QuadCrawler

Kit Assembly, Tuning and Example Programs

VERSION 3.1

Author – Alex Dirks

WARRANTY

CrustCrawler warrants its products against defects in materials and workmanship for a period of 30 days. If you discover a defect, CrustCrawler 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 CrustCrawler. 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 CrustCrawler.

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. CrustCrawler will refund the purchase price of the product, excluding shipping / handling costs. This does not apply if the product has been altered or damaged. Please refer to our web site for the latest warranty information.

COPYRIGHTS AND TRADEMARKS

DISCLAIMER OF LIABILITY

CrustCrawler, 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 CrustCrawler products. CrustCrawler 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 and robotic 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 CrustCrawler, and communicate with other customers. Access information is shown below:

E-mail: Support@CrustCrawler.com
Web: http://www.CrustCrawler.com

INTERNET BASIC STAMP DISCUSSION LIST

Parallax maintains two e-mail discussion lists for people interested in BASIC Stamps (subscribe at http://www.Parallax.com under the technical support button). The BASIC Stamp list server includes engineers, hobbyists, and enthusiasts. The list works like this: lots of people subscribe to the list, and then all questions and answers to the list are distributed to all subscribers. It’s a fun, fast, and free way to discuss BASIC Stamp issues and get answers to technical questions. This list generates about 40 messages per day. The Stamps in Class list is for students and educators who wish to share educational ideas. To subscribe to this list go to http://www.Parallax.com/sic and look for the E-groups list. This list generates about 5 messages per day.

Table of Contents

Preface

The QuadCrawler is an original design from Alex Dirks of CrustCrawler (www.crustcrawler.com).
The QuadCrawler hexapod is an advanced robotic kit consisting of a walking platform.

The applications CrustCrawler provides on our QuadCrawler web pages are project-oriented.
Experience programming the BASIC Stamp is helpful, but if you need more help in this regard you can find plenty of robotic programming resources on the CrustCrawler and Parallax web site. All of the sensors we sell include BASIC Stamp program examples which you could readily adapt to the QuadCrawler.

Putting the hardware together also requires some skill with hand tools. If you are not semi-skilled with common hand tools we recommend you return the kit prior to assembly unless you have some patience.

But, the CrustCrawler team assures you that if you can successfully complete the QuadCrawler you're in for an exciting series of robotic projects that you will find highly rewarding. Our office staff has customized the QuadCrawler by adding, cameras, ultrasonic sensors and infrared detectors.

Chapter #1: Preparing to Assemble the QuadCrawler

REQUIRED TOOLS

The following tools will be required to build your QuadCrawler:

- Phillips screwdriver
- Drill
- 1/8” drill bit
- Small adjustable crescent wrench or socket set
- Wire cutters
- A small amount of white grease or equivalent

QUADCRAWLER FULL KIT INVENTORY

The QuadCrawler Complete Kit contains the following components:

Electronics:

- (1) BASIC Stamp 2 Module
- (1) Board of Education (BOE)
- (8) HiTec HS-322 HD Servos
- (1) Serial Cable
- (1) Parallax CD-ROM
- (1) PSC servo controller
- (1) Seven-segment LED (red)
- (2) 220 Ω resistors (red, red, brown)
- (7) 1 kΩ resistors (brown,black,red)
- (2) 10 kΩ resistors (brown, black, orange)
- (2) pushbuttons
- (1) GP2D12 - 80cm Infrared distance detector
- Package 3” jumper wires

Aluminum Parts:

- (1) upper main body deck
- (1) lower main body deck
- (2) square front and rear support braces
- (2) side braces
- (8) lower, horizontal leg braces
- (8) vertical leg braces
- (4) leg actuators
- (4) leg actuator supports
- (4) servo mounts
- (8) Leg braces - (4) short, (4) Long

Nuts, Bolts, Washers and Screws

- (8) #2 nuts
- (8) #2 lock washers
- (8) #2 washers
- (16) #4 washers
- (16) #4 -1/2” screws
- (26) #4 -5/16” screws
- (14) #4 nuts
- (12) #4 lock nuts
- (4) #4 ¼” nylon spacers
- (8) #4 3/16” nylon spacers
- (16) #6 –3/8” screws
- (16) #6 lock nuts
- (40) #8 flat washers
- (4) #8 -1” screws
- (20) #8 lock nuts
- (16) #8- 1.25” screws
- (8) #8 -½” nylon spacers
- (12) #8 -¼” nylon spacers
- (24) #8 flat nylon spacers
- (8) #8- 7/16” nylon spacers
- (4) ¼” SAE flat washers

Miscellaneous

- QuadCrawler Manual
- (4)- ¾” long - 2/56 threaded rods
- (4) rubber feet
- (8) dog bones
- (8) ball links
- (10) cable ties

Source Code from the CrustCrawler Web Site (www.crustcrawler.com)

QuadCrawler Source Code

The CrustCrawler web site QuadCrawler page contains sample BS2-IC source code, this installation guide and additional pictures of the QuadCrawler to aid you during the construction process. Additionally, we frequently post additional applications for our products on the web site.

The included Parallax CD-ROM includes the *BASIC Stamp Manual* and many valuable resources to assist you with your BASIC Stamp programming efforts.

Projects and Accessories are available from the Parallax Web Site (www.parallax.com) and the CrustCrawler web sites (www.CrustCrawler.com)

QuadCrawler Projects / Accessories

Projects that include updated code, electronics and hardware accessories are always being added to the Crustcrawler and Parallax web sites. Check with our site often for the latest updates.

ADDITIONAL PARTS YOU NEED TO SUPPLY

Like other hobby kits, completing the QuadCrawler kits requires additional parts that you will need to supply. This hardware is not included in the Parallax kits because it would only drive the cost higher as most of these items are readily available at hobby stores or www.towerhobbies.com (purchasing them from Parallax, if available, would probably have a higher price).

- 7.2 V NiMH or NiCd six-cell rechargeable battery for servo power. This is a standard 1800 mAH to 3000 mAH battery pack, commonly used in R/C cars. A good source for this product is Tower Hobbies (www.towerhobbies.com). Expect to pay between $30 and $60. Or, you can go to Radio Shack who also sells good quality 7.2V NiMH batteries and chargers.
- AC/DC Digital Peak Charger for the 7.2 V NiCd/NiMH battery. One acceptable product is the Piranha Digital Charger from Tower Hobbies (their stock code #LXCLD5). Probably around $50 or less. Radio Shack also sells chargers for their R/C cars.
- Zip-ties of the smaller sizes are very useful for securing wires in a tidy fashion. Approximate cost is probably a few dollars. Available from any hardware store. (12) Zip ties have been included in your kit.
- Robot sensors The QuadCrawler is a platform and the opportunities for sensor integration are endless. Selecting the appropriate sensor is left up to you, our customer. Parallax and CrustCrawler (www.crustcrawler.com) have many acceptable add-ons for the CrustCrawler. There is no cost limit in this regard.

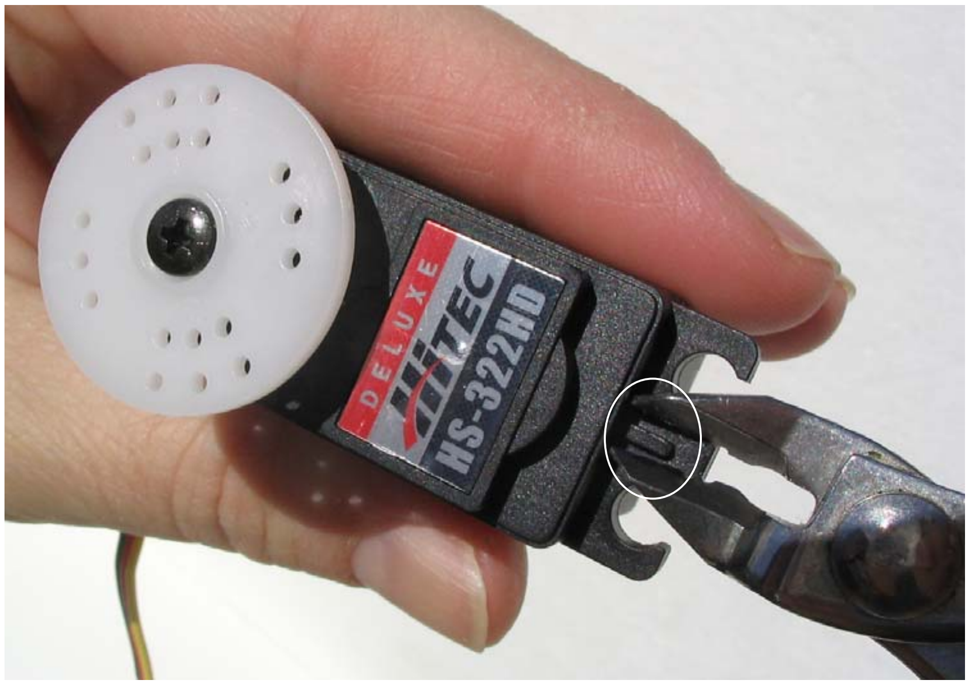
Chapter #2: Pre-Assembly Tips

PAY ATTENTION TO DETAILS

- Work in a well lit, clean environment with lots of workspace
- Obtain a small stack of books that are approximately the length and width of the QuadCrawler's lower deck and is approximately 3.5" to 4" tall when stacked. The added height will be required when attaching the lower and upper decks together.
- Organize your nuts, bolts and screws so that you have each specific size of lock-nuts, screws, washers in the same group and are easily within reach.
- Take your time! The QuadCrawler kit is a precision made product that contains a lot of parts and requires all parts to be assembled in the *exact* order as described in this installation manual.
- The average time to construct a kit is between 5 to 8 hrs.
- Additional pictures of the construction process may be downloaded from the Crusrcrawlers web site's QuadCrawler pages. During the construction process, please refer to these pictures and the ones in this guide frequently as a reference. Refer to and study the pictures and close-up diagrams carefully **before** starting the construction of any part of your kit.
- Always note the **orientation** and **direction** of screws and aluminum parts and which **side** of your QuadCrawler you are constructing! It absolutely makes a difference!

PREPARING THE SERVOS

- Remove the aluminum body parts from their protective bags and lay them loosely in their respective groups on your work surface.
- Gather (4) of the servos and remove the servo horn and screw, setting them aside in a safe place.
- Remove the riser tabs from both sides of (4) of the servos as shown below. The removal of this plastic riser will allow the servo to sit flush against the servo holder. These servos will be installed in the next few steps of the construction process.



Chapter #3: QuadCrawler Assembly

MOUNTING THE SERVOS

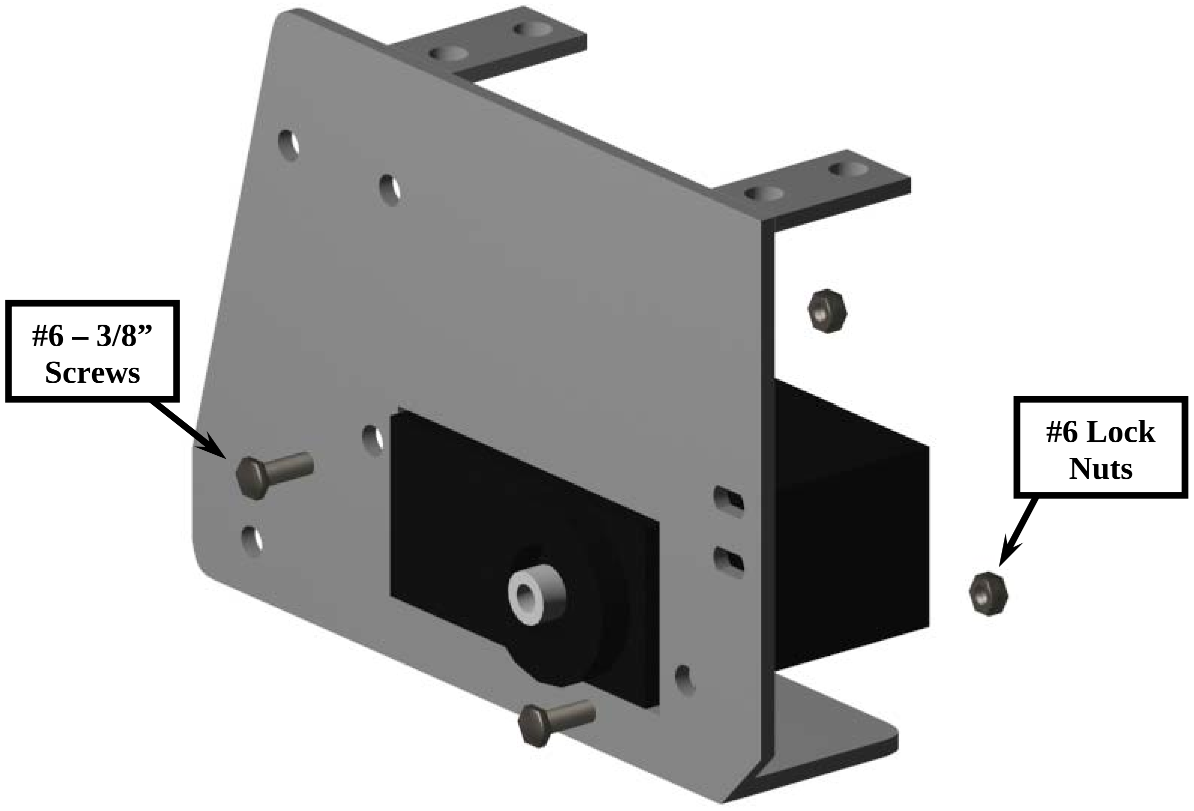


Figure 2: Mounting Servos

1. Install the servo into the rectangular portion of the servo mount. The servo gear head should always be orientated towards the straight end of the servo mount as shown in figures 2 and 3.

2. Using (2) #6 - 3/8” screws and lock nuts, install the servo to the aluminum servo assembly. Complete this step for all (6) legs of your QuadCrawler.

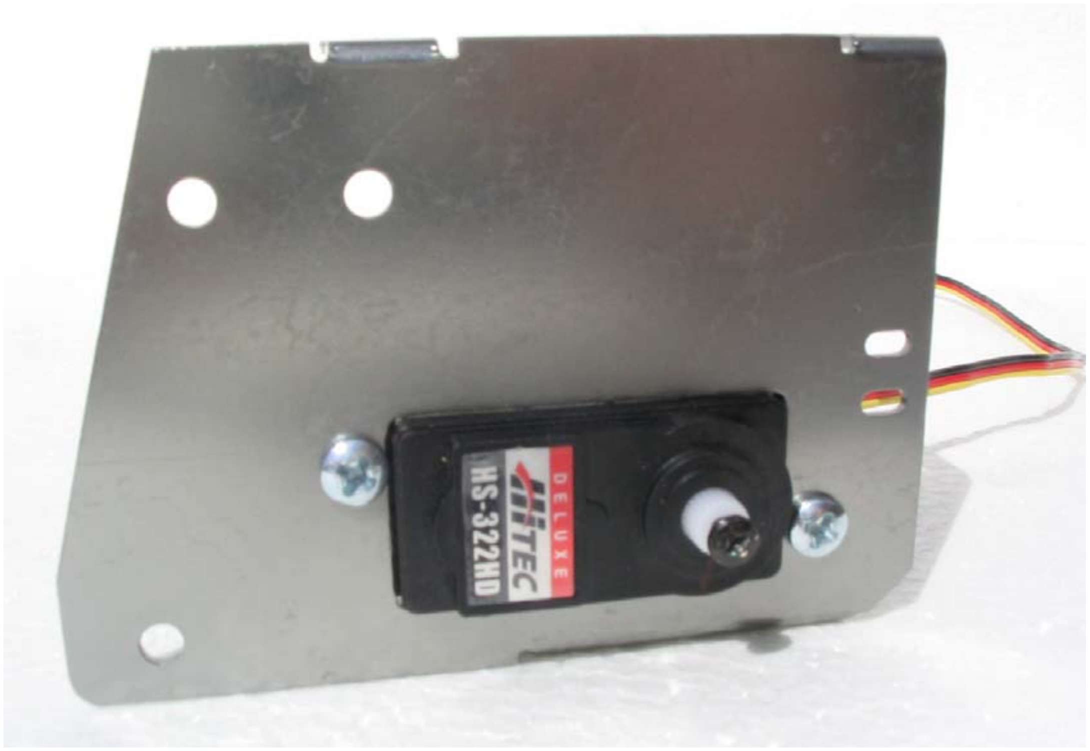


Figure 3: Servo Orientation

3. Using (2) #6 - 3/8" screws and lock nuts, install the top servo to the aluminum servo assembly as shown in figure 4. Ensure that the servo is mounted on top of the servo tabs as shown in figure 4,5 and figure 6. Complete this step for all (4) legs of your QuadCrawler.

