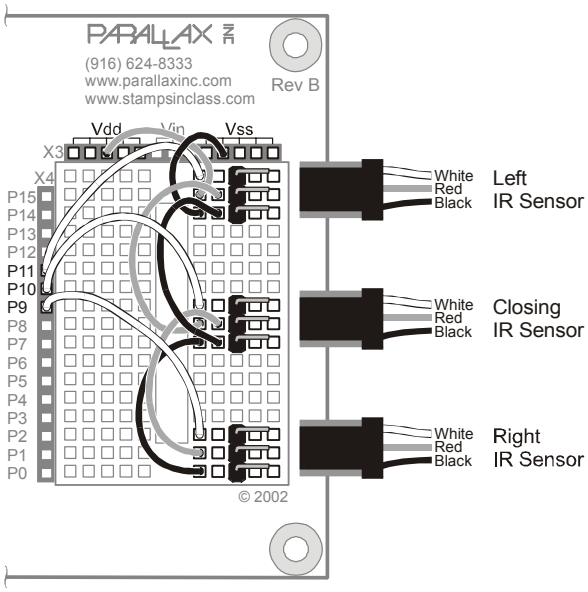


Plug the middle servo extension cable into the closing IR sensor, following the color code shown.



Connect the servo extension cables to the 3-pin headers. Follow the color code shown.

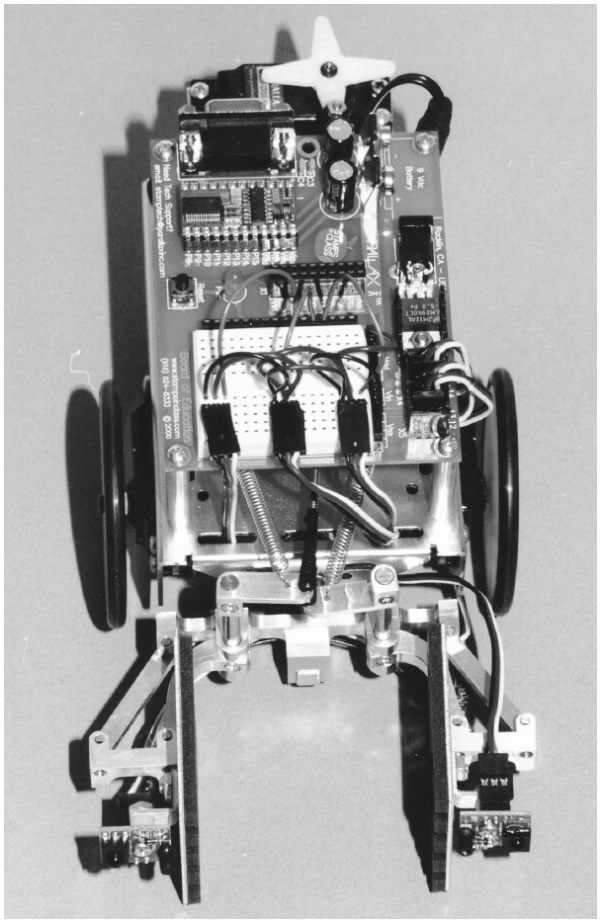
You may wish to wrap the cables around the standoffs to take up some slack. Alternately, route the cables through the slots in the Boe-Bot chassis, and store excess cable underneath the wheel servos.



Step #5: Install batteries

Re-install the 4 AA batteries.

Congratulations! Your Boe-Bot Gripper is fully assembled.



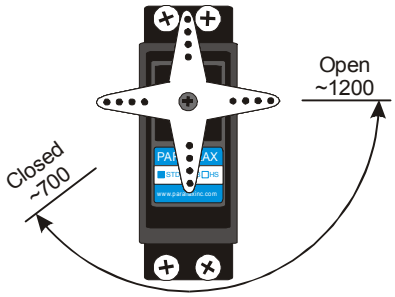
Chapter #3: Tuning the Gripper in Software

Open and Close Gripper

Now that the Gripper's mechanics are working smoothly, it's time to tune the Gripper's software. The PBASIC PULSOUT command is used to position the Gripper servo motor[5]. We must determine the PULSOUT period needed to make the Gripper open and close fully. The approximate values are shown below, but due to variation in mechanical setup and servo characteristics, your Gripper may open and close at slightly different values.

Approximate PULSOUT Values for Open/Close

Gripper Position	Approximate PULSOUT Period
Closed	700
Open	1200



The first program will close the Gripper.

- Turn on the Boe-Bot. The green LED on the Board of Education should light.
- Connect the serial cable between the Boe-Bot and your PC.
- Manually move the Gripper to the fully open position.
- Open the the BASIC Stamp Editor.
- Enter the program Gripper Program 2.1: Close Gripper.bs2, shown below.
- Edit the code to reflect the servos you are using. Set `NewGripServo = 1` if you are using the newer Standard Servo, or `NewGripServo = 0` for the older STD servo. You can comment out the incorrect declaration.

```
#DEFINE NewGripServo = 1                    ' Newer standard servo
'#DEFINE NewGripServo = 0                   ' Older STD servo
```

[5] Refer to"What's A Microcontroller?" for a complete discussion on servo positioning.

- Download the code by using Run from the Run menu, or pressing the F9 key.

```
' -----[ Title ]-----------------------------------------------------------
' Gripper Program 2.1: Close Gripper
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]---------------------------------------------
' This program closes the gripper
' Manually move gripper to open position before beginning

' -----[ I/O Definitions ]--------------------------------------------------

GripperServo   PIN     14                    ' Gripper Servo on P14

' -----[ Constants ]-------------------------------------------------------
' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                     ' Newer standard servo
'#DEFINE NewGripServo = 0                    ' Older STD servo

#IF (NewGripServo = 1) #THEN                 ' For newer Standard gripper servo
  StepSize     CON     20                    ' Step size for gripper servo
#ELSE                                        ' For older STD gripper servo
  StepSize     CON     10                    ' Step size for griper servo
#ENDIF

CloseGripper   CON     700                   ' Decrease to close more
OpenGripper    CON     1200                  ' Increase to open more

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

pulsewidth     VAR     Word

' -----[ Main Code ]-------------------------------------------------------

FOR pulsewidth = OpenGripper TO CloseGripper STEP StepSize
  PULSOUT GripperServo, pulsewidth
  PAUSE 20
NEXT
```

If the Gripper doesn't close fully, edit the program and decrease the value of **`CloseGripper`** from 700 to 650. Manually open the Gripper, and run the program again. Continue decreasing the value until the Gripper closes fully. Make a note of the final **`CloseGripper`** value.

Now it's time to fine-tune the Gripper open value.

- Start with the Gripper in the fully closed position.