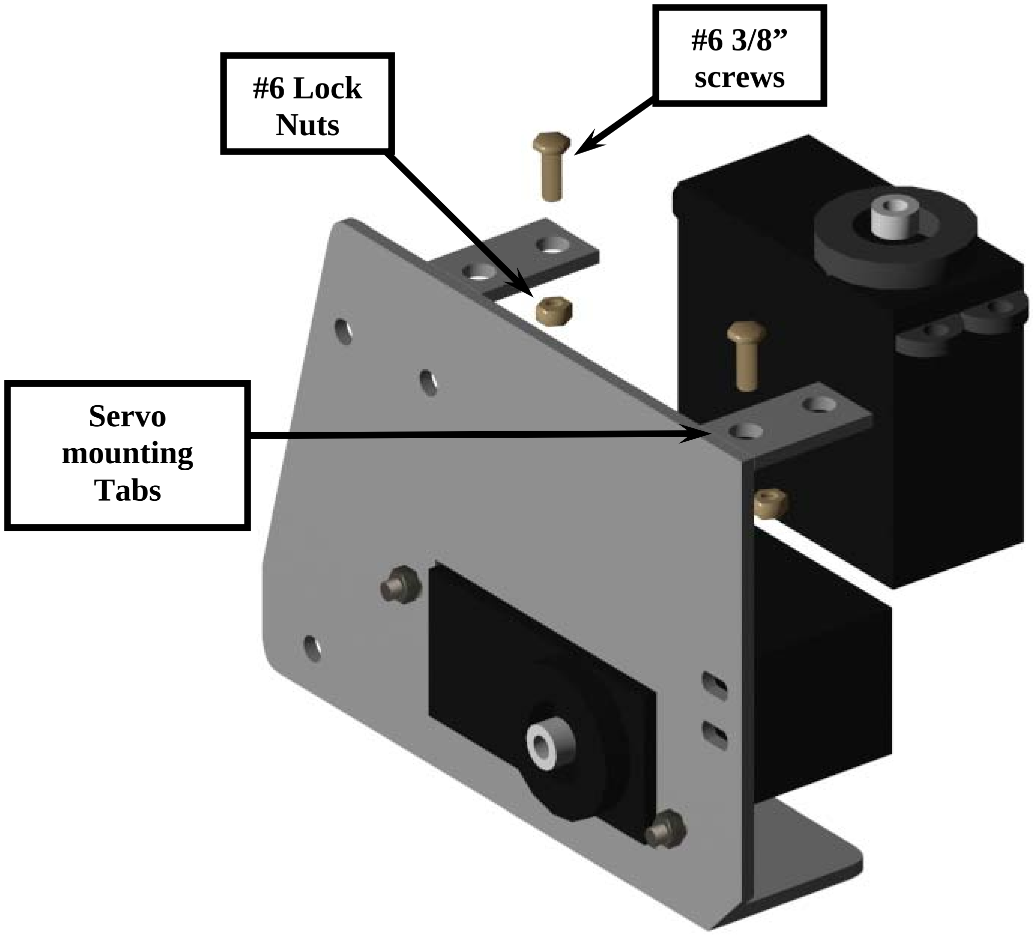


Figure 4: Top Servo Installation

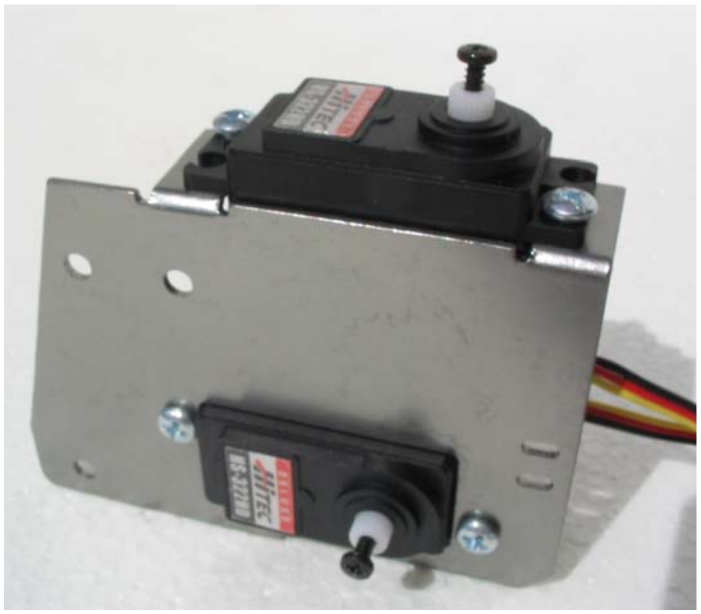


Figure 5: Completed Servo Installation – Front view

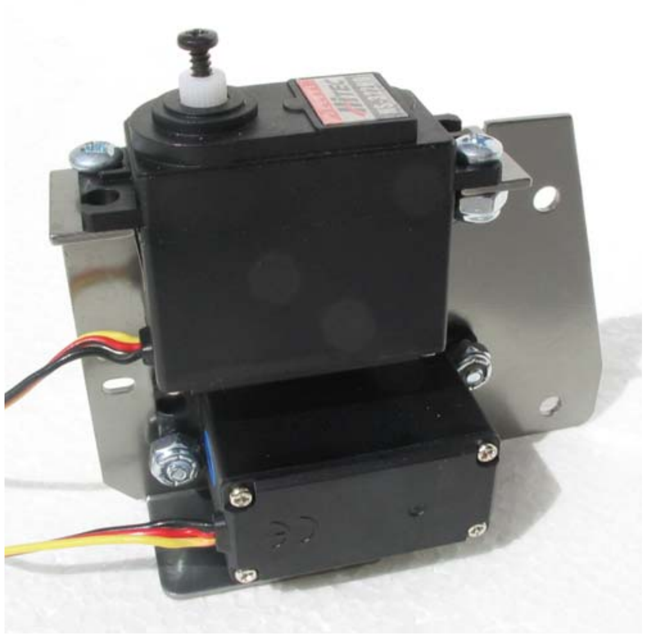


Figure 6: Completed Servo Installation – Rear view

4. Install the Du-Bro 2-56 threaded ball link to all (4) of the main leg actuators as shown in Figure 7 below. Ensure that the ball link is centered in the slot.

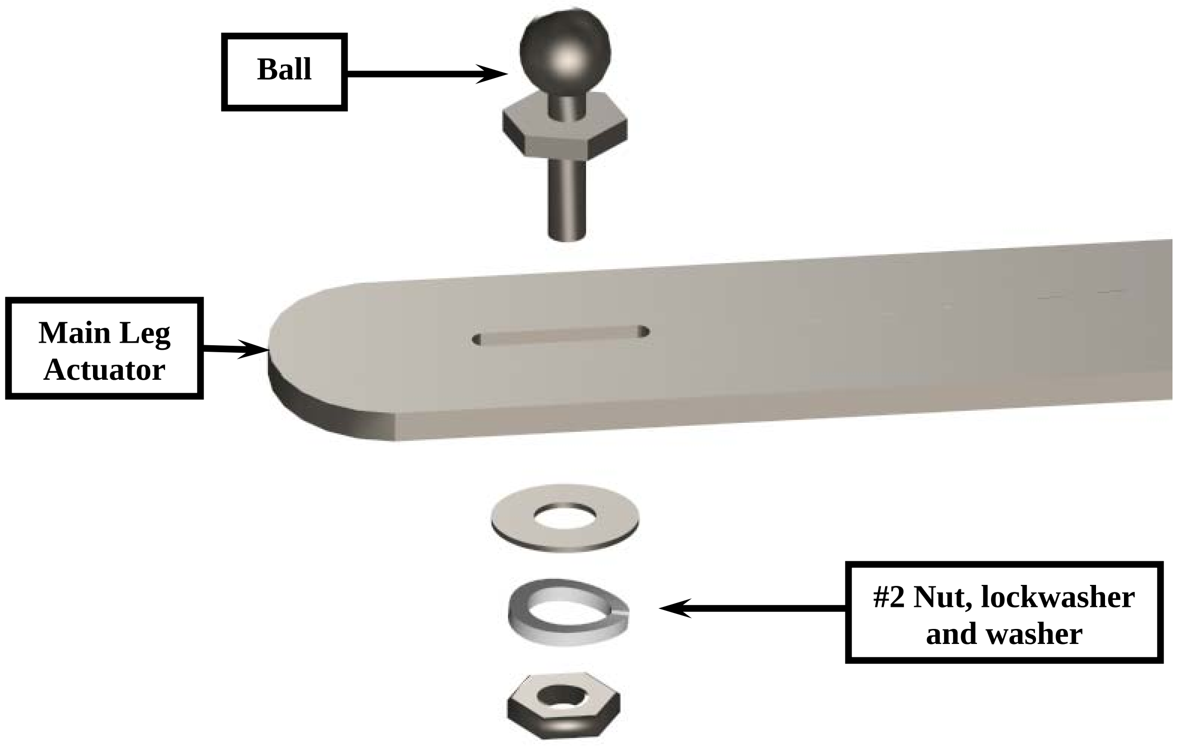


Figure 7: Leg Actuator Assembly

5. Steps 6 - 14 should be performed for each leg in your kit. Once one leg has been assembled, use the finished leg as a quick reference for the assembly of the rest of the legs.

6. Using (1) #8, 1.25” screw, install the leg actuator to the leg actuator support through the #8 hole near the top of the servo mount as shown in Figure 8. Use the same hole for the right and left leg assemblies.

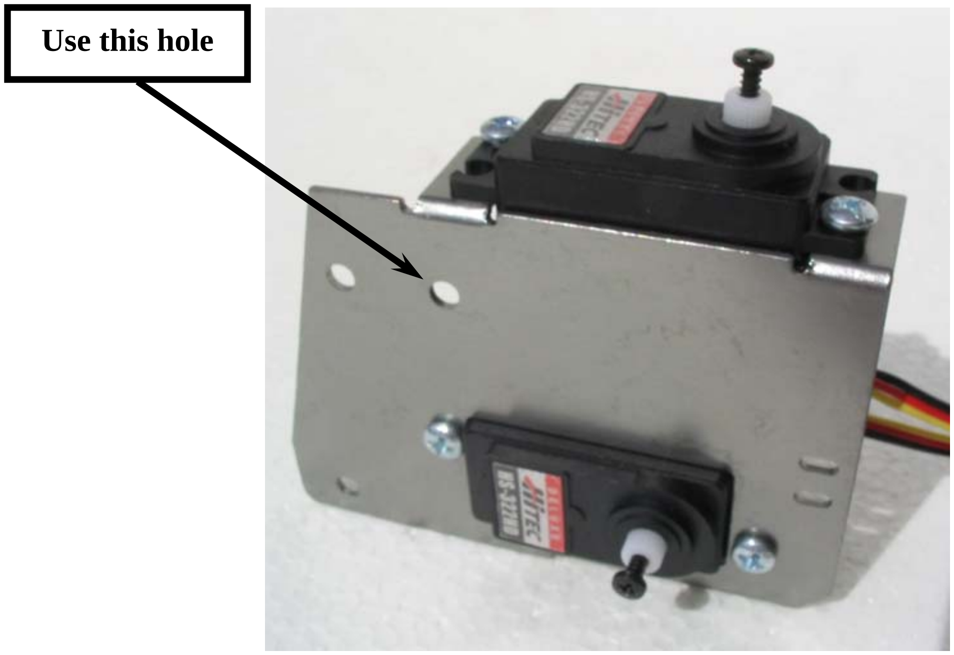


Figure 8: Leg Actuator Assembly

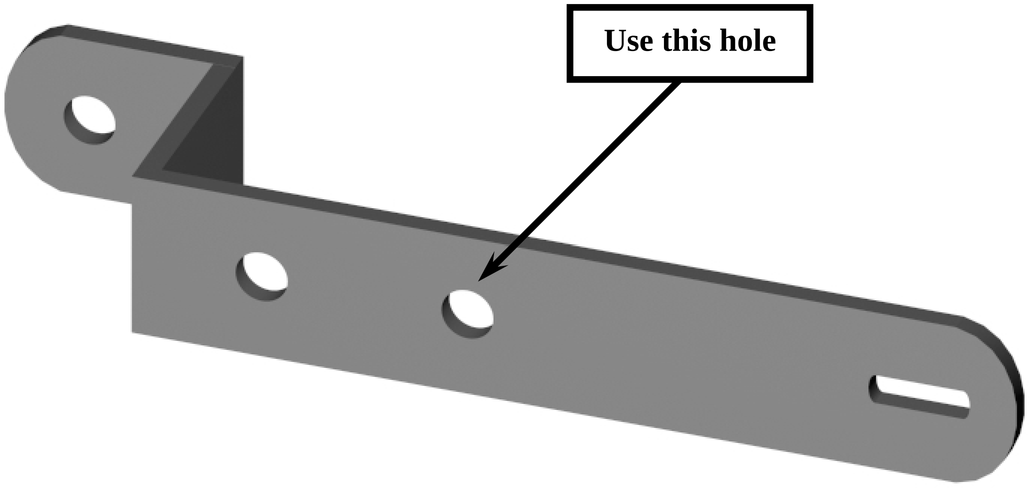


Figure 9: Leg Actuator

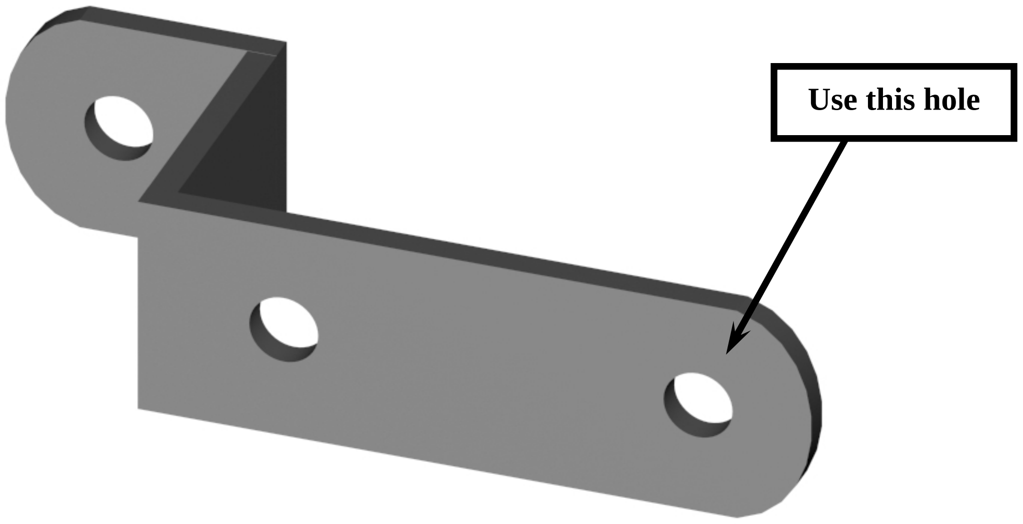


Figure 10: Leg Actuator Support

7. Use the close-up diagram in Figure 11 for the complete assembly sequence.

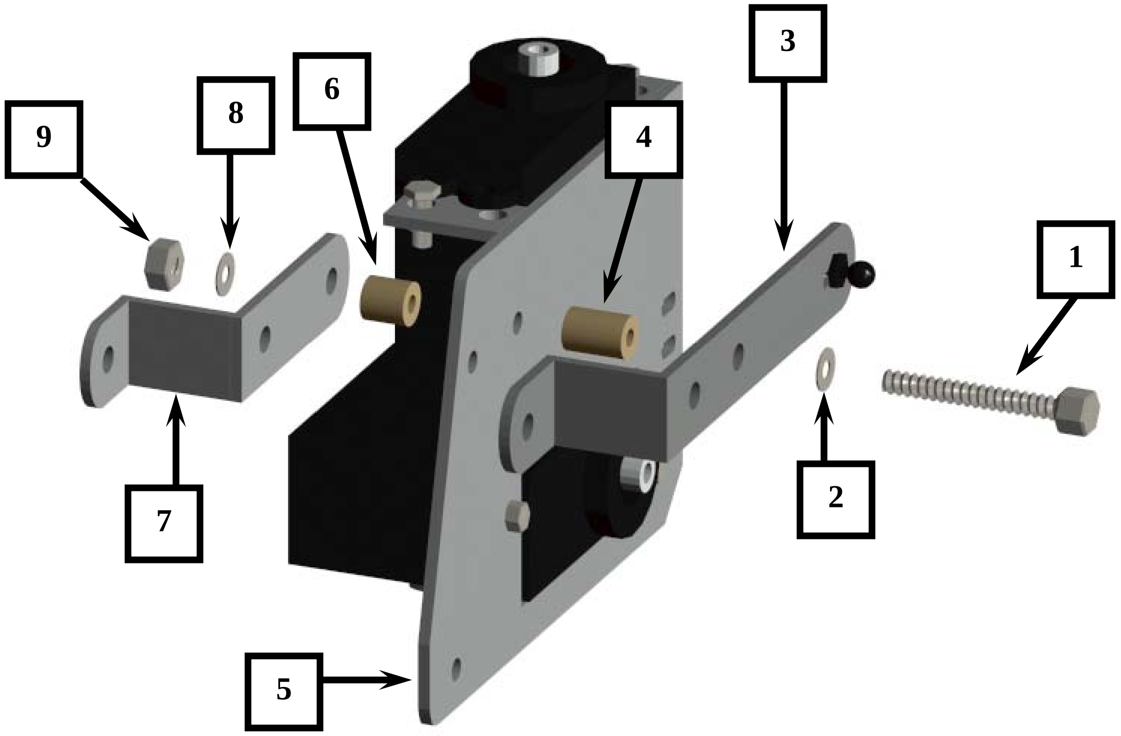


Figure 11: Leg Actuator Assembly

1 - #8 1.25" Screw	6- #8 ¼" nylon spacer
2 – #8 Stainless steel washer	7 – Leg Actuator Support
3 – Leg Actuator	8 – #8 Stainless steel washer
4 – #8 7/16" Nylon spacer	9 – #8 Lock Nut
5 – Servo Mount	

8. At this point in the process, the leg assembly should look like Figure 12 and figure 13. Note the location and orientation of the holes and ball socket slot. Tighten the screw and lock nut just enough so that the 2 pieces can be easily moved by hand but rigid enough to stand on their own. These screws will be loosened later in the construction process to ensure smooth, friction free leg movement. Do not over-tighten the screws as it makes the rest of the leg assembly difficult to complete.

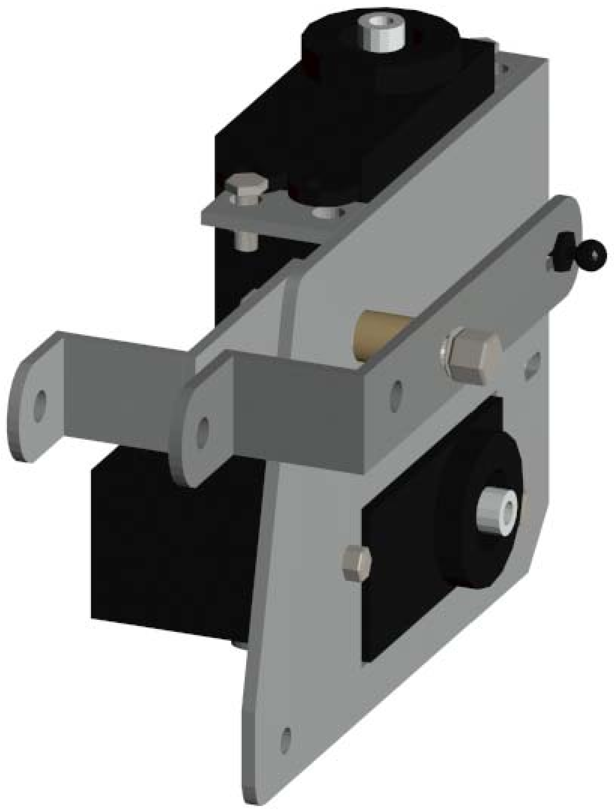


Figure 12: Leg Actuator Assembly

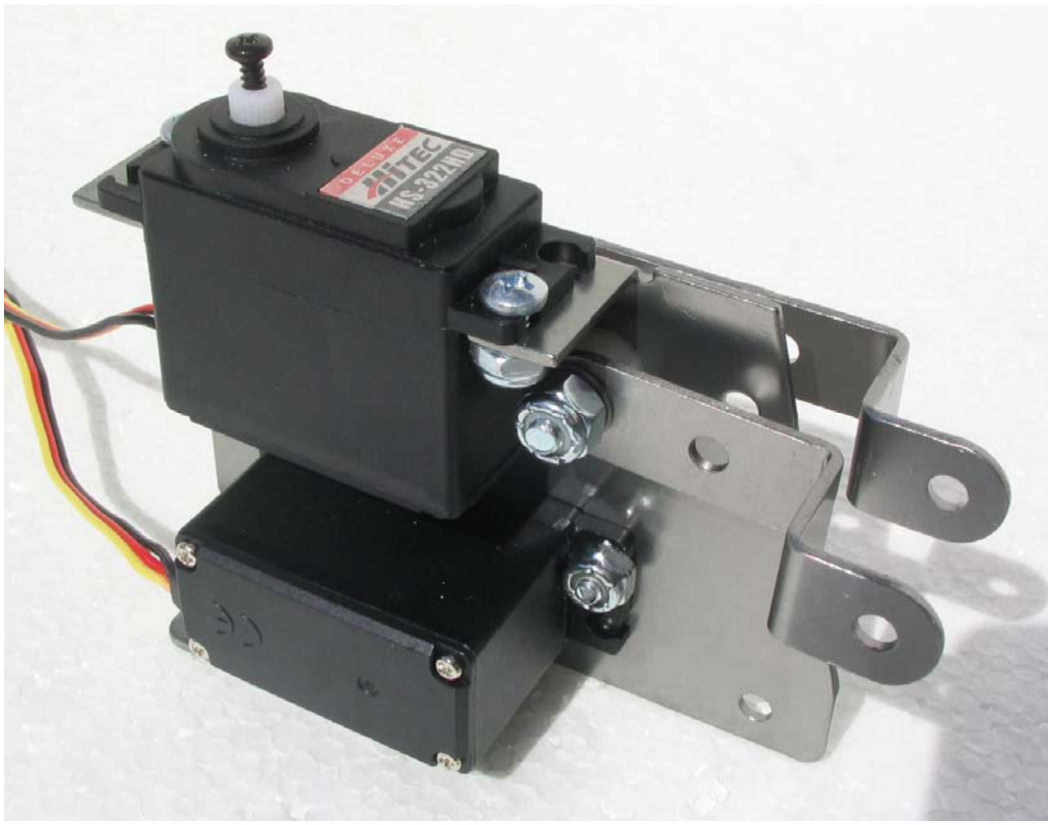


Figure 13 – The Completed Leg Actuator Assembly

9. Attach (2) lower horizontal leg braces with the same hardware and in the same order as the previous steps to the lower hole on the servo mount as shown in Figure 14. Use the close-up diagram on the next page as a reference. Ensure that the shorter end of the lower horizontal leg brace is installed to the servo mount as shown in figure 14 and figure 15.

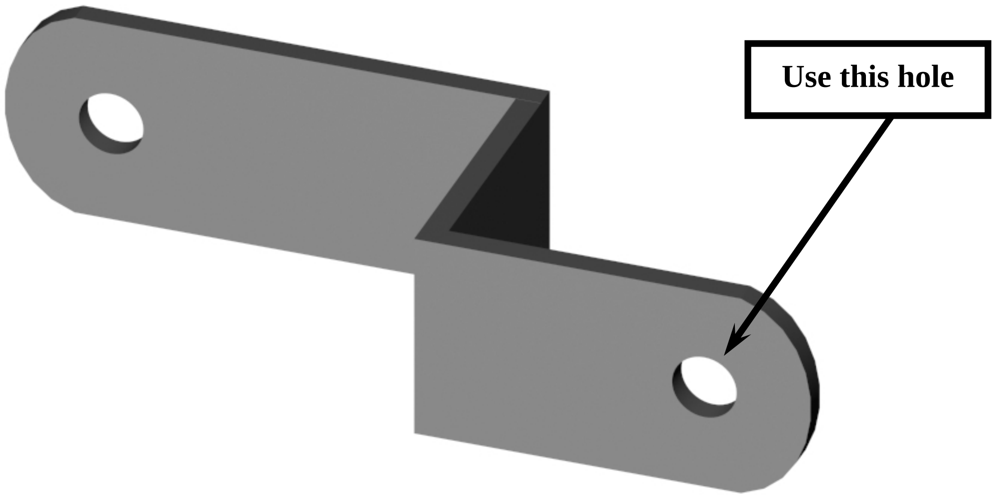


Figure 14: Lower Horizontal Leg Brace

9. Use the close-up diagram in Figure 15 for the complete assembly sequence.

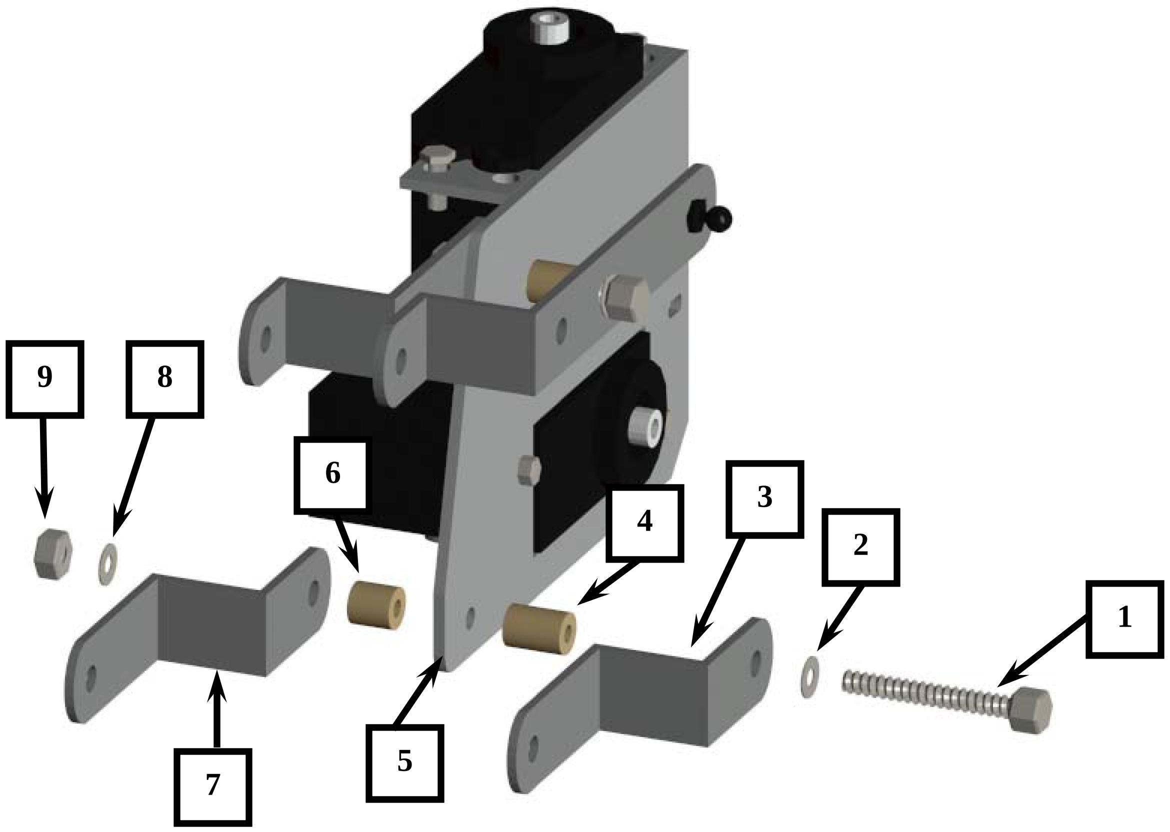


Figure 15: Lower Horizontal Leg Assembly

1 - #8 Screw 1.25”	6- #8 ¼” Nylon spacer
2 – #8 Stainless steel washer	7 – Lower horizontal leg brace
3 – Lower horizontal leg brace	8 – #8 Stainless steel washer
4 – #8 7/16” Nylon spacer	9 – #8 Lock nut
5 –Servo Mount	

When completed, the leg assembly should look like Figure 16 and Figure 17.

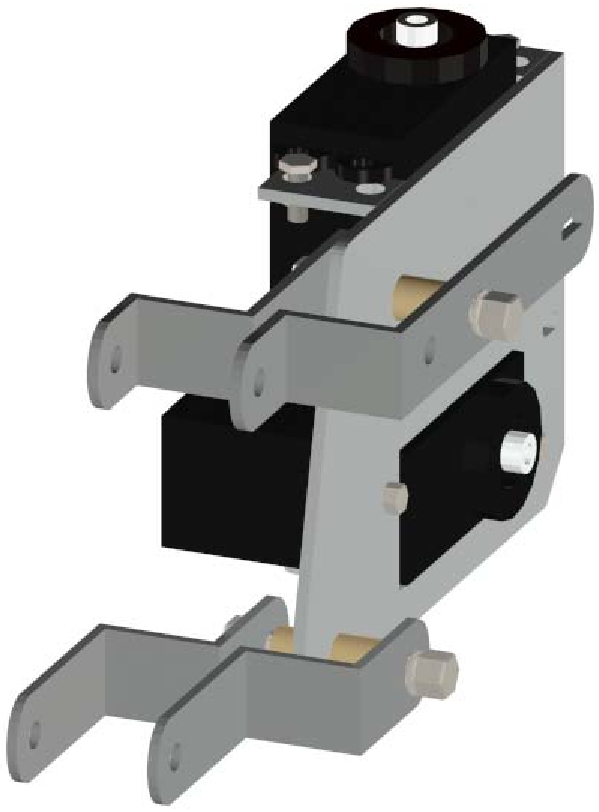


Figure 16

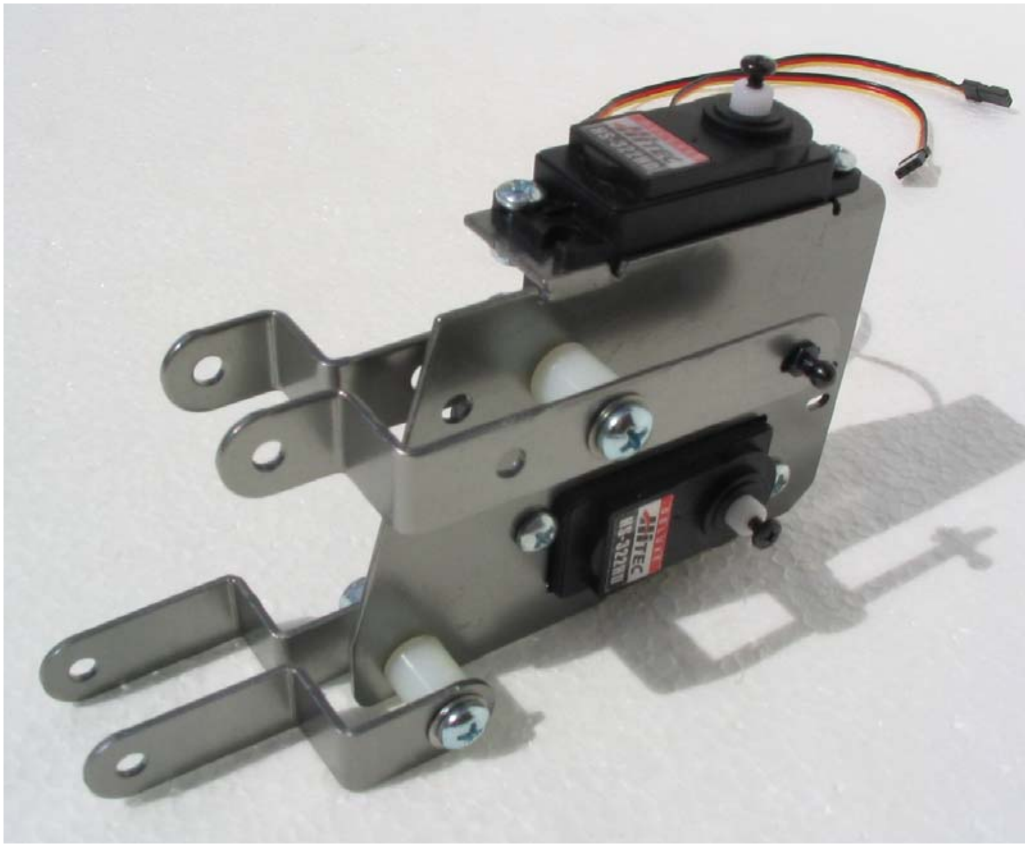


Figure 17

10. Next, install the 2 vertical leg braces (Figure 18) to the leg assembly. Use Figure 19 to assemble the vertical leg braces. Note that the top vertical brace assembly is identical to the lower vertical brace assembly.

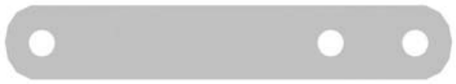


Figure 18: Vertical Leg Brace

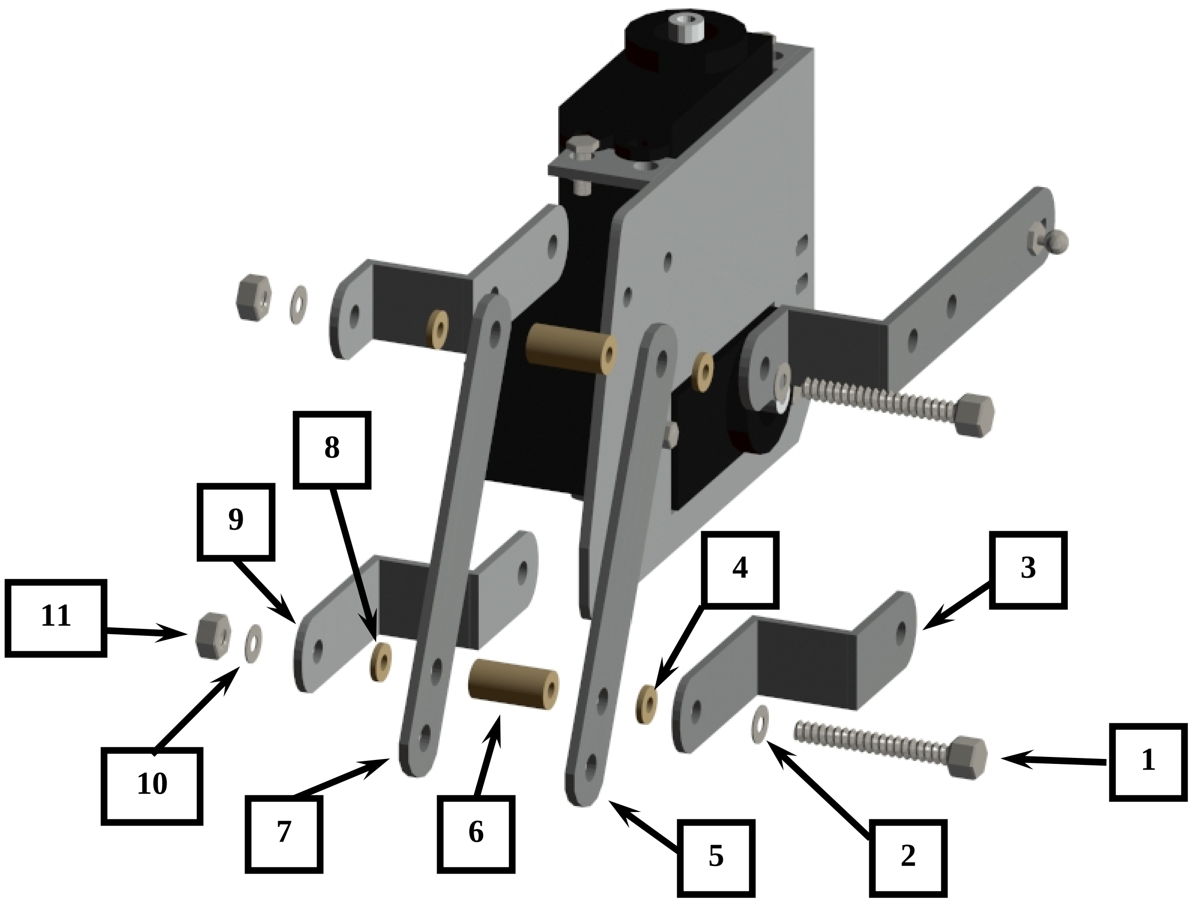


Figure 19: Vertical Leg Assembly

1 - #8 1 1/4” screw	7 - Vertical leg brace
2 - #8 Stainless flat washer	8 - #8 Flat nylon spacer
3 - Lower horizontal leg brace	9 - Lower horizontal leg brace
4 - #8 Flat nylon spacer	10 - #8 Stainless flat washer
5 - Vertical leg brace	11 - #8 Lock nut
6 - #8 ½” nylon spacer	

Tip Assemble the top of the vertical leg brace assembly first and then assemble the lower vertical leg assembly. As shown in the close-up view below, the assembly sequence and parts are identical. Do not completely tighten the upper or the lower #8 screws. Leave the screws just loose enough so that the leg assembly can be moved up and down easily by hand.

When completed, the leg assembly should look like Figure 20.