- Enter the program named Gripper Program 2.2: Open Gripper.bs2, shown below.
- Edit the program. Modify the **`ClsoeGripper`** value in the program to match the value you found for your Gripper.
- Edit the code to reflect the servos you are using. Set **`NewGripServo = 1`** if you are using the newer standard servo, or **`NewGripServo = 0`** for the older STD servo. You can comment out the incorrect declaration.

```
#DEFINE NewGripServo = 1                    ' Newer standard servo
'#DEFINE NewGripServo = 0                   ' Older STD servo
```

- Download the code by using Run from the Run menu, or pressing the F9 key.

```
' -----[ Title ]-------------------------------------------------------------
' Gripper Program 2.2: Open Gripper
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]-------------------------------------------
' This program opens the gripper
' Begin with the gripper closed.

' -----[ I/O Definitions ]-----------------------------------------------

GripperServo   PIN     14                   ' Gripper Servo on P14

' -----[ Constants ]-----------------------------------------------------
' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                    ' Newer standard servo
'#DEFINE NewGripServo = 0                   ' Older STD servo

#IF (NewGripServo = 1) #THEN                ' For newer Standard gripper servo
  StepSize     CON     20                   ' Step size for gripper servo
#ELSE                                       ' For older STD gripper servo
  StepSize     CON     10                   ' Step size for griper servo
#ENDIF

CloseGripper   CON     700                  ' Decrease to close more
OpenGripper    CON     1200                 ' Increase to open more

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

pulsewidth     VAR     Word

' -----[ Main Code ]-----------------------------------------------------
```

```
FOR pulsewidth = CloseGripper TO OpenGripper STEP StepSize
  PULSOUT GripperServo, pulsewidth
  PAUSE 20
NEXT
```

If the Gripper does not stay open, increase the value of **`OpenGripper`** from 1200 to 1250, close the Gripper, and run the program again. Continue until you find the value that makes the Gripper stay open.

Record the values below. You will need to modify each program in this text to match your unique values.

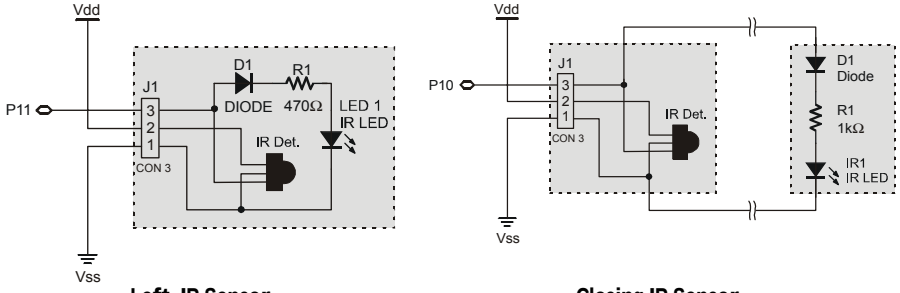
Gripper Position	My PULSOUT Period
Closed	`CloseGripper  CON  _____`
Open	`OpenGripper  CON  _____`

Infrared Sensors

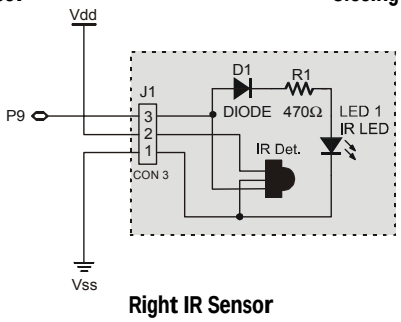
The Boe-Bot Gripper is equipped with three sensors. Two are mounted on the Gripper plates, facing forward. These are referred to as the Left and Right IR sensors. The third sensor is the Closing IR sensor. It consists of two parts, connected by a wire, mounted on the Gripper plates facing each other.

The Left and Right IR Sensors are used to detect objects in front of the Boe-Bot, whereas the Closing IR Sensor detects objects within the jaws of the Gripper. Each sensor consists of an infrared emitter/detector pair. In the schematics below, the IR emitter is labeled "IR LED", and the detector is labeled "IR Det.".

Sensor Schematics



Left IR Sensor **Closing IR Sensor**



Right IR Sensor

In all three sensors, the infrared LED (emitter) and the infrared detector are both connected to the same BASIC Stamp pin. This means that just one BASIC Stamp I/O pin can control both the detector and the emitter. This is possible since BASIC Stamp I/O pins can be configured either as inputs or as outputs, and the assignment can be changed anytime in the program.

The Left and Right IR Sensors

The left and right IR sensors have infrared LEDs and infrared detectors built into them. Infrared LEDs are controlled like ordinary LEDs, but instead of emitting visible light, they emit light in the infrared (IR) region. To detect objects, the infrared LEDs are used somewhat like a car's headlights. They emit infrared, and in some cases, the infrared reflects off objects, and bounces back. This reflected infrared is picked up by the infrared detector. By using the BASIC Stamp to send signals out the IR LED and then read the IR detector, objects can be detected.

The IR detector is only sensitive to infrared at a certain frequency, in the neighborhood of 38.5 kHz, so the IR LED must send out IR at this frequency. The PBASIC command **FREQOUT** is used to generate the signal at the required frequency.[6]

Testing the Left and Right IR Sensors

The key to making each IR pair work is to send 1 ms of unfiltered 38.5 kHz **FREQOUT** harmonic followed immediately by testing the signal sent by the IR detector and saving its output value. The IR detector's normal output state when it sees no IR signal is high. When the IR detector sees the 38500 Hz harmonic sent by the IR LED, its output will drop from high to low. Of course, if the IR does not reflect off an object, the IR detector's output simply stays high. Gripper Program 2.3 shows an example of this method of reading the detectors.

- Enter and run Gripper Program 2.3 Test LR IR Sensors.bs2
- This program makes use of the Debug Terminal, so leave the serial cable connected to the Boe-Bot while Gripper Program 2.3 is running.

```
' -----[ Title ]-----------------------------------------------------------
' Gripper Program 2.3: Test LR IR Sensors
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]---------------------------------------------
' Tests the Gripper's Left and Right IR Sensors
' Point Boe-Bot Gripper at the ceiling for best results
'
' Expected Results:  Both left and right sensors read "1" until
' an object is detected.  Object detected = "0".

' -----[ I/O Definitions ]--------------------------------------------------
```

[6] Refer to "Robotics with the Boe-Bot" for more information on FREQOUT and object detection using infrared.

```
LeftIRPin      PIN     11                    ' Left IR detector/emitter
RightIRPin     PIN     9                     ' Right IR detector/emitter

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

leftIRVal      VAR     Bit                   ' Stores left IR reading
rightIRVal     VAR     Bit                   ' Stores right IR reading

' -----[ Main Code ]-------------------------------------------------------
DO
  FREQOUT LeftIRPin, 1, 38500                ' Send signal to left IR
  leftIRVal = LeftIRPin                      ' Record return

  FREQOUT RightIRPin, 1, 38500               ' Send signal to right IR
  rightIRVal = RightIRPin                    ' Record return

  DEBUG HOME, "Left Sensor  = ", BIN1 leftIRVal, CR  ' Print sensor values
  DEBUG       "Right Sensor = ", BIN1 rightIRVal, CR
  PAUSE 20

LOOP
```

- While Gripper Program 2.3 is running, point the IR detectors so nothing nearby could possibly reflect infrared back at the detectors. The best way to do this is to point the Boe-Bot up at the ceiling. The Debug Terminal should display both left and right values as equal to "1."

- By placing your hand in front of a sensor, it should cause the Debug Terminal display for that detector to change from "1" to "0." Removing your hand should cause the output for that detector to return to a "1" state. This should work for each individual detector, and you also should be able to place your hand in front of both detectors and make both their outputs change from "1" to "0."

- If the sensors passed all these tests, you're ready to move on; otherwise, check your program and wiring for errors.

The Closing IR Sensor

The Closing IR sensor is used to detect when an object is within the Gripper's jaws. This sensor also uses an IR detector and emitter, but in this case, the emitter is mounted on one Gripper plate, and the detector on the other. Unless there is an object in the way, there is a direct "beam" of IR from the emitter to the detector, so the output from the detector will be low (0). When an object is within the Gripper's jaws, the beam is broken, and the detector's output will be high (1).

Logically, this reading is the opposite of the readings from the Left and Right IR sensors. The Left and Right IR sensors return a "low" output for "object detected", yet the Closing sensor returns a "high" output for "object detected." This is simply a result of the difference in the way the sensors are set up in the physical world, and can be dealt with in software. The Closing Sensor's output will be inverted in software, resulting in all three sensors reading "low" for "object detected." To do this, the Inverse operator (~) is used.[7]

Testing the Closing IR Sensor

- Manually open the Gripper plates by turning the servo horn.
- Enter and run Gripper Program 2.4 Test Closing IR Sensor.bs2
- This program makes use of the Debug Terminal, so leave the serial cable connected to the Boe-Bot while Gripper Program 2.4 is running.

```
' -----[ Title ]-----------------------------------------------------------
' Gripper Program 2.4: Test Closing IR Sensor
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]---------------------------------------------

' Tests the Gripper's Closing IR sensor.

' Expected Results:  Sensor reads "1" with nothing in gripper.
'                    Sensor reads "0" with object in gripper.

' -----[ I/O Definitions ]--------------------------------------------------

ClosingIRPin   PIN     10                    ' Closing IR detector/emitter

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

closingIRVal   VAR     Bit                   ' Stores closing IR reading

' -----[ Main Code ]-------------------------------------------------------

DO
  FREQOUT ClosingIRPin, 1, 38500             ' Send signal to closing IR
  closingIRVal = ~ClosingIRPin               ' Record return, invert logic
  DEBUG HOME, "Closing IR Sensor = ", BIN1 closingIRVal, CR  ' Print IR value
  PAUSE 20
LOOP
```

[7] Refer to BASIC Stamp Manual Version 2.0, page 64, for more information on the Inverse operator.

- The Debug Terminal output should display a sensor output of "1".
- Now place your hand or some solid object between the Gripper jaws. This should cause the Debug Terminal display to change from "1" to "0." Removing your hand should cause the output to return to a "1" state. (A solid or dark object will work best - don't use a white piece of paper.)
- If the sensor passed this test, you're ready to move on.

Otherwise, check your program, wiring, and sensor alignment. Check that both parts of the Closing sensor, the detector and emitter, are visible through the slots in the Gripper plates. Slide the sensors in the slots so that the emitter and detector are lined up with one another. Make sure the wires connecting the two parts have not been pinched or detached.

Chapter #4: Application Examples

Opening and Closing the Gripper

This program demonstrates moving the Boe-Bot around to pick up objects, and opening and closing the Gripper. The Boe-Bot will move forward, turn right, then close the Gripper to pick up and object. Then it will back up, turn left, and open the Gripper.

- Open Gripper Program 2.5 Open Close Demo.bs2.
- Edit the program. Change the **`CloseGripper`** and **`OpenGripper`** constants to match your values.
- Edit the code to reflect the servos you are using. Set **`NewGripServo = 1`** if you are using the newer standard servo, or **`NewGripServo = 0`** for the older STD servo. Set **`NewWheelServo = 1`** if you are using the newer continuous rotation wheel servo, or **`NewWheelServo = 0`** for the older pre-modified (PM) servo. You can comment out the incorrect declarations.
- Detach the serial cable.
- Place the Boe-Bot Gripper on the floor where it can move around.
- Press the "Reset" button on the Board of Education to repeat the maneuvers.
- Place an object where the Gripper can pick it up.

```
' -----[ Title ]-----------------------------------------------------------
' Gripper Program 2.5: Open Close Demo
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]---------------------------------------------
' Demonstrates opening and closing the gripper
' This program will move the Boe-Bot forward and right, then
' close the gripper to pick up an object. The Bot-Bot then moves
' backward and left, then opens the gripper to set down the object.
'
' Revision History
' Based on Gripper Program 1.5, updates include:
' - Both new and old gripper servos are supported.
'   Set #DEFINE NewGripServo = 1 for newer servo, = 0 for older.
'    Newer are marked "standard", older are marked "STD"
' - Both new and old wheel servos are supported.
```

```
'   Set #DEFINE NewWheelServo = 1 for newer servo, = 0 for older.
'   Newer are marked "continuous rotation", older marked "PM" (Pre-Modified)
' - PBASIC 2.5

' -----[ I/O Definitions ]-------------------------------------------------

GripServo      PIN     14                    ' Gripper servo on P14
LeftServo      PIN     13                    ' Left wheel servo on P13
RightServo     PIN     12                    ' Right wheel servo on P12

' -----[ Constants ]-------------------------------------------------------

' Choose your type of wheel servo.  1 for newer, 0 for older.
#DEFINE NewWheelServo = 1                    ' Newer continuous rotation servo
'#DEFINE NewWheelServo = 0                   ' Older pre-modified servo

' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                     ' Newer standard servo
'#DEFINE NewGripServo = 0                    ' Older STD servo

#IF (NewWheelServo = 1) #THEN                ' Newer continuous rotation servo
  ClockWise    CON     650                   ' Clockwise wheel rotation
  CtrClkWise   CON     850                   ' Counter clockwise wheel rotation
  TravelPulses CON     20                    ' Num pulses for fwd/back travel
  TurnPulses   CON     12                    ' Num pulses for turning
#ELSE                                        ' For older pre-modified servo
  ClockWise    CON     500                   ' Clockwise wheel rotation
  CtrClkWise   CON     1000                  ' Counter clockwise wheel rotation
  TravelPulses CON     60                    ' Num pulses for fwd/back trav
  TurnPulses   CON     30                    ' Num pulses for turning
#ENDIF

#IF (NewGripServo = 1) #THEN                 ' For newer Standard gripper servo
  StepSize     CON     20                    ' Step size for gripper servo
#ELSE                                        ' For older STD gripper servo
  StepSize     CON     10                    ' Step size for gripper servo
#ENDIF

CloseGripper   CON     700                   ' Replace with values
OpenGripper    CON     1200                  ' for your gripper

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

pulses         VAR     Word                  ' Quantity of pulses to servo
pulsewidth     VAR     Word                  ' Width of pulse sent to servo

' -----[ Main Routine ]----------------------------------------------------

Main:
  GOSUB Open
  PAUSE 500
```

```
  GOSUB Forward
  PAUSE 500
  GOSUB Right
  PAUSE 500

  GOSUB Close
  PAUSE 1000

  GOSUB Backward
  PAUSE 500
  GOSUB Left
  PAUSE 500

  GOSUB Open
  END

' -----[ Subroutines ]------------------------------------------------------

Forward:                                    ' Move Boe-Bot a set distance
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Backward:
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Left:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Right:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Open:                                       ' Open the gripper
  FOR pulsewidth = CloseGripper TO OpenGripper STEP StepSize
```

```
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN

Close:                                    ' Close the gripper
  FOR pulsewidth = OpenGripper TO CloseGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN
```