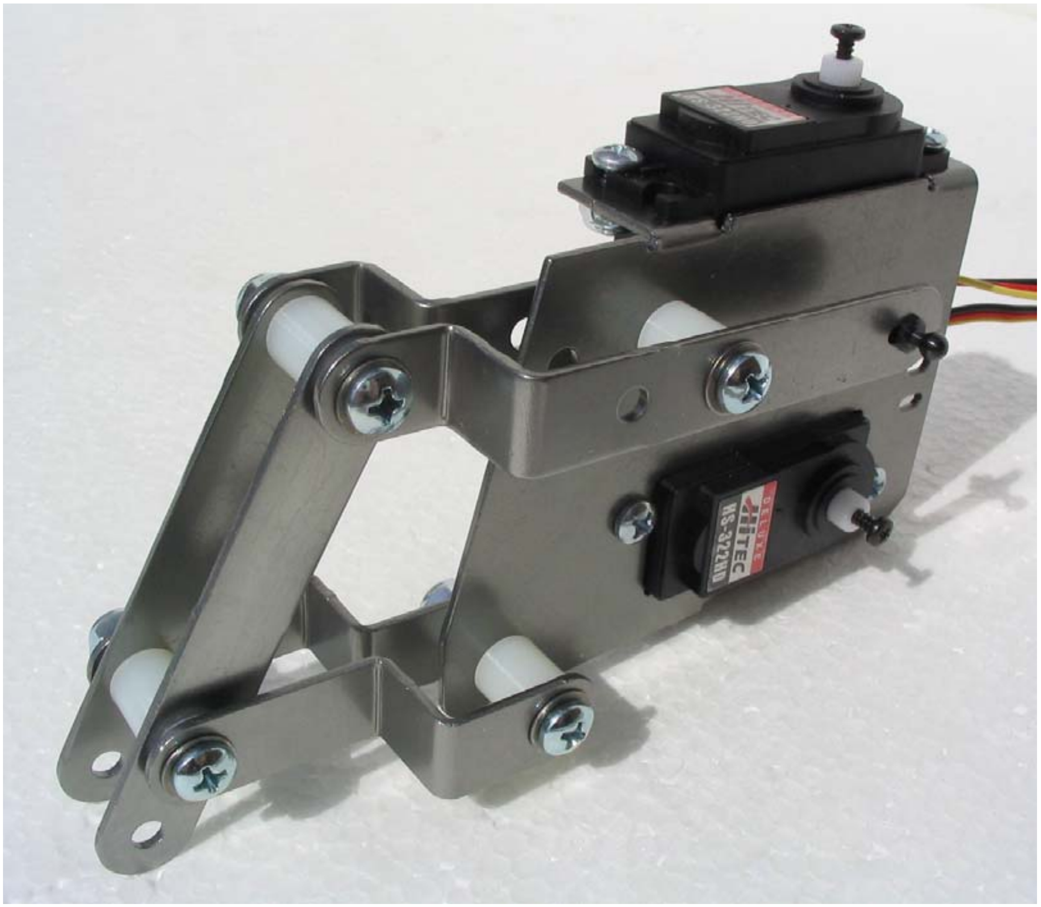


Figure 20: Completed Leg Assembly

ASSEMBLING THE LOWER LEG AND ATTACHING IT TO THE UPPER LEG ASSEMBLY

11. Use figure 21 to assemble the lower leg assembly. Please note the orientation of the longer leg brace and the shorter brace. The shorter leg brace is always installed on top of the longer leg brace. To make construction easier, follow the numbered steps as outlined in figure 21.

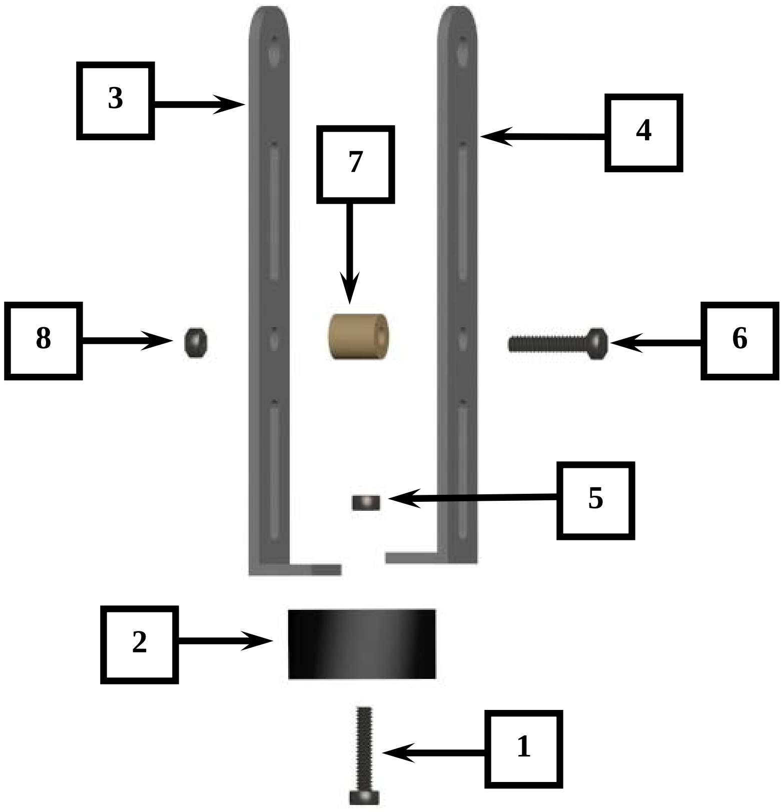


Figure 21 – Lower Leg assembly

1- #4 1/2” screw	5- #4 nut
2- Rubber Foot	6- #4 ½” screw
3- Long Leg Brace	7- #4 ¼” Nylon spacer
4- Short Leg Brace	8 - #4 Lock Nut

When completed, the leg assembly should look like figure 22.

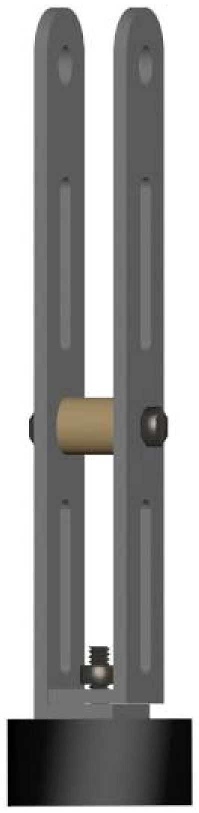


Figure 22 – Completed Lower Leg Assembly

12. Attach the lower leg assembly to the upper leg assembly as shown in figure 23.

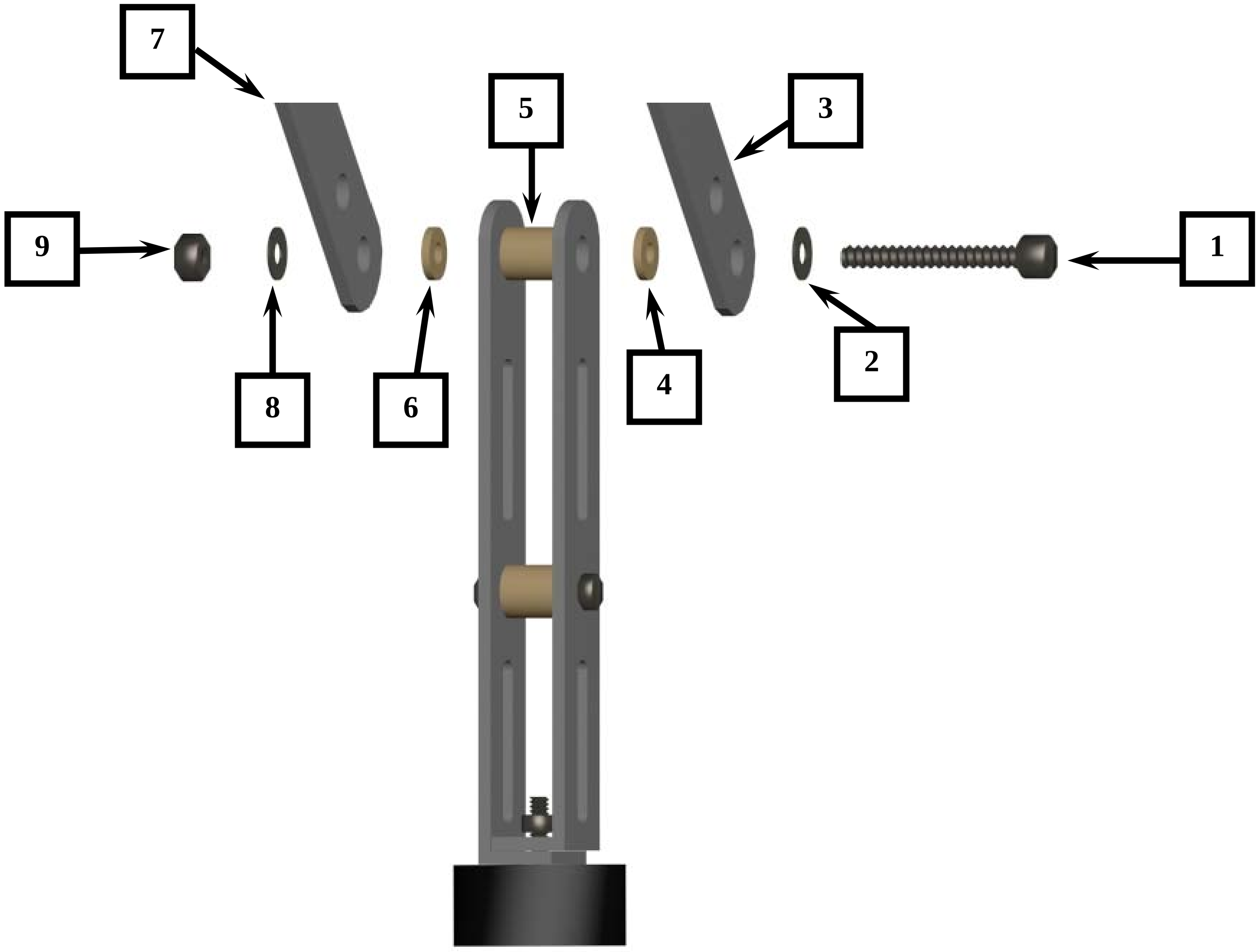


Figure 23

1. #8 – 1" Screw	3. Vertical leg brace	5. #8 – ¼" nylon spacer	7. Vertical leg brace	9. #8 Lock nut
2. #8 Stainless washer	4. #8 Flat nylon spacer	6. #8 Flat nylon spacer	8. #8 stainless washer	

When completed, the leg assembly should look like Figure 24.

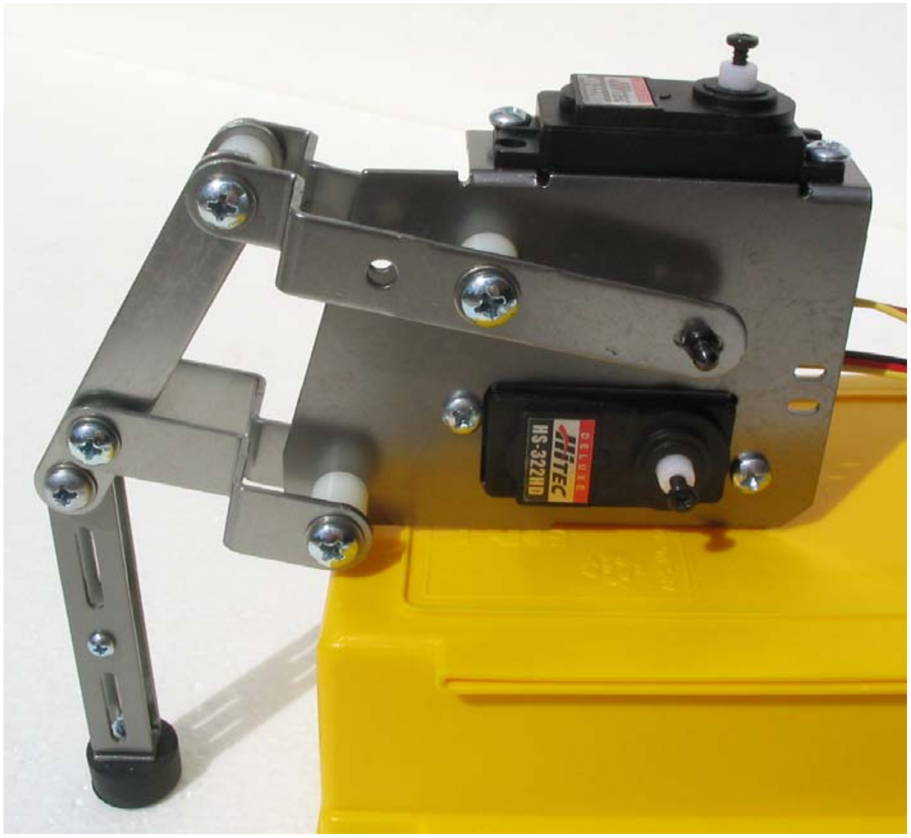


Figure 24 – Completed Lower Leg Assembly

13. Using the supplied black tie wraps, secure the 2 servo wires to the servo mount as shown in Figure 25 and Figure 26. Be sure not to stretch or extend the servo wire when securing them to the servo mount.

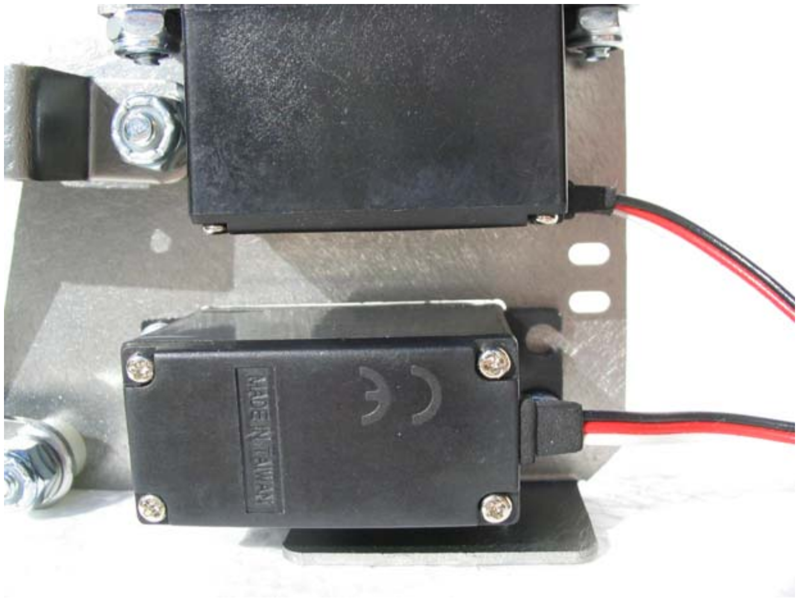


Figure 25 – Securing the servo wires to the servo mount

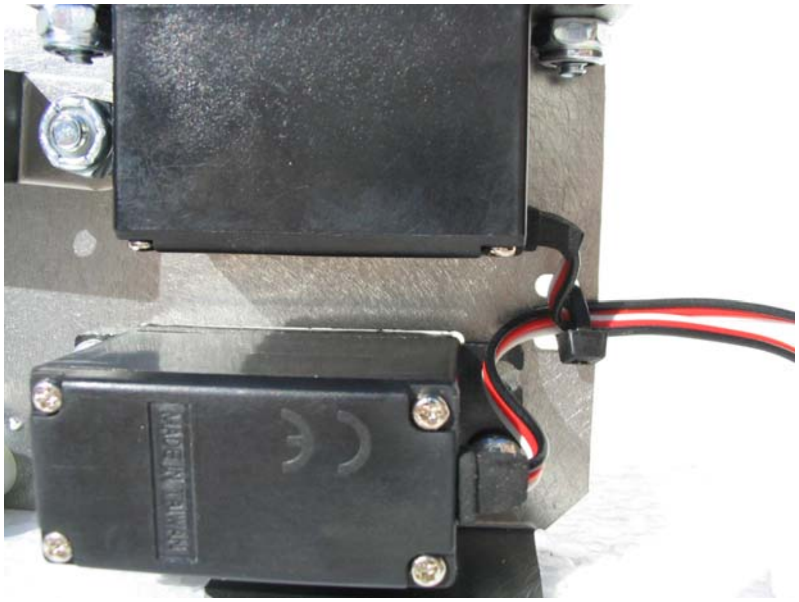


Figure 26 – The completed assembly

PREPARING THE QUADCRAWLER'S UPPER DECK

The upper deck can be identified by the 3 holes at the ends of each of the leg decks (1 large hole and 2 smaller holes). Locate the upper deck and have it ready.

1. Drill out the second hole from the end of the longest (2) arms of the servo with a 1/8" drill bit (Figure 27). To make this process easier, temporarily place the servo arm on an available servo spindle to secure the arm while it is being drilled. Also, the round servo arms can be used instead of the star shaped servo arms for this step. Place the round servo arm onto the leg deck to identify the proper holes to drill.

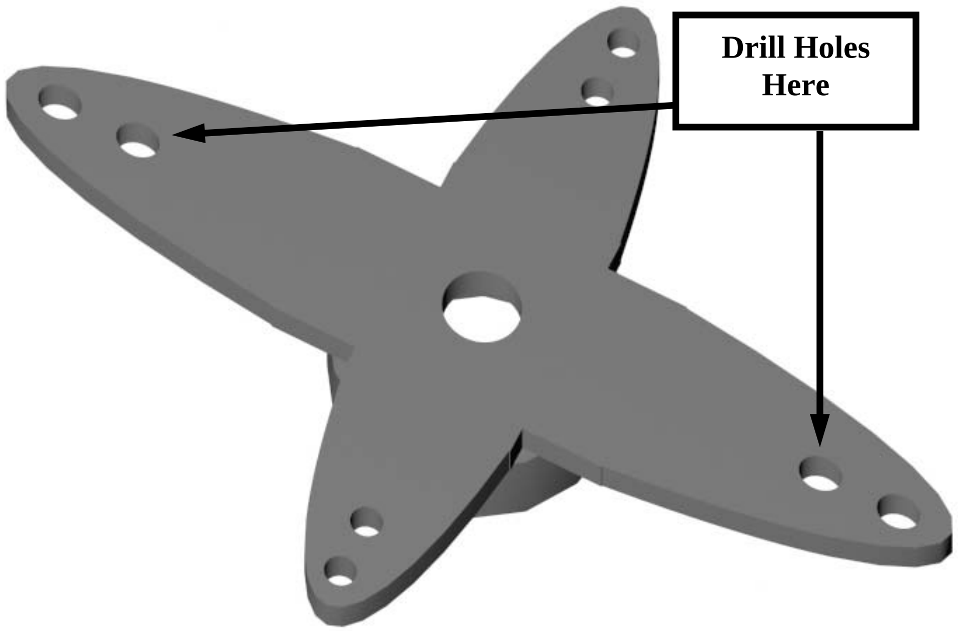


Figure 27

2. Using (2) 5/16” #4 screws, washers and lock nuts, install the servo arm to the upper deck ensuring that the flat side is mounted to the upper deck as shown in Figure 28.

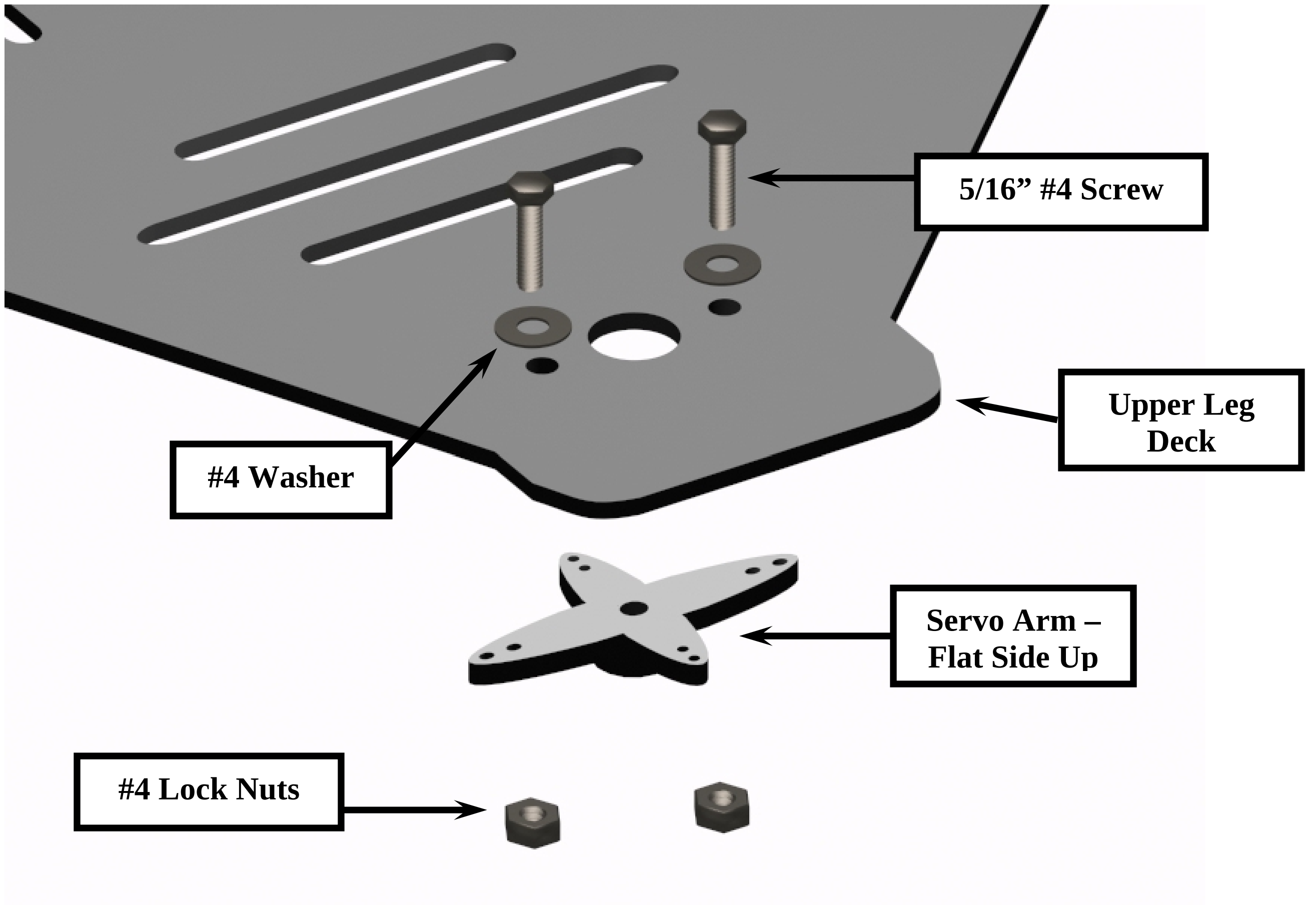


Figure 28: Upper Deck – Servo Arm Installation

3. Repeat step #1 and #2 for all of the leg decks.

INSTALLING THE SIDE SUPPORTS

4. Using (2) 5/16" #4 screws, install the side braces to the upper deck as shown in Figure 29.

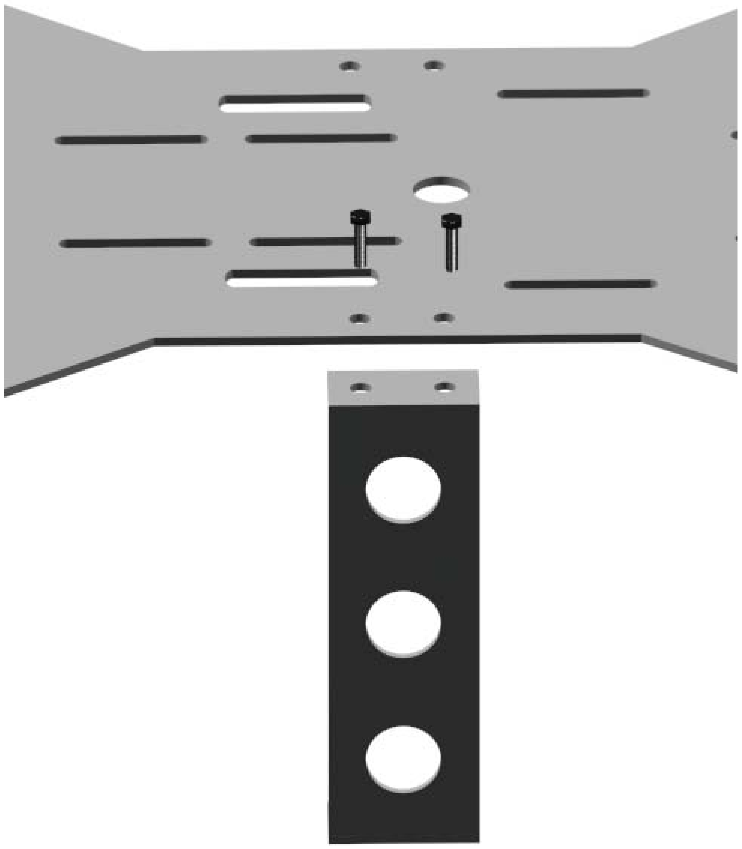


Figure 29

!	**CAUTION:** Ensure that the shape of the side support matches the shape in the upper deck when installing.

5. Repeat this process for the remaining side supports in your kit.

INSTALLING THE FRONT AND REAR SUPPORTS

6. Using (2) 5/16" #4 screws and washers, install the front and rear support brackets as shown in Figure 30 and Figure 30a.

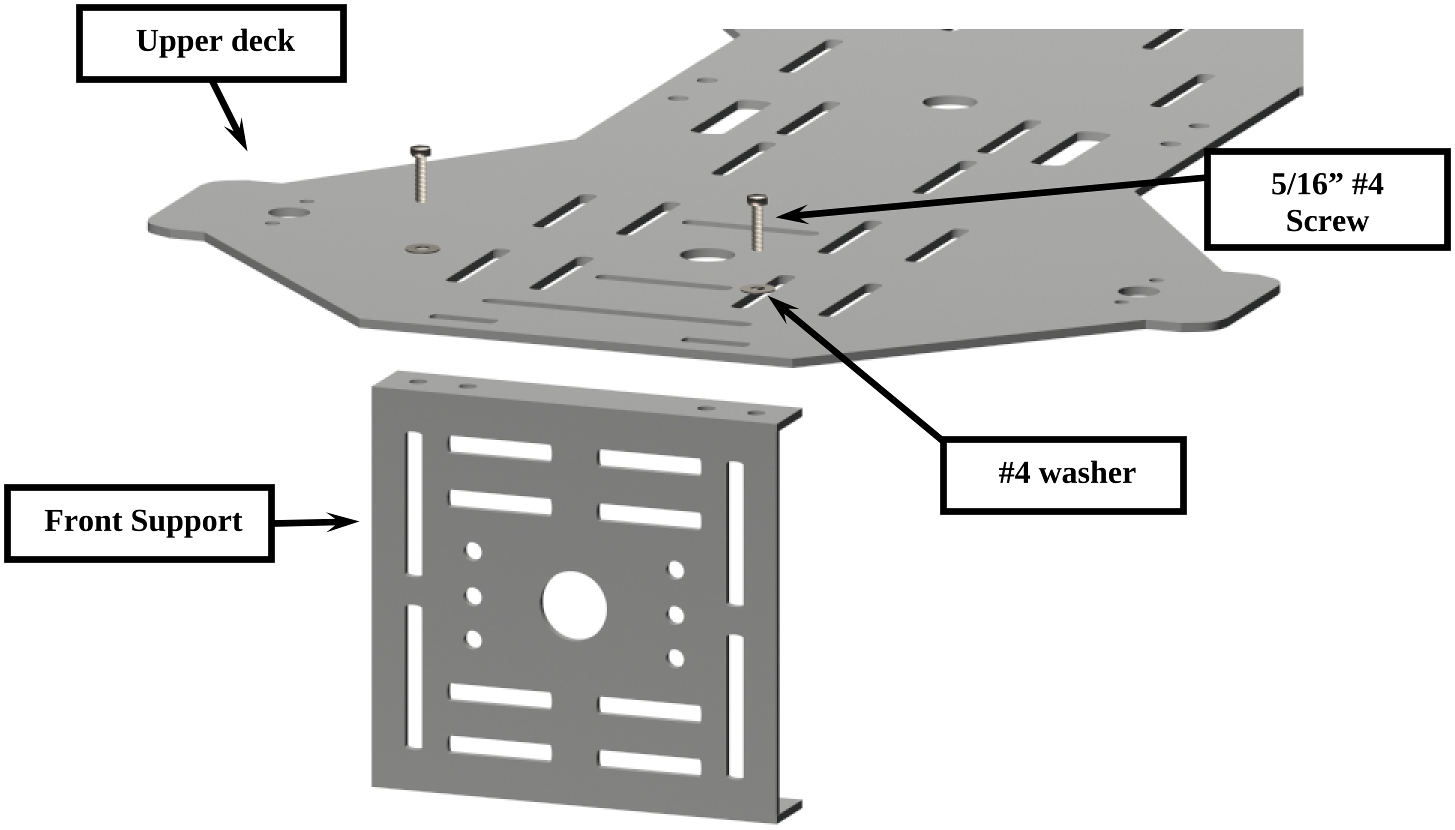


Figure 30

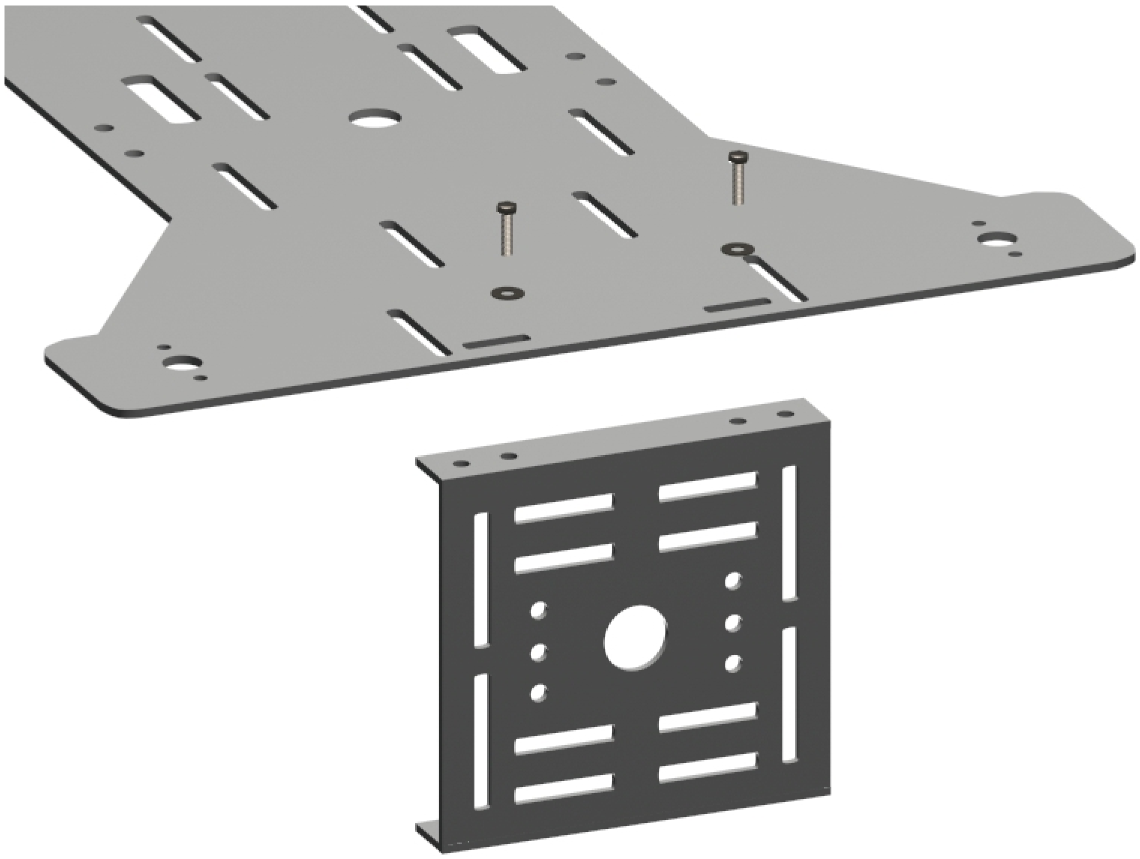


Figure 30a

INSTALLING THE LEG ASSEMBLIES TO THE UPPER DECK

1. To ensure that the servo wires do not get in the way of the following steps, secure the servo wires with the cable ties provided in your kit.

CAUTION: Perform the following procedures one leg at a time.

2. Place the assembled upper deck on a clean, flat surface with the front and side braces facing up as shown in Figure 31.

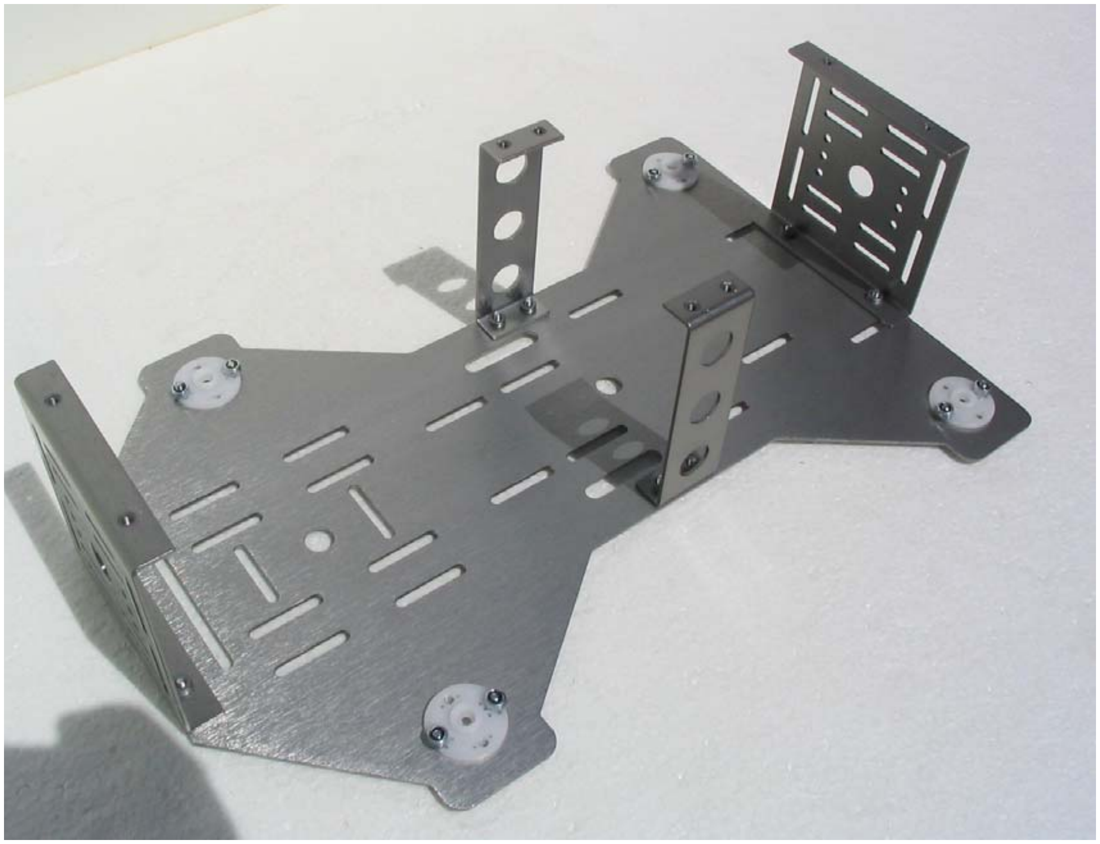


Figure 31: Upper Deck Assembly

3. Take a completed leg assembly and insert the legs upper round servo spindle into the upper decks round servo arm as shown in Figure 32.

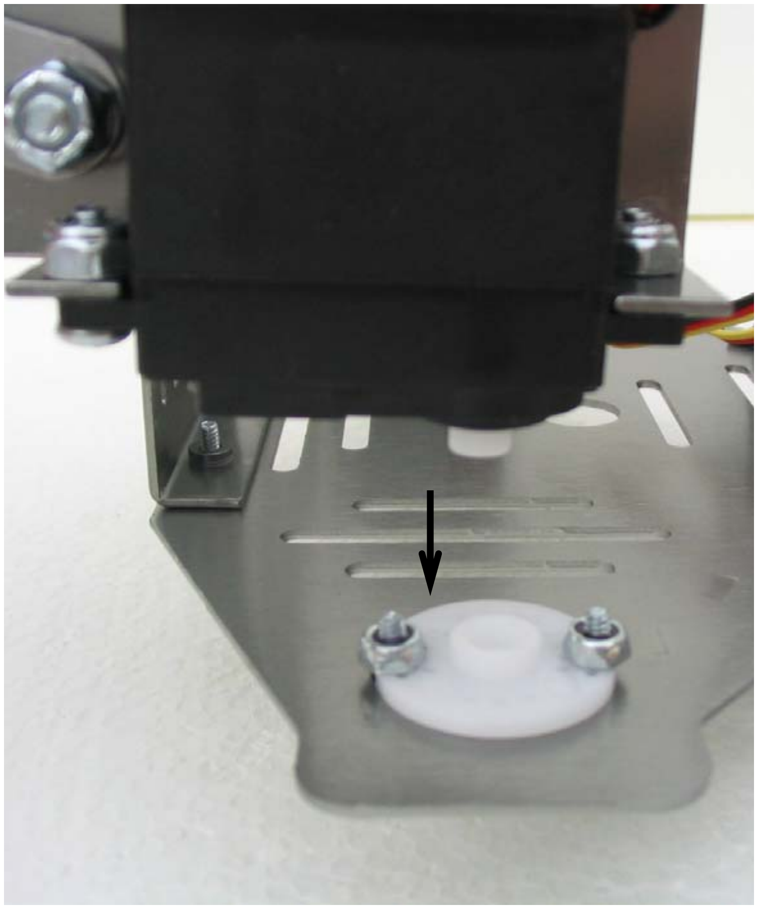


Figure 32: Installing a Completed Leg Assembly

!	**CAUTION:** Remember to attach the correct leg to the correct side of the upper deck. The back face of the servos should <u>always</u> be facing towards the rear of the upper deck.