Autonomous Object Pickup and Drop-off

The following program is a great crowd pleaser! The Boe-Bot will spin around until it detects an object, then go toward the object, find it, and pick it up in the Gripper. Then Boe-Bot will then back up, turn, and set the object down. The process is repeated over and over.

- Enter the program Gripper Program 2.6 Autonomous Object Pickup and Dropoff.bs2.
- Edit the program. Change the **`CloseGripper`** and **`OpenGripper`** constants to match your values.
- Edit the code to reflect the servos you are using. Set **`NewGripServo = 1`** if you are using the newer standard servo, or **`NewGripServo = 0`** for the older STD servo. Set **`NewWheelServo = 1`** if you are using the newer continuous rotation wheel servo, or **`NewWheelServo = 0`** for the older pre-modified (PM) servo. You can comment out the incorrect declarations.
- Detach the serial cable.
- Place the Boe-Bot Gripper on the floor where it can move around.
- Place an object where the Gripper can pick it up.

```
' -----[ Title ]-----------------------------------------------------------
' Gripper Program 2.6: Autonomous Object Pickup and Dropoff
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]---------------------------------------------
' Boe-Bot Gripper spins, finds object, goes and picks it up, then moves
' and sets it down.  This repeats over and over.
'
```

```
' Revision History
' Based on Gripper Program 1.5, updates include:
' - Both new and old gripper servos are supported.
'   Set #DEFINE NewGripServo = 1 for newer servo, = 0 for older.
'    Newer are marked "standard", older are marked "STD"
' - Both new and old wheel servos are supported.
'   Set #DEFINE NewWheelServo = 1 for newer servo, = 0 for older.
'   Newer are marked "continuous rotation", older marked "PM" (Pre-Modified)
' - PBASIC 2.5

' -----[ I/O Definitions ]--------------------------------------------------

GripServo      PIN     14                    ' Gripper servo on P14
LeftServo      PIN     13                    ' Left wheel servo on P13
RightServo     PIN     12                    ' Right wheel servo on P12
LeftIRPin      PIN     11                    ' Left IR detector/emitter
ClosingIRPin   PIN     10                    ' Closing IR detector/emitter
RightIRPin     PIN     9                     ' Right IR detector/emitter

' -----[ Constants ]-------------------------------------------------------

' Choose your type of wheel servo.  1 for newer, 0 for older.
#DEFINE NewWheelServo = 1                    ' Newer continuous rotation servo
'#DEFINE NewWheelServo = 0                   ' Older pre-modified servo

' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                     ' Newer standard servo
'#DEFINE NewGripServo = 0                    ' Older STD servo

#IF (NewWheelServo = 1) #THEN                ' Newer continuous rotation servo
  ClockWise    CON     650                   ' Clockwise wheel rotation
  CtrClkWise   CON     850                   ' Counter clockwise wheel rotation
  CWSlow       CON     730                   ' Clockwise, but slower
  CtrCWSlow    CON     770                   ' Counter clockwise, but slower
  TravelPulses CON     20                    ' Num pulses for fwd/back travel
  TurnPulses   CON     25                    ' Num pulses for turning
#ELSE                                        ' For older pre-modified servo
  ClockWise    CON     500                   ' Clockwise wheel rotation
  CtrClkWise   CON     1000                  ' Counter clockwise wheel rotation
  CWSlow       CON     650                   ' Clockwise, but slower
  CtrCWSlow    CON     860                   ' Counter clockwise, but slower
  TravelPulses CON     40                    ' Num pulses for fwd/back trav
  TurnPulses   CON     50                    ' Num pulses for turning
#ENDIF

#IF (NewGripServo = 1) #THEN                 ' For newer Standard gripper servo
  StepSize     CON     20                    ' Step size for gripper servo
#ELSE                                        ' For older STD gripper servo
  StepSize     CON     10                    ' Step size for gripper servo
#ENDIF

CloseGripper   CON     700                   ' Replace with values
```

```
OpenGripper    CON     1200                  '   for your gripper
Object         CON     0                     ' IR value=0 when object detected

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

leftIRVal      VAR     Bit                   ' Stores left IR reading
rightIRVal     VAR     Bit                   ' Stores right IR reading
closingIRVal   VAR     Bit                   ' Stores closing IR reading

pulses         VAR     Word                  ' Quantity of pulses to servo
pulsewidth     VAR     Word                  ' Width of pulse sent to servo

' -----[ Initialization ]---------------------------------------------------

GOSUB Open                                   ' Open the gripper

' -----[ Main Routine ]-----------------------------------------------------

Main:
DO
  GOSUB Look_For_Object                      ' Spin around to look for object
  GOSUB Center_On_Object                     ' When found,center gripper on it
  GOSUB Move_Object                          ' Pick up object and move it
LOOP

' -----[ Subroutines ]------------------------------------------------------

Look_For_Object:
  DO
    GOSUB Left_Pulse                         ' Spin left 1 pulse
    PAUSE 15                                 ' Slow the spin down
    GOSUB Read_LR_IRs                        ' Check left/right IR sensors
  LOOP UNTIL ((rightIRVal = Object) OR (leftIRVal = Object))
  RETURN

Center_On_Object:
  DO
    GOSUB Forward_Pulse                      ' Forward one pulse
    GOSUB Zero_In                            ' Move left or right
    GOSUB Read_Closing_IR                    ' Check closing IR sensor
  LOOP UNTIL (closingIRVal = Object)         ' Until object is in gripper
  RETURN

Move_Object:
  GOSUB Close                                ' Close gripper to pick up item
  GOSUB Backward                             ' Backup and move to new location
  GOSUB Right
  GOSUB Open                                 ' Open gripper to drop item
  GOSUB Backward                             ' Move away
  GOSUB Left
  RETURN
```

```
Zero_In:                                ' Zero in on the object
  GOSUB Read_LR_IRs                     ' Read left & right IR sensor
  IF (leftIRVal  = Object) THEN
    GOSUB Left_Pulse                    ' If object's on left, move left
  ELSEIF (rightIRVal = Object) THEN     '  to center up on it.
    GOSUB Right_Pulse                   ' If object's on right, move rt
  ENDIF
  RETURN

Forward_Pulse:                          ' Send only 1 pulse to servos
  PULSOUT RightServo, CWSlow
  PULSOUT LeftServo,  CtrCWSlow
  RETURN

Left_Pulse:
  PULSOUT RightServo, CWSlow
  PULSOUT LeftServo,  CWSlow
  RETURN

Right_Pulse:
  PULSOUT RightServo, CtrCWSlow
  PULSOUT LeftServo,  CtrCWSlow
  RETURN

Forward:                                ' Move Boe-Bot a set distance
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Backward:
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Left:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Right:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  CtrClkWise
```

```
    PAUSE 20
  NEXT
  RETURN

Read_LR_IRs:                                ' Read Left & Right IR sensors
  FREQOUT LeftIRPin, 1, 38500               ' Send signal to left IR
  leftIRVal = LeftIRPin                     ' Record return
  FREQOUT RightIRPin, 1, 38500              ' Send signal to right IR
  rightIRVal = RightIRPin                   ' Record return
  RETURN

Read_Closing_IR:                            ' Read Closing IR sensor
  FREQOUT ClosingIRPin, 1, 38500            ' Send signal to closing IR
  closingIRVal = ~ClosingIRPin              ' Record return,due to mechanics,
  RETURN                                    ' invert logic to match LR IRs.

Open:                                       ' Open the gripper
  FOR pulsewidth = CloseGripper TO OpenGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN

Close:                                      ' Close the gripper
  FOR pulsewidth = OpenGripper TO CloseGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN
```