4. Once inserted (do not secure it with the stock servo screw yet), gently swivel and adjust the leg (taking it out of the servo arm rotating the leg clockwise or counterclockwise and then re-attaching it to the servo arm) so that it swings freely from approximately the 10 o’clock position to the 2 o’clock position as illustrated in Figure 33.

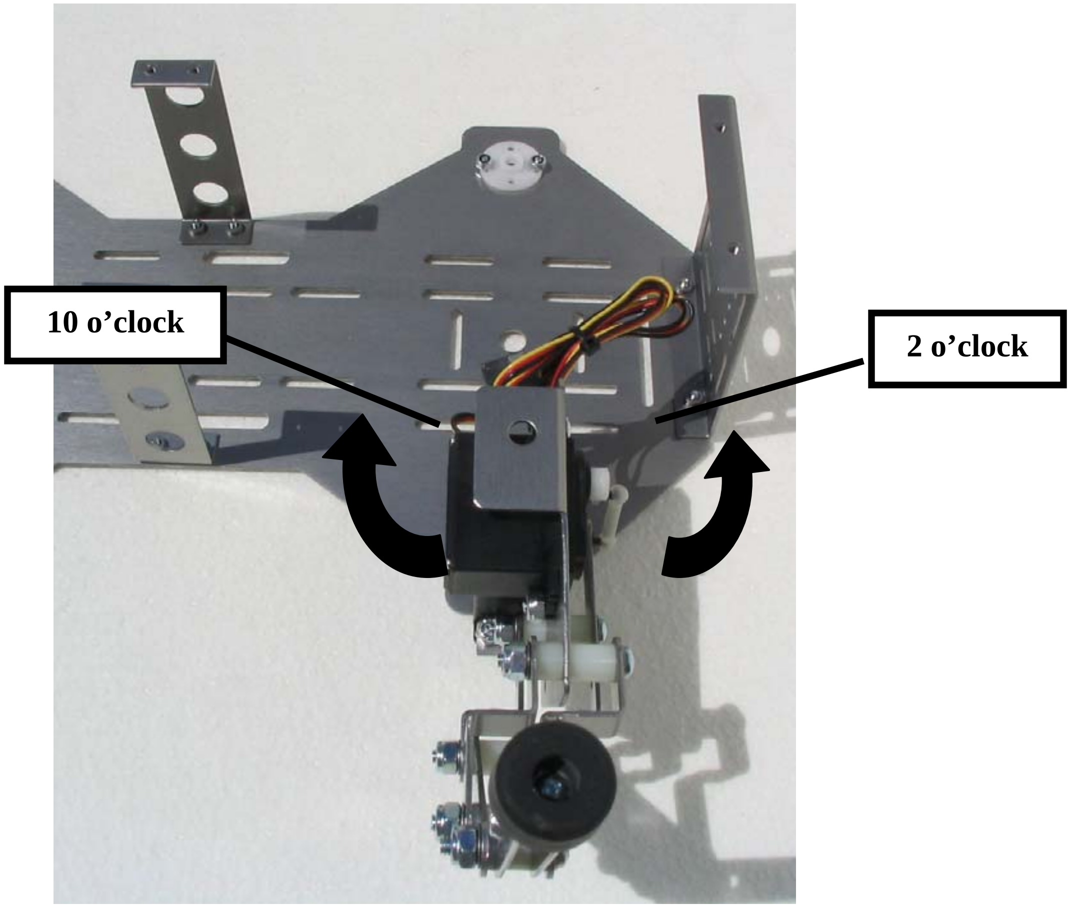


Figure 33

5. Once the leg is set correctly, secure the leg using the stock servo screw as shown in Figure 34.

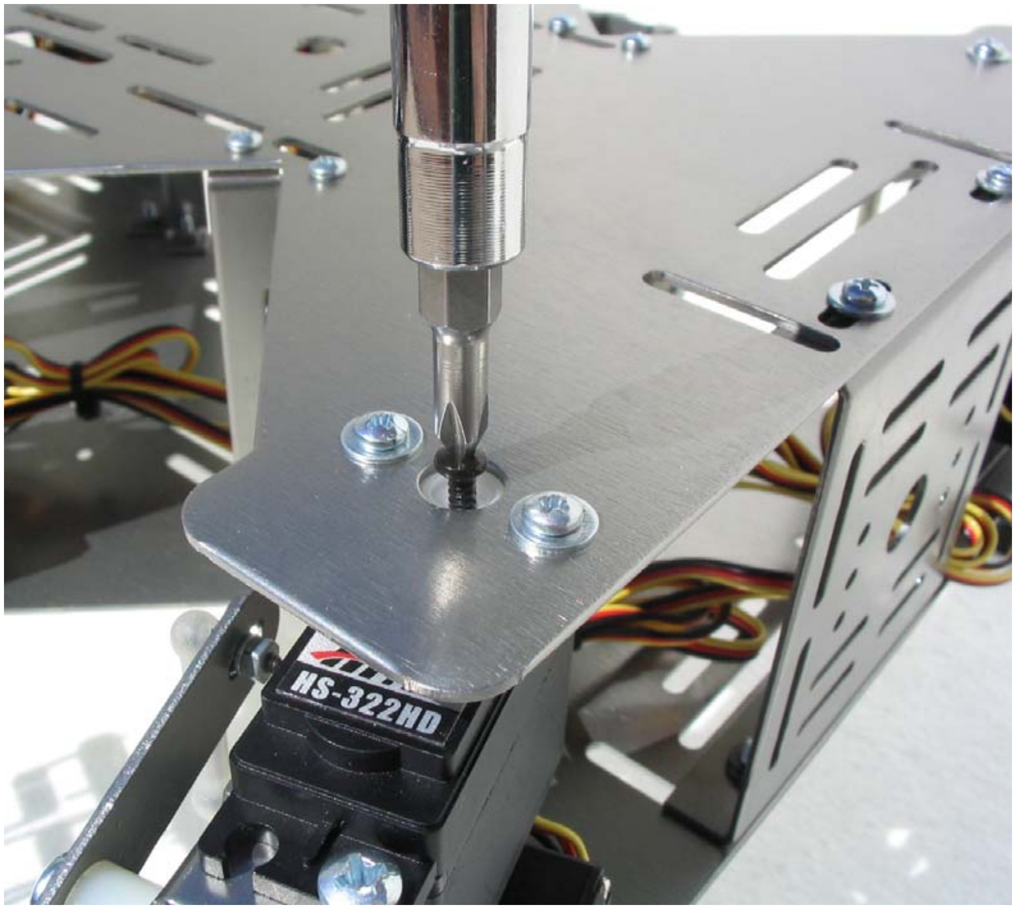


Figure 34 – Securing the leg assembly

6. Repeat steps 3 through 5 for the remaining (5) legs in your kit. Once all of the legs have been installed, your QuadCrawler should appear like Figure 35.

7. Place the upper deck assembly to the side of your work area.

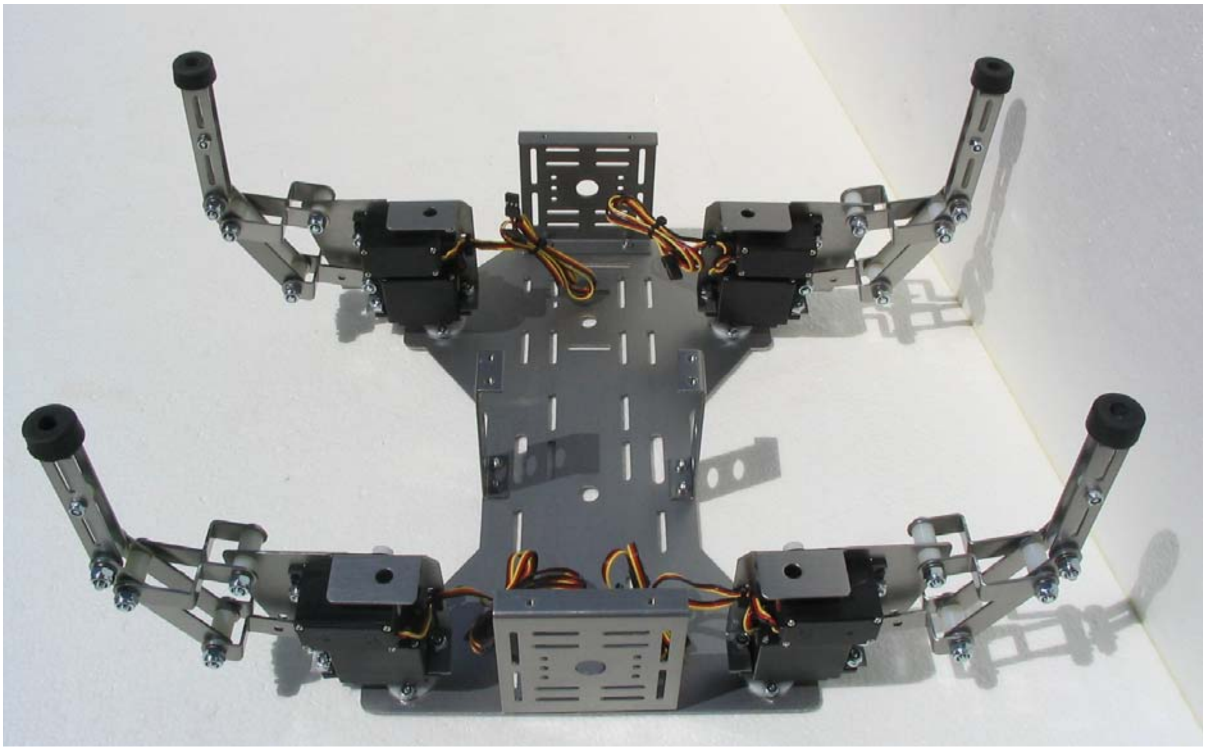


Figure 35: Completed Upper Deck Assembly

PREPARING THE LOWER DECK

1. Apply white grease (or any heavy grease) around the base of all 4 pem studs of the lower deck as shown in Figure 36.

2. Install the SAE ¼ flat washers (flat side down) to each of the pem studs and apply grease to the top of the SAE flat washers as shown in figure 37.

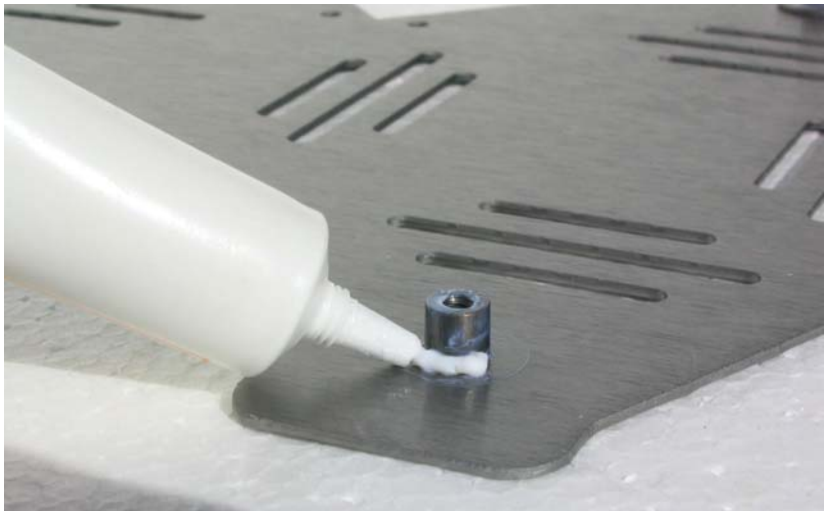


Figure 36: Grease application – Pem Stud

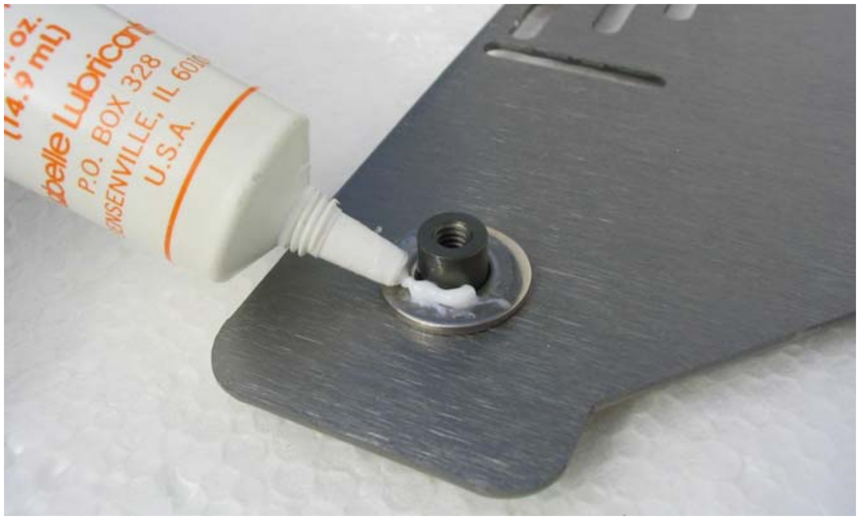


Figure 37: Grease application – SAE Flat washer

ATTACHING THE UPPER DECK ASSEMBLY TO THE LOWER DECK ASSEMBLY

ALTERNATIVE: Dead-bug assembly is also possible in this step. The white grease will hold the washers on the lower deck in place. Place the robot on it's back and then lower the bottom deck onto the bottom of the QuadCrawler.

1. In order to properly attach the upper deck assembly to the lower deck assembly, you will need to place the lower deck assembly onto an elevated platform. A pile of books or a box that is slightly wider and longer than the lower deck assembly will work fine. The height of the books or box combined should be greater than 3.5" to 4.00" inches (See figure 38). This height will allow for clearance of the legs when attaching the upper and lower deck assemblies together.

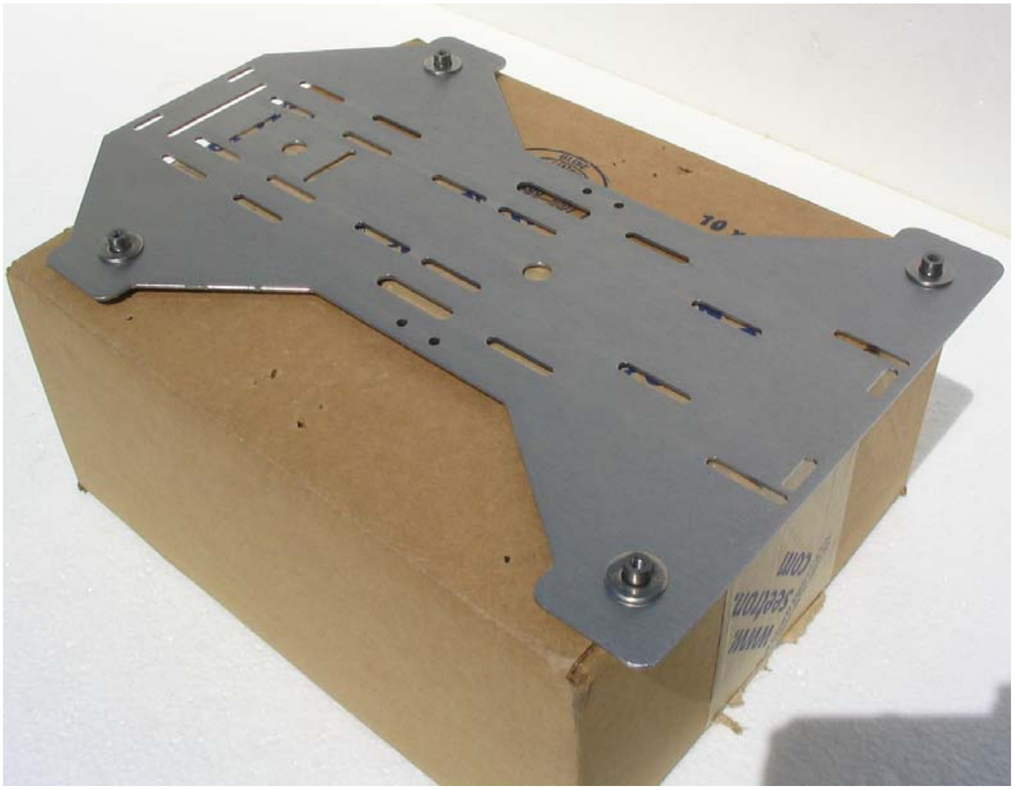


Figure 38

2. Carefully take the upper deck assembly and lower it onto the lower deck assembly, ensuring that the pem studs align with the leg pivot holes at the bottom of each of the leg assemblies.

3. Once **all** of the pem studs are aligned and inserted into the leg pivot holes, hold both decks together and place the QuadCrawler onto its back. Secure the lower deck to the upper deck with #4 5/16” screws and washers (where applicable) as shown in figures 39, 40 and 41.