How the Program Works

The program starts with some useful **PIN** declarations in the **I/O Definition** section. The IR emitters (IR LED) are output devices; the IR detectors are input devices. They are both connected to the same BASIC Stamp pin. The **PIN** declaration lets the compiler determine whether a given line of code requires a numeric value for the pin, the output bit for that pin, or the input bit for that pin. Using the **PIN** declaration simplifies IO pin aliasing in sophisticated programs.

A constant definition has been created for **Object,** which makes the program easier to understand.

Bit variables **leftIRVal, rightIValR**, and **closingIRVal** have been declared to hold the readings returned from the sensors.

The heart of object detection is the next two lines:

```
FREQOUT LeftIRPin, 1, 38500
leftIRVal = LeftIRPin
```

The **FREQOUT** command flashes the IR LED at 38.5 kHz for one millisecond[8]. The next line of code immediately reads the IR detector and saves its output value in the bit variable. The bit variable now holds the state of the detector, with a "1" stored meaning an object has been detected, and a "0" stored meaning no object was detected.

These same two lines of code are repeated for each sensor. Note that the Inverse operator ("~" or tilde) is used to invert the reading returned from the closing IR sensor to make the logic match that of the Left and Right IR sensors.

These bit variables can be combined with program logic to make the Boe-Bot Gripper detect objects, seek out objects, follow object, pick up object, avoid objects, or any other idea you think of.[9] This program provides an example of finding objects and picking them up.

The first thing the Boe-Bot Gripper does, in the **Look_For_Object** routine, is spin around looking for an object to pick up. The **Left_Pulse** subroutine causes the Boe-Bot to turn about its axis, while the **Read_LR_IRs** subroutine checks the Left and Right IR sensors for detected objects. The **Read_LR_IRs** subroutine is identical to the code in Gripper Program 1.3 Test LR IR Sensors. The **DO...LOOP...UNTIL** condition keeps the spinning going until an object is spied by either the Left or Right IR sensor.

If an object was detected, the **Center_On_Object** routine is executed. The Boe-Bot will go forward slightly, then read the Left and Right IR sensors, and correct its heading based on the sensor readings. The correction is done in the **Zero_In** subroutine, by moving the Boe-Bot a little left or a little right, depending on the sensor readings.

After this, the Closing IR sensor is checked. If no object is within the Grippers jaws, the program will loops back to **Center_On_Object** due to the **DO...LOOP...UNTIL** construct.

If an object was detected within the Grippers jaws, the code returns to the **GOSUB Move_Object** statement in **Main**. In the **Move_Object** subroutine, the Boe-Bot is instructed to close the Gripper, then back up and move to a new location. At this point the Gripper is opened, and the item is set down. The Boe-Bot then moves away and starts looking for another item. The main **DO...LOOP** sends the program back to where it began, the **Look_For_Object** subroutine.

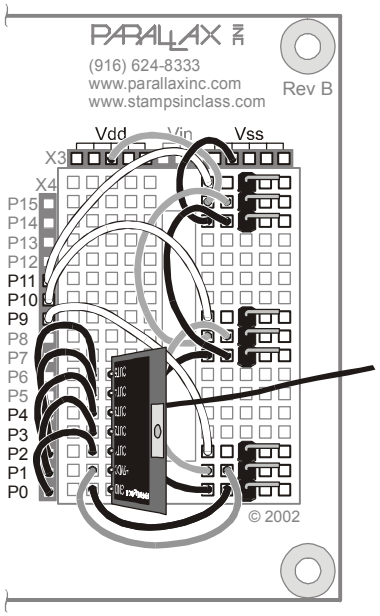
[8] Refer to "Robotics with the Boe-Bot" for more information on FREQOUT and object detection using infrared.
[9] Refer to "Robotics with the Boe-Bot" for some great example programs.

RF Key chain Remote Control

The Parallax Key Chain Transmitter and Receiver modules can be used to implement wireless remote control of the Boe-Bot Gripper. These modules are not included with the Boe-Bot Gripper but can be ordered separately.[10] This program demonstrates how to read the key chain receiver and control the Boe-Bot Gripper. The key chain's middle button is used to open or close the Gripper, and the four directional buttons are used to drive the Boe-Bot.