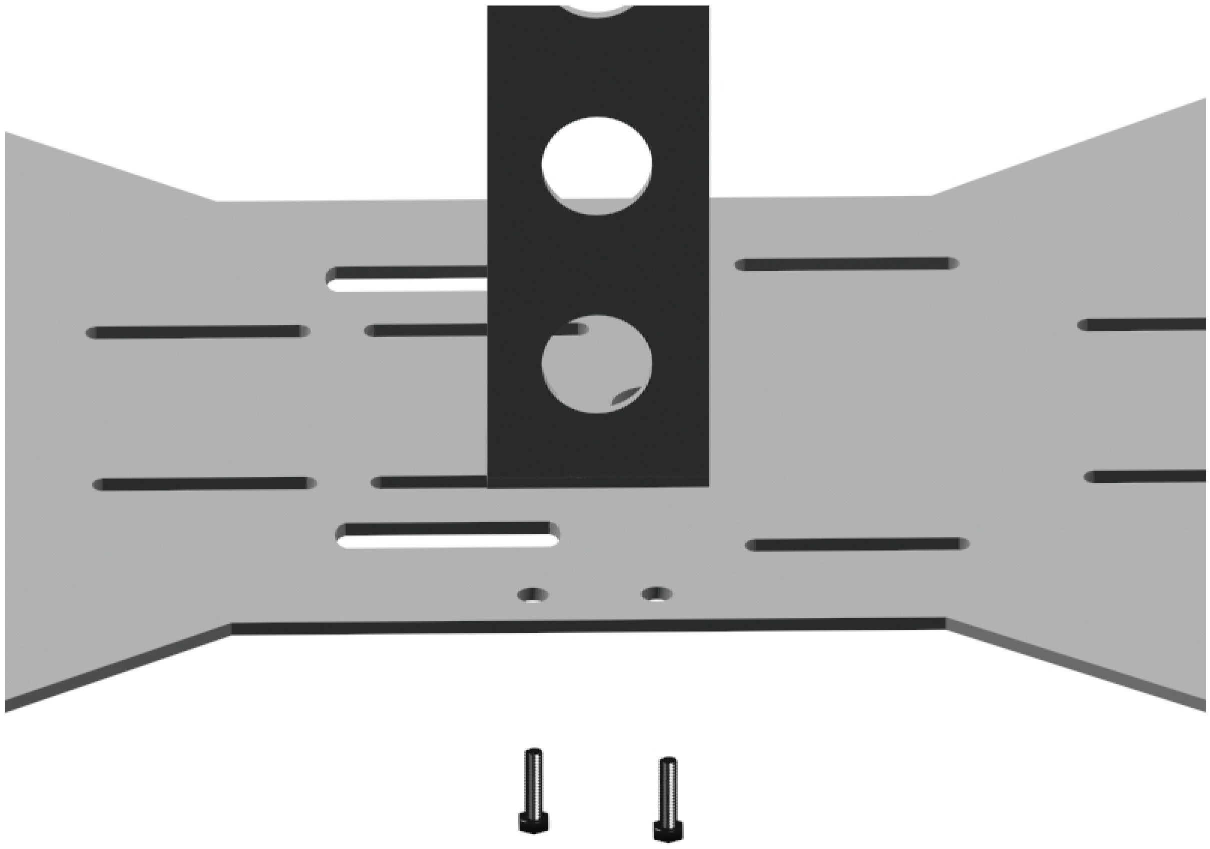


Figure 39: Side Support Assembly

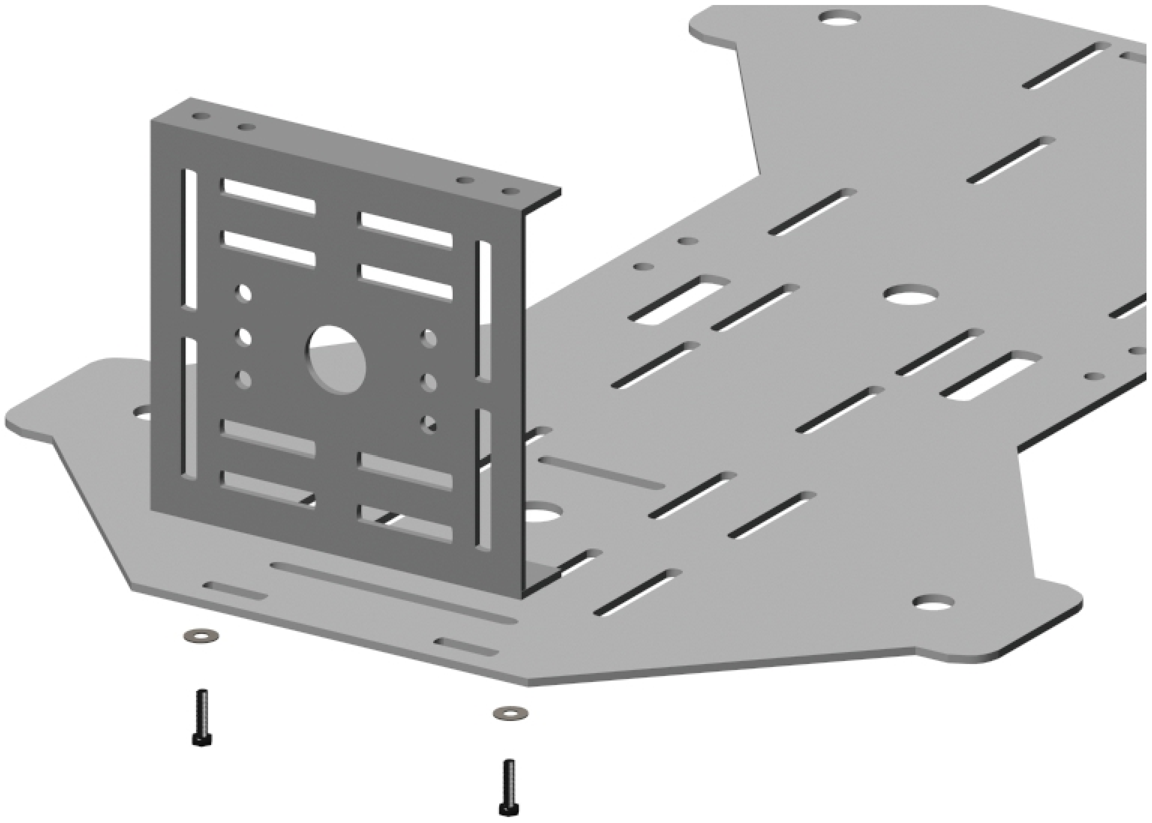


Figure 40: Front Support Assembly

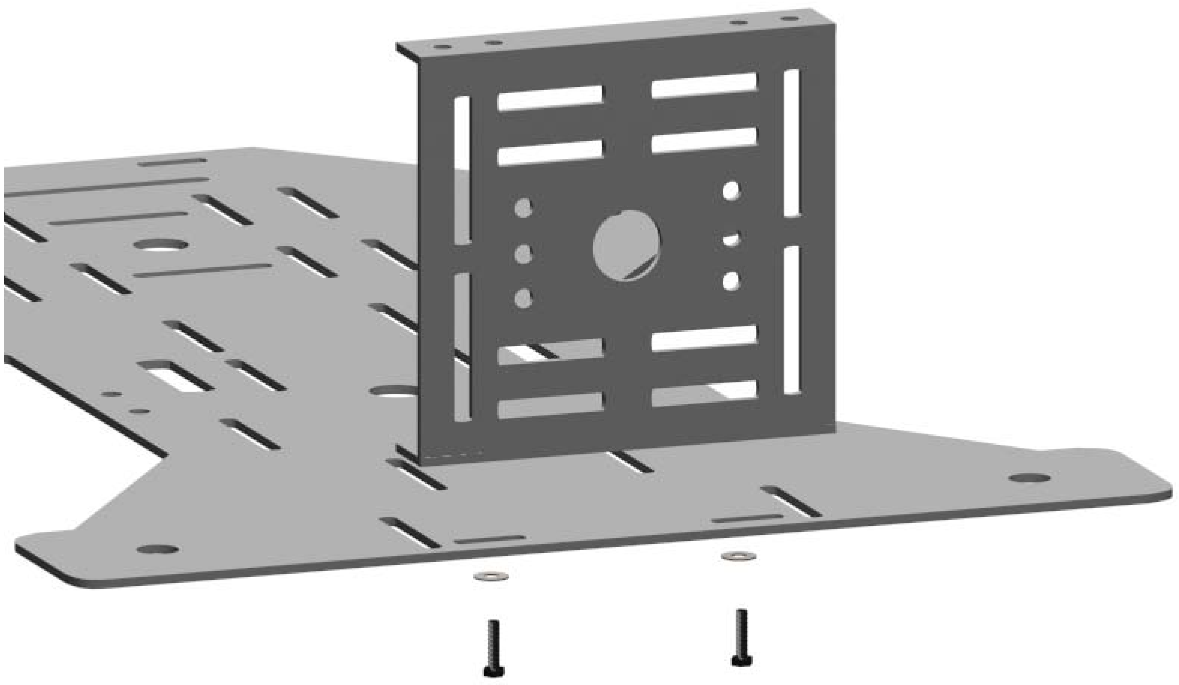


Figure 41: Rear Support Assembly

5. Carefully place the QuadCrawler back onto the elevated platform in the standing position.

INSTALLING THE “DOGBONES” AND SERVO ARMS

1. With a 1/8” drill bit, drill out the second hole from the center hole of the servo arms (4 total) as shown in Figure 42. It does not matter which side of the (2) arms are drilled. To secure the servo arm while its being drilled, place the arm on an available servo’s spindle.

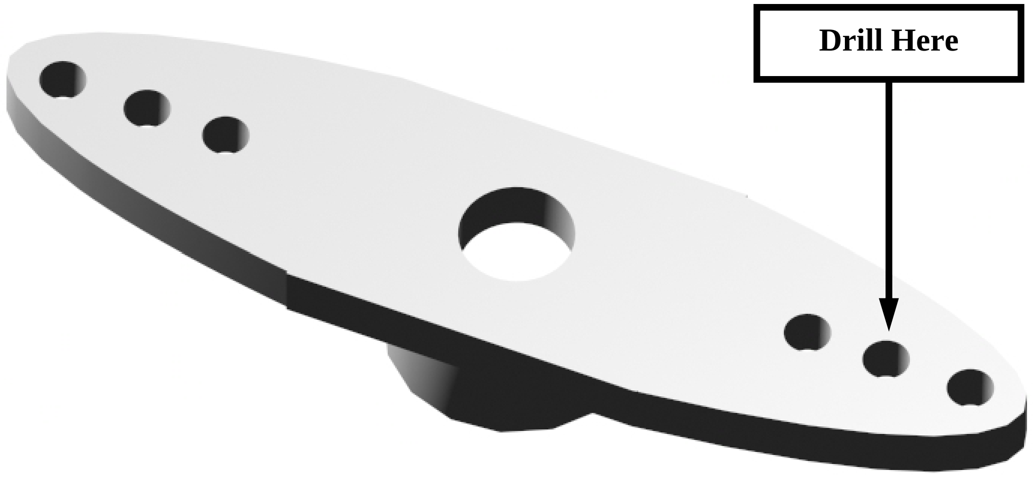


Figure 42

2. Attach the ball link with the #2 washer, lock washer and lock nut (Figure 43). The ball should be orientated towards the flat side of the servo arm.

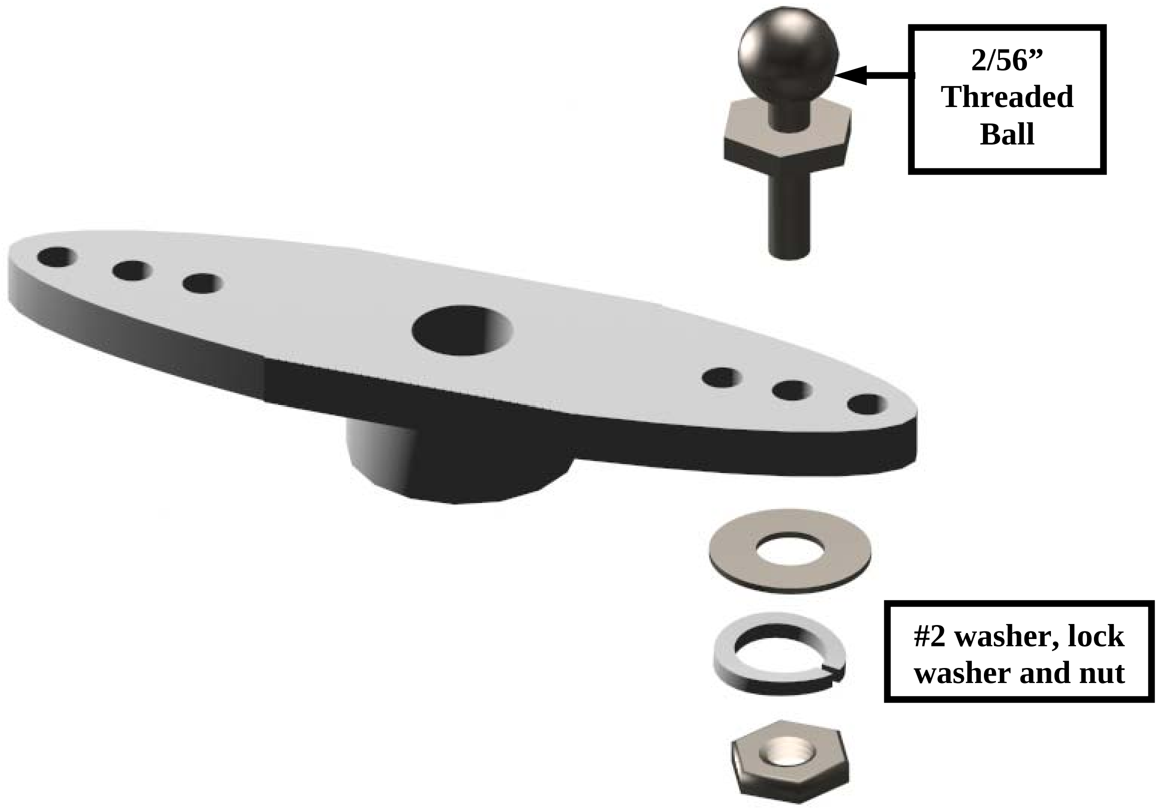


Figure 43

3. With the 2/56" threaded ball oriented upwards, install the servo arm to the servo spindle on the lower servo of the leg assembly and secure the assembly with the stock servo screw (see Figure 44.

Note: Cut off the remaining arm on the servo horn, which is not being used before the threaded ball is installed. Failure to remove the remaining servo arm will result with the servo arm hitting the servo spindle during the QuadCrawlers walking sequence.

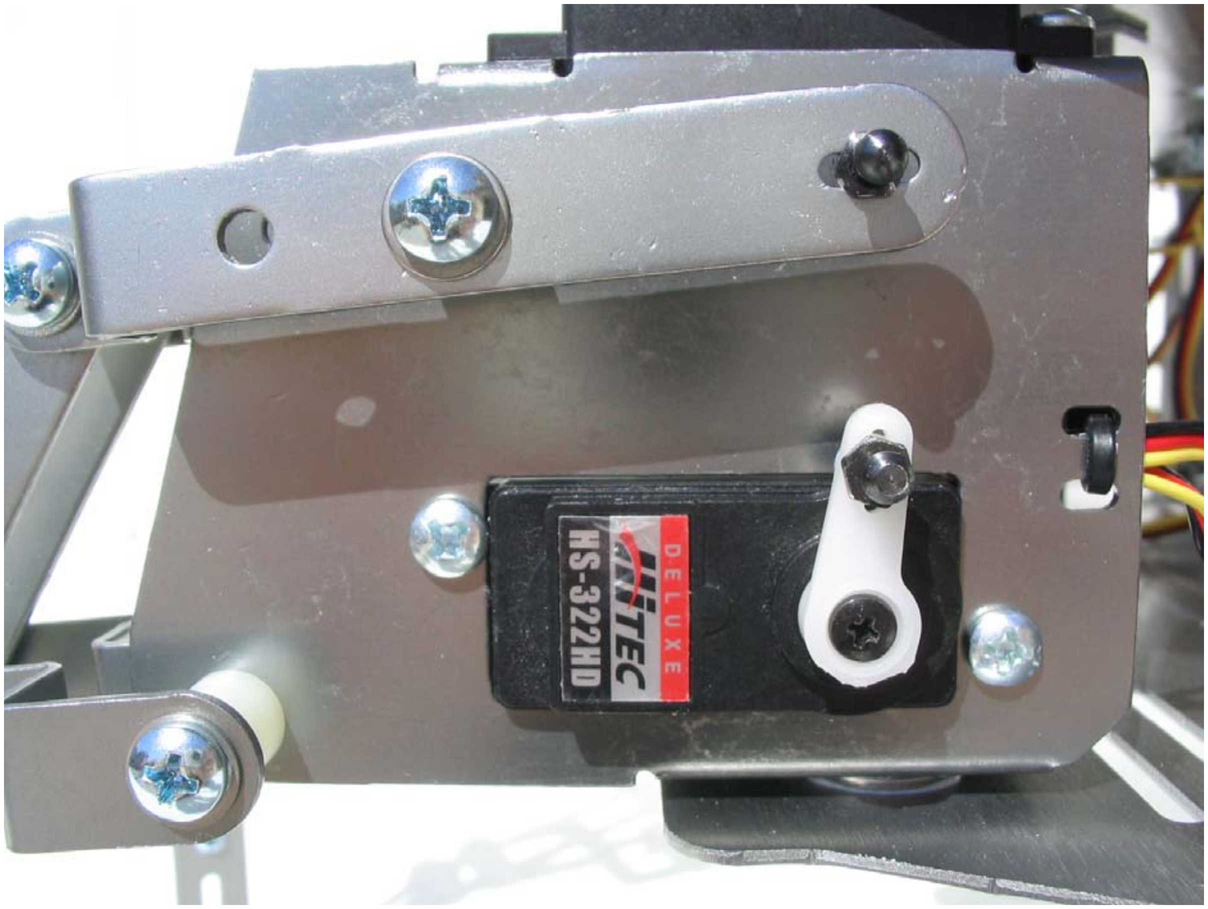


Figure 44

4. Join the 2 dog bones by threading the 2/56" threaded rod into each end of the dog-bones. Install the threaded rod completely into 1 dog-bone before threading the other (Figure 45). Repeat this step for all of the legs on your QuadCrawler. This is often easiest done with a small vise. Don't grip the dog bones or the threaded rod too hard or you could damage the parts.

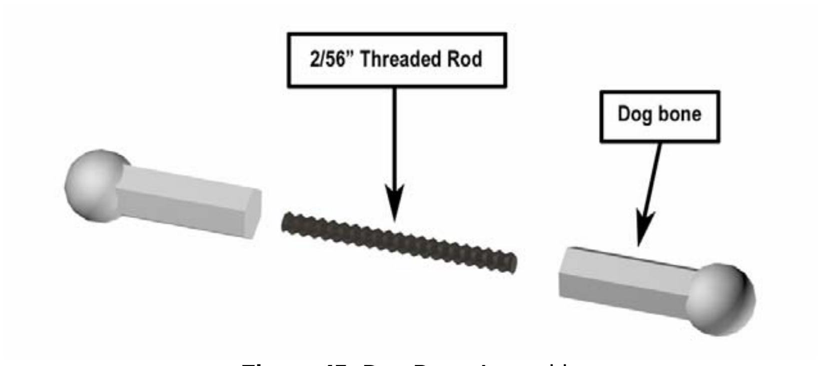


Figure 45: Dog Bone Assembly

!	**Tip**: If turning the threaded rod onto the dog bone is difficult by hand, place the 2/56" threaded rod into a padded vise and use an adjustable wrench to turn the dog-bone onto the threaded rod.