[10] Key Chain Receiver, Parallax Part Number 28004. Key Chain Transmitter, Parallax Part Number 28005.

Wiring the Key chain Receiver	
From: Key Chain Receiver Pin	**To: BASIC Stamp Pin**
GND	Vss
+5VDC	Vdd
OUT1	P0
OUT2	P1
OUT3	P2
OUT4	P3
OUT5	P4



```
' -----[ Title ]-----------------------------------------------------------
' Gripper Program 2.7: KeyChain Remote Control
'{$STAMP BS2}
'{$PBASIC 2.5}

' -----[ Program Description ]---------------------------------------------
' Control Boe-Bot Gripper with KeyChain Transmitter.
' Up    Arrow = Forward
' Down  Arrow = Backward
' Left  Arrow = Left
' Right Arrow = Right
' Round Btn   = Gripper Open/Close
'
' Revision History
' Based on Gripper Program 1.7, updates include:
' - Both new and old gripper servos are supported.
'   Set #DEFINE NewGripServo = 1 for newer servo, = 0 for older.
'    Newer are marked "standard", older are marked "STD"
' - Both new and old wheel servos are supported.
'   Set #DEFINE NewWheelServo = 1 for newer servo, = 0 for older.
'   Newer are marked "continuous rotation", older marked "PM" (Pre-Modified)
' - PBASIC 2.5

' -----[ Hardware Wiring ]-------------------------------------------------
' Parallax KeyChain Receiver:                              (Part No: 28004)
'  Recvr                  Stamp
' ------                  ------
'   OUT1 -> LEFT ARROW   -> Pin P0
'   OUT2 -> DOWN ARROW   -> Pin P1
'   OUT3 -> RIGHT ARROW  -> Pin P2
'   OUT4 -> UP ARROW     -> Pin P3
'   OUT5 -> ROUND BUTTON -> Pin P4

' -----[ I/O Definitions ]-------------------------------------------------

GripServo      PIN     14                    ' Gripper servo on P14
LeftServo      PIN     13                    ' Left wheel servo on P13
RightServo     PIN     12                    ' Right wheel servo on P12

' -----[ Constants ]------------------------------------------------------

' Choose your type of wheel servo.  1 for newer, 0 for older.
#DEFINE NewWheelServo = 1                    ' Newer continuous rotation servo
'#DEFINE NewWheelServo = 0                   ' Older pre-modified servo

' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                     ' Newer standard servo
'#DEFINE NewGripServo = 0                    ' Older STD servo

#IF (NewWheelServo = 1) #THEN                ' Newer continuous rotation servo
  ClockWise    CON     650                   ' Clockwise wheel rotation
```

```
  CtrClkWise   CON     850                 ' Counter clockwise wheel rotation
  TravelPulses CON     20                  ' Num pulses for fwd/back travel
  TurnPulses   CON     12                  ' Num pulses for turning
#ELSE                                      ' For older pre-modified servo
  ClockWise    CON     500                 ' Clockwise wheel rotation
  CtrClkWise   CON     1000                ' Counter clockwise wheel rotation
  TravelPulses CON     30                  ' Num pulses for fwd/back trav
  TurnPulses   CON     15                  ' Num pulses for turning
#ENDIF

#IF (NewGripServo = 1) #THEN               ' For newer Standard gripper servo
  StepSize     CON     20                  ' Step size for gripper servo
#ELSE                                      ' For older STD gripper servo
  StepSize     CON     10                  ' Step size for gripper servo
#ENDIF

CloseGripper   CON     700                 ' Replace with values
OpenGripper    CON     1200                ' for your gripper
ToOpen         CON     1                   ' These become values telling
ToClose        CON     0                   ' gripper whether to open or close

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

pulses         VAR     Word                ' Quantity of pulses to servo
pulsewidth     VAR     Word                ' Width of pulse sent to servo
dataRx         VAR     Byte                ' Data received from KeyChain Rx
gripCommand    VAR     Bit                 ' Should the gripper open or close
' -----[ Initialization ]--------------------------------------------------

DIRS = %11111111100000                     ' Inputs: P0-P4 KeyChain Receiver
OUTS = %00000000000000

GOSUB Open

' -----[ Main Routine ]----------------------------------------------------

Main:
DO
  GOSUB Read_Rx                            ' Read keychain receiver
  GOSUB Navigate                           ' Move Boe-Bot correct direction
LOOP

' -----[ Subroutines ]-----------------------------------------------------

' Read_Rx subroutine reads data from a Parallax KeyChain Receiver (PN 28004)
' and stores data into variable dataRx
'
' How it works --
'   INL reflects the state of input pins P0 though P7
'   NCD returns ( position of highest set bit ) + 1
' Here's a chart showing how dataRx ends up with it's value after button press
'
```

```
' KeyChain Tx Keypress:    INL Value    NCD INL Value    dataRx value
'                            P7.. P0
' No button pressed          0000000               0               0
' Left button, OUT1 high     0000001               1               1
' Down button, OUT2 high     0000010               2               2
' Right button, OUT3 high    0000100               3               3
' Up button, OUT4 high       0001000               4               4
' Round button, OUT5 high    0010000               5               5

Read_Rx:
  dataRx = NCD INL                          ' Read the pins,store in dataRx
  RETURN

Navigate:                                   ' GOSUB to appropriate motion
  ON dataRx GOSUB Do_Nothing, Left, Backward, Right, Forward, Open_Close
  RETURN

Open_Close:                                 ' Decides whether to
  IF (gripCommand = ToOpen) THEN            ' open or close the gripper
    GOSUB Open
  ELSE
    GOSUB Close
  ENDIF
  gripCommand =  ~gripCommand               ' Next time, do the opposite
  RETURN

Forward:                                    ' Move Boe-Bot a set distance
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Backward:
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Left:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Right:
  FOR pulses = 1 TO TurnPulses
```

```
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Open:                                       ' Open the gripper
  FOR pulsewidth = CloseGripper TO OpenGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN

Close:                                      ' Close the gripper
  FOR pulsewidth = OpenGripper TO CloseGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN

Do_Nothing:
  RETURN
```


How the Program Works

The five outputs of the key chain receiver, labeled OUT1 through OUT5, are wired to BASIC Stamp pins P0 through P4. Whenever a button is pressed, the corresponding pin on the receiver will go high. The layout of the buttons is shown below.

Key chain Button	Receiver Pin	BASIC Stamp Pin
Button1 - Left	OUT1	Pin P0
Button2 - Down	OUT2	Pin P1
Button3 - Right	OUT3	Pin P2
Button4 - Up	OUT4	Pin P3
Button5 - Round	OUT5	Pin P4

In the Read_Rx subroutine, the **INL** keyword is used to read the state of all the buttons in parallel. The **INS** register reflects the state of all 16 I/O pins, while **INL** reflects the state of the lower byte, P0 through P7.

```
dataRx = NCD INL
```

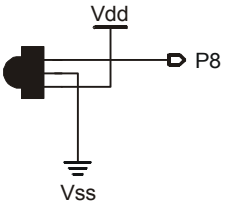
Once the receiver outputs have been read using **INL**, the decoding process is done with the **NCD** operator. **NCD** returns the highest set ("1") bit of a given number, or zero if no bits are set. A non-zero value will be the highest set bit position plus one. Using **NCD**, the value of dataRx will fall between 0 and 5.

Keypress:	INL Value (P7..P0)	NCD INL Value	dataRx value
No button pressed	%0000000	0	0
Left button, OUT1 high	%0000001	1	1
Down button, OUT2 high	%0000010	2	2
Right button, OUT3 high	%0000100	3	3
Up button, OUT4 high	%0001000	4	4
Round button, OUT5 high	%0010000	5	5

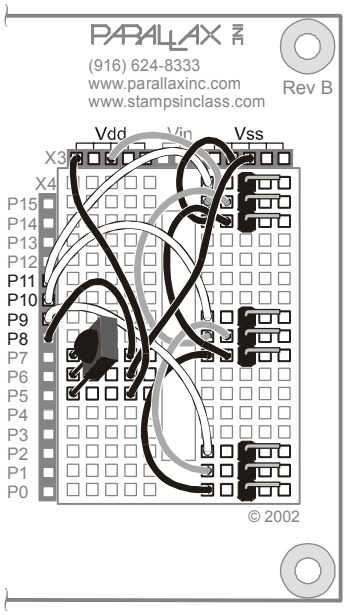
This works out great. A **dataRx** value of one means key chain button 1 was pressed, two means button 2, and so on. In the Navigate subroutine, the **ON...GOSUB** command is used to jump to the appropriate motion routine.

```
ON dataRx GOSUB Do_Nothing, Left, Backward, Right, Forward, Open_Close
```

Infrared Remote Control



Infrared Detector
Parallax Part Number 350-00014



An ordinary television remote control can control the Boe-Bot Gripper. An infrared detector, Parallax part number 350-00014, can be mounted on the Boe-Bot, and any Sony-compatible remote control can be used to send signals to the detector. The signals are received by the detector, interpreted by the BASIC Stamp, and used to control the Boe-Bot and Gripper. The infrared detector is not included with the Boe-Bot Gripper but can be ordered separately, as can a universal remote control.[11]

In the program below, the arrow keys (Channel Up/Down, Volume Up/Down) on the infrared remote control are used to move the Boe-Bot, and the "5" key is used to open and close the Gripper. This program implements 100% manual control, without using the Gripper's sensors at all.

```
' -----[ Title ]-------------------------------------------------------------
' Gripper Program 2.8: Infrared Remote Control

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

[11] Infrared Detector, Parallax Part No. 350-00014. Universal Programmable Remote, Parallax Part No. 020-00001.

```
' -----[ Program Description ]-------------------------------------------
' Control Boe-Bot Gripper with Sony Infrared Remote Control
'
' Remote Control Key:        Robot Motion:
' ----------------------     -------------
' Up    Arrow / Channel +    Forward
' Down  Arrow / Channel -    Backward
' Right Arrow / Volume  +    Right
' Left  Arrow / Volume  -    Left
' "5"                        Gripper Open/Close
'
' Revision History
' Based on Gripper Program 1.8, updates include:
' - Both new and old gripper servos are supported.
'   Set #DEFINE NewGripServo = 1 for newer servo, = 0 for older.
'    Newer are marked "standard", older are marked "STD"
' - Both new and old wheel servos are supported.
'   Set #DEFINE NewWheelServo = 1 for newer servo, = 0 for older.
'   Newer are marked "continuous rotation", older marked "PM" (Pre-Modified)
' - PBASIC 2.5

' -----[ I/O Definitions ]-------------------------------------------------

GripServo      PIN     14                    ' Gripper servo on P14
LeftServo      PIN     13                    ' Left wheel servo on P13
RightServo     PIN     12                    ' Right wheel servo on P12
SonyIR         PIN     8                     ' Infrared Detector on P8

' -----[ Constants ]-------------------------------------------------------

' Choose your type of wheel servo.  1 for newer, 0 for older.
#DEFINE NewWheelServo = 1                    ' Newer continuous rotation servo
'#DEFINE NewWheelServo = 0                   ' Older pre-modified servo

' Choose your type of gripper servo.  1 for newer, 0 for older.
#DEFINE NewGripServo = 1                     ' Newer standard servo
'#DEFINE NewGripServo = 0                    ' Older STD servo

#IF (NewWheelServo = 1) #THEN                ' Newer continuous rotation servo
  ClockWise    CON     650                   ' Clockwise wheel rotation
  CtrClkWise   CON     850                   ' Counter clockwise wheel rotation
  TravelPulses CON     20                    ' Num pulses for fwd/back travel
  TurnPulses   CON     12                    ' Num pulses for turning
#ELSE                                        ' For older pre-modified servo
  ClockWise    CON     500                   ' Clockwise wheel rotation
  CtrClkWise   CON     1000                  ' Counter clockwise wheel rotation
  TravelPulses CON     30                    ' Num pulses for fwd/back trav
  TurnPulses   CON     15                    ' Num pulses for turning
#ENDIF

#IF (NewGripServo = 1) #THEN                 ' For newer Standard gripper servo
  StepSize     CON     20                    ' Step size for gripper servo
```

```
#ELSE                                       ' For older STD gripper servo
  StepSize     CON     10                   ' Step size for gripper servo
#ENDIF

CloseGripper   CON     700                  ' Replace with values
OpenGripper    CON     1200                 ' for your gripper
ToOpen         CON     1                    ' These become values telling
ToClose        CON     0                    ' gripper whether to open or close
NegEdge        CON     0                    ' For 0-1-0 config,IRremote PULSIN

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

pulses         VAR     Word                 ' Quantity of pulses to servo
pulsewidth     VAR     Word                 ' Width of pulse sent to servo
gripCommand    VAR     Bit                  ' Should the gripper open or close
startPulse     VAR     Word                 ' Start pulse duration
pulse0         VAR     Word                 ' Pulse 0 duration
pulse1         VAR     Word                 ' Pulse 1 duration
pulse2         VAR     Word                 ' Pulse 2 duration
dataRemote     VAR     Nib                  ' Stores data from SonyRemoteCtrl

' -----[ Initialization ]--------------------------------------------------

DIRS = %11111011111111                      ' Inputs: P8 Infrared Detector
OUTS = %00000000000000

GOSUB Open                                  ' Start with gripper open

' -----[ Main Routine ]----------------------------------------------------

Main:
DO
  GOSUB Read_Remote                         ' Get data from Sony remote ctrl
  GOSUB Navigate                            ' Move the Boe_Bot
LOOP

' -----[ Subroutines ]-----------------------------------------------------

' The Read_Remote subroutine reads data from a Sony-Compatible infrared
' remote control and stores data into the variable dataRemote
'
' dataRemote Value:     Corresponding KeyPress:
'                 0     Up Arrow / Channel + Key
'                 1     Dn Arrow / Channel - Key
'                 2     Rt Arrow / Volume  + Key
'                 3     Lt Arrow / Volume  - Key
'                 4     "5" Key
'
Read_Remote:
  DO                                        ' Look for start pulse
    PULSIN SonyIR, NegEdge, startPulse      ' Read in first 4 pulses
    PULSIN SonyIR, NegEdge, pulse0
```

```
    PULSIN SonyIR, NegEdge, pulse1
    PULSIN SonyIR, NegEdge, pulse2           ' Start pulse is 2.4ms, and
  LOOP UNTIL (startPulse >= 1000)            ' 1000 PULSIN units is 2000us=2ms

' If BIT9 is 0, then was a short pulse.  If BIT9 is 1, was a long pulse.
' Short pulse stands for zero, long pulse stands for one (Sony Standard)
  dataRemote.BIT0 = pulse0.BIT9        ' Store each pulse as a 0 or 1
  dataRemote.BIT1 = pulse1.BIT9        ' into dataRemote variable
  dataRemote.BIT2 = pulse2.BIT9

  RETURN

Navigate:                                  ' GOSUB to appropriate motion
  ON dataRemote GOSUB Forward, Backward, Right, Left, Open_Close
  RETURN

Open_Close:                                ' Decides whether to
  IF (gripCommand = ToOpen) THEN           ' open or close the gripper
    GOSUB Open
  ELSE
    GOSUB Close
  ENDIF
  gripCommand =  ~gripCommand              ' Next time, do the opposite
  RETURN

Forward:                                   ' Move Boe-Bot a set distance
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Backward:
  FOR pulses = 1 TO TravelPulses
    PULSOUT RightServo, CtrClkWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Left:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, ClockWise
    PULSOUT LeftServo,  ClockWise
    PAUSE 20
  NEXT
  RETURN

Right:
  FOR pulses = 1 TO TurnPulses
    PULSOUT RightServo, CtrClkWise
```

```
    PULSOUT LeftServo,  CtrClkWise
    PAUSE 20
  NEXT
  RETURN

Open:                                       ' Open the gripper
  FOR pulsewidth = CloseGripper TO OpenGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN

Close:                                      ' Close the gripper
  FOR pulsewidth = OpenGripper TO CloseGripper STEP StepSize
    PULSOUT GripServo, pulsewidth
    PAUSE 20
  NEXT
  RETURN
```


How the Program Works

A document explaining how to decode infrared messages using the BASIC Stamp is posted on the Parallax website at the link below. The program listing above is derived from this information. The discussion below assumes you have first read and understood the document.

"Weekend Special - IR LED & 40 kHz Detector.PDF"

To download the document, go to the Parallax web page for the IR Transmitter, at the link below:

http://www.parallax.com/detail.asp?product_id=350-00017

Once on page, click on the link entitled Downloads: Infrared Decoding and Detection appnote

You will be prompted to download a file named IRdecodeDetectAppnote.zip. Save this package and unzip it. The PDF document is inside, along with program listings referred to by the PDF document.

The Read_Remote subroutine counts the durations of a set of pulses and stores the durations in the variables startPulse, pulse0, pulse1, and pulse2. This part of the program will continuously loop looking for a start pulse.

```
Read_Remote:
  DO
    PULSIN sony_IR, NegEdge, startPulse
    PULSIN sony_IR, NegEdge, pulse0
    PULSIN sony_IR, NegEdge, pulse1
    PULSIN sony_IR, NegEdge, pulse2
  LOOP UNTIL (startPulse >= 1000)
```

Once the start pulse has been found, the periods of the three remaining pulses (pulse0, pulse1, pulse2) are evaluated to determine whether they denote binary 1's or 0's. The information contained in an infrared message is carried in the duration of the pulses. A short pulse, 0.6 ms, stands for Binary 0, and a pulse twice that length, 1.2ms, denotes a Binary 1. The unit of time used by the PULSIN command is 2us (BS2), so a short pulse would result in a PULSIN reading of 300, and a long pulse, 600. See chart below.

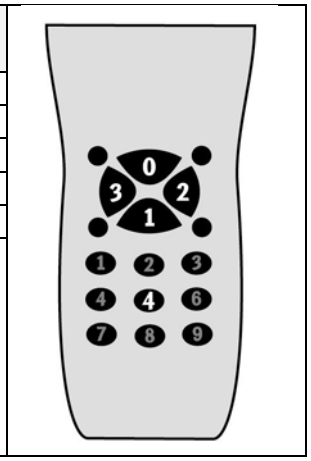
Information	Pulse Duration (in milliseconds)	Pulse Duration (in microseconds)	PULSIN period (in 2us increments)
Start Pulse	2.4ms	2400us	1200
Binary 1	1.2 ms	1200us	600
Binary 0	0.6ms	600us	300

```
dataRemote.BIT0 = pulse0.BIT9
dataRemote.BIT1 = pulse1.BIT9
dataRemote.BIT2 = pulse2.BIT9
```

The duration of the pulses measured won't be exactly 600 or 300, but "around there". Typical "binary 1" pulses may be 608us, 570us, 586us, etc. Typical binary "0" pulses may be 314us, 285us, 301us, etc. The program uses a clever technique in the lines above. It uses the 9th bit of the pulse duration as the "0" or "1" information. If the 9th bit is set, the PULSIN value must have been 512 or greater (2^9 = 512). If the PULSIN value was 300, the 9th bit would not be set. See chart below.

PULSIN period - Decimal	PULSIN period - Binary	9th Bit Set?	9th Bit	Information
600	%000001001011000	Yes	1	Binary 1
300	%000000100101100	No	0	Binary 0

The program places the three bits of information into the dataRemote variable. This variable now represents the key that was pressed on the remote control, as shown in the following table.

dataRemote value	dataRemote value	Corresponding Key Press	
(in binary)	(in decimal)		
%000	0	Up Arrow / Channel + Key	
%001	1	Down Arrow / Channel - Key	
%010	2	Right Arrow / Volume + Key	
%011	3	Left Arrow / Volume - Key	
%010	4	The "5" key	


```
Navigate:
  ON dataRemote GOSUB Forward, Backward, Right, Left, Open_Close
```

The Navigate routine then branches to the appropriate motion routine based on which key was pressed.