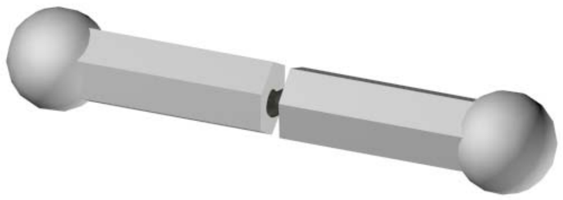


Figure 46: Completed Dog Bone Assembly

!	**CAUTION**: If your dog bones do not thread together correctly, don't use glue to hold them in place. You have two choices for fixing this problem: • Contact Crustcrawler for replacement parts support@crustcrawler.com • Go to a local hobby shop and get Du-Bro #188 ball link sockets for replacement sockets and standard 2/56 threaded rod.

5. Snap on the assembled dog-bone to the 2 ball joints on the leg assembly. Repeat steps 4 and 5 for each leg (see Figures 47A and Figure 47B).

!	**Warning:** Ensure that you support the back of the main leg actuator when snapping on the dog-bone to the ball joint or the leg actuator may bend with excessive force.

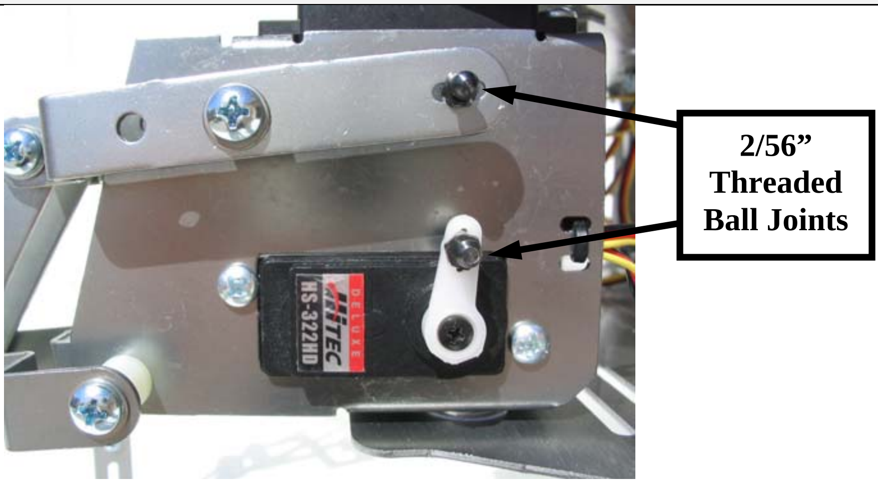


Figure 47A

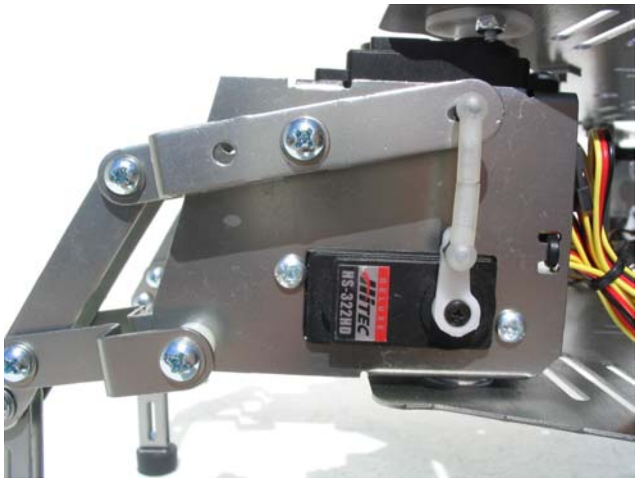


Figure 47B

6. **This is one of the most important steps in this manual.** Facing the front of the QuadCrawler so the servo horns are showing, gently turn and re-install as necessary the servo control horn (keep the screw off for the moment) so that it swings from 11 to 7 o'clock in the **clockwise** direction for the legs on the QuadCrawler's **right side** and from 1 to 7 o'clock in the **counterclockwise** direction for the legs on the **left side** of the QuadCrawler (see figures 48 and 49 below).

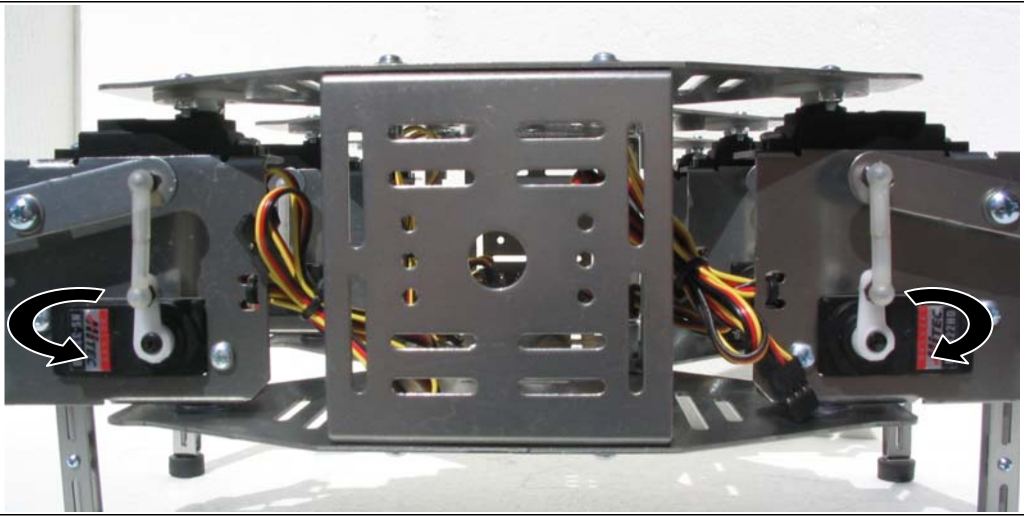


Figure 48 – Servo Configuration – Starting position

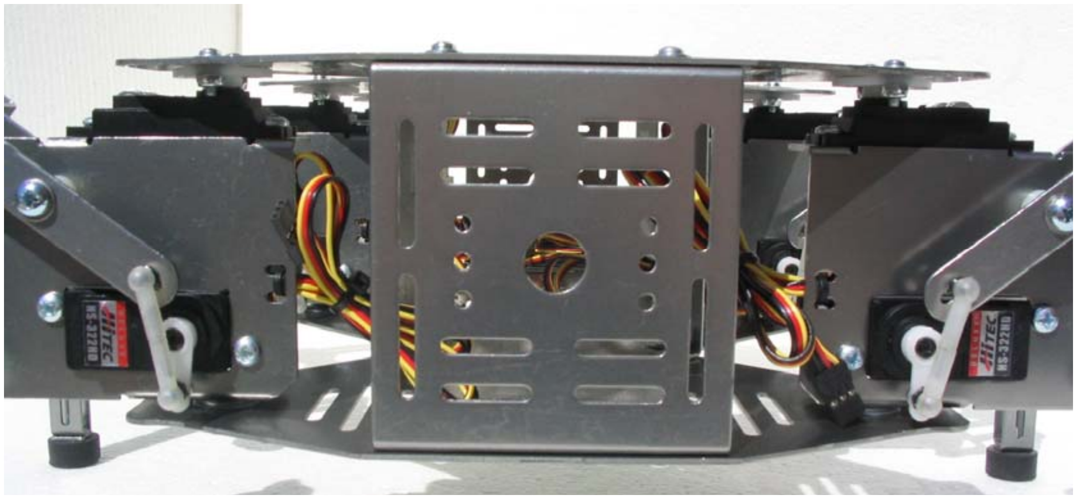


Figure 49 – Servo Configuration – Ending position

Repeat steps 1 thru 6 for all of the legs of your QuadCrawler.

"TUNING" THE LEGS OF YOUR QUADCRAWLER

1. Using a 11/32" wrench or socket along with a Phillips screwdriver, tighten all of the lock nuts on the 1.25" #8 screws just tight enough so they can be turned by hand. (screws 1 – 4 only, see Figure 50)

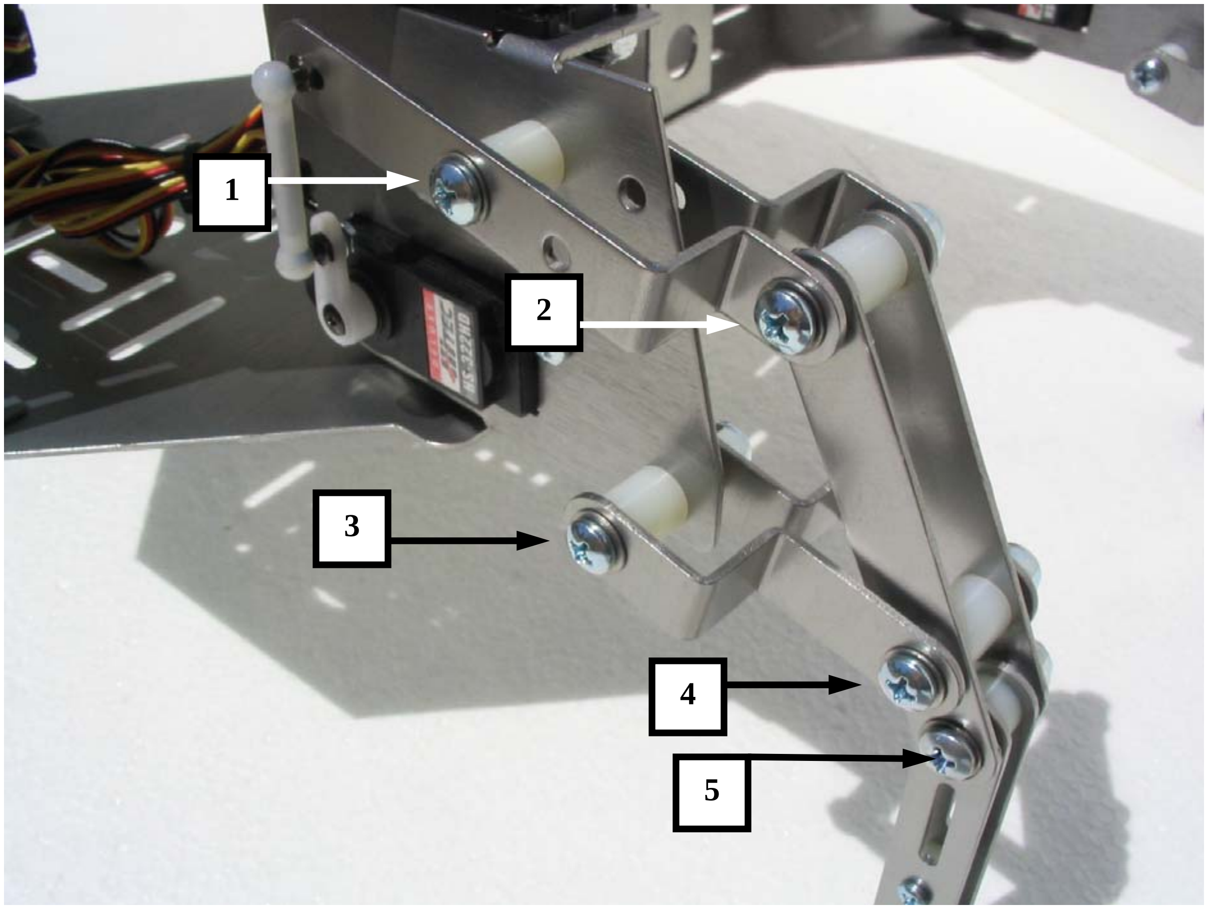


Figure 50

2. Ensure the lower leg is as vertical as possible (for best accuracy, use a level) and then completely tighten the 1" #8 screw with the locknut (box 5 in Figure 50).

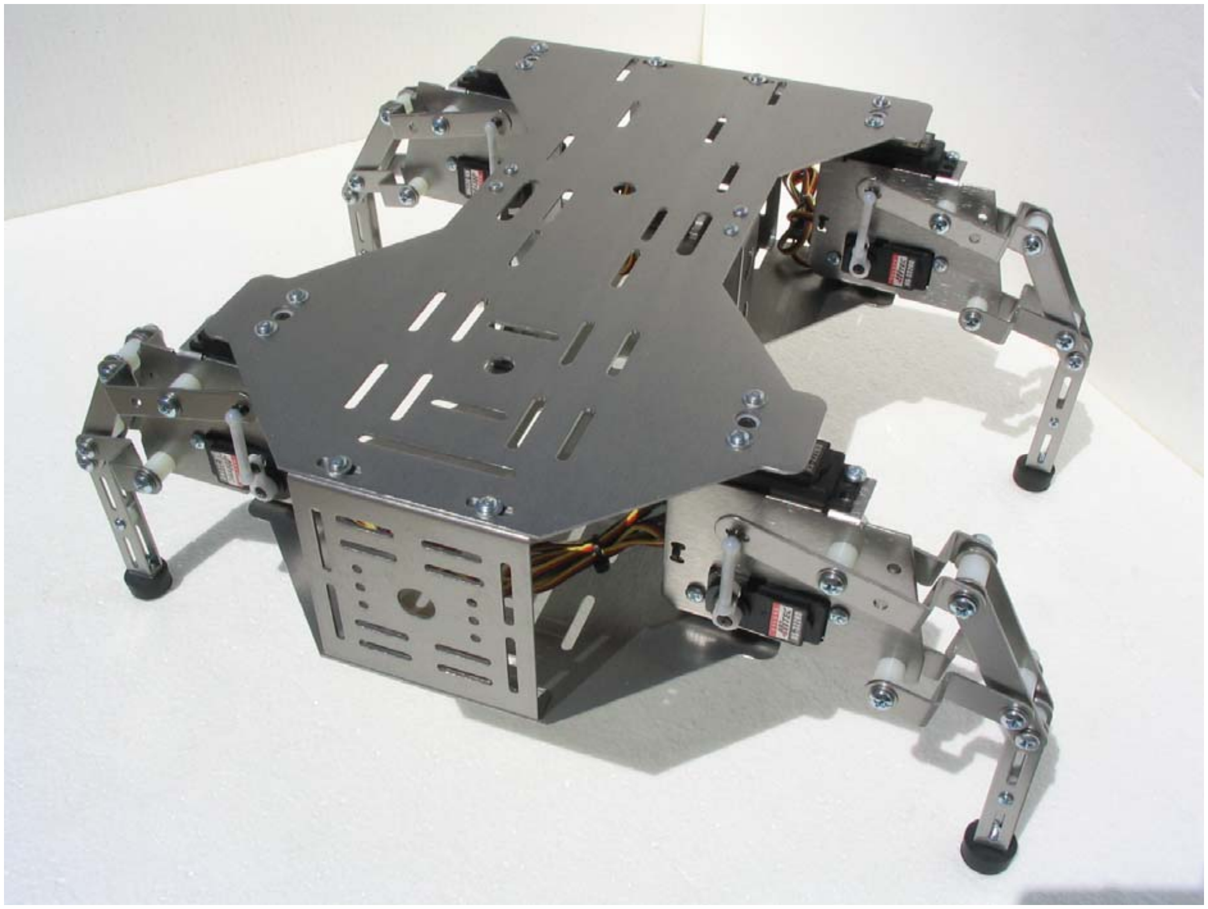
3. Repeat the steps above for all of the legs on the QuadCrawler.

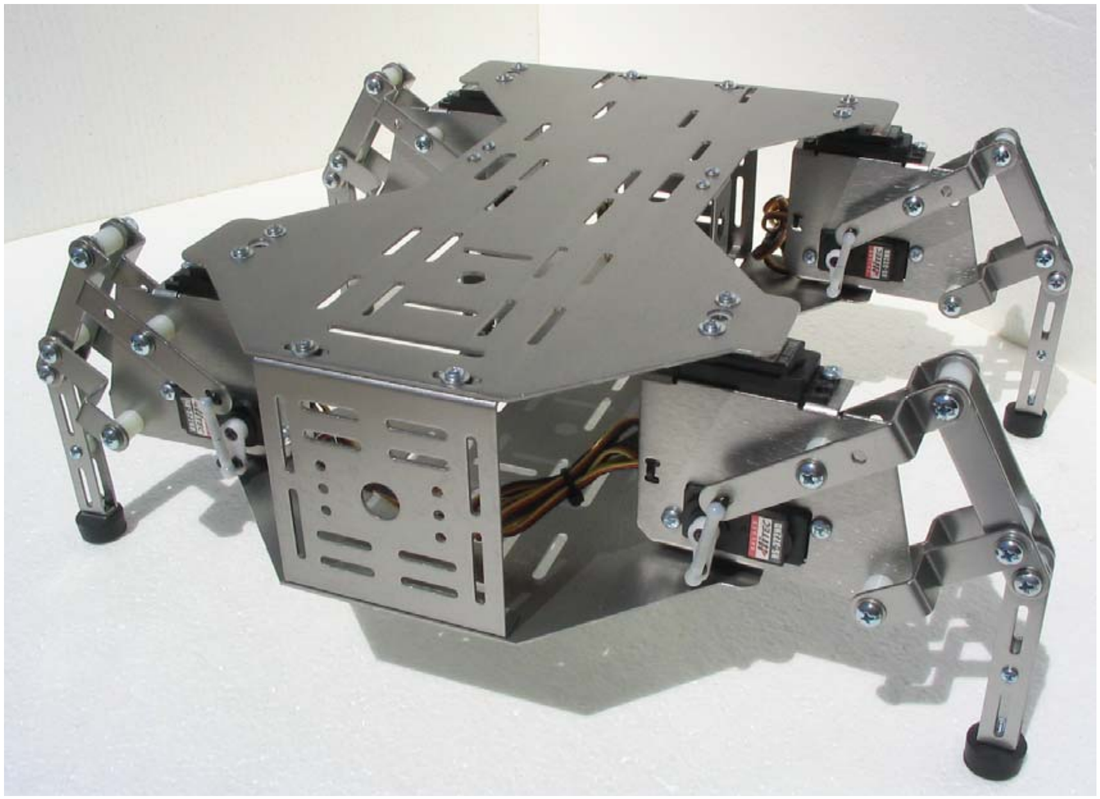
4. After tightening each leg as per the steps outlined above, perform the following leg screw check:

- **Tight Screws -** Move the vertical lift servo so that the leg moves up and down and ensure that the leg moves freely and is not difficult to move or appears stiff. If the leg is difficult to move or appears stiff, one or more screws are too tight. **Hand** turn each screw until the tight screw(s) are found and loosen them just enough so that they can be turned by hand.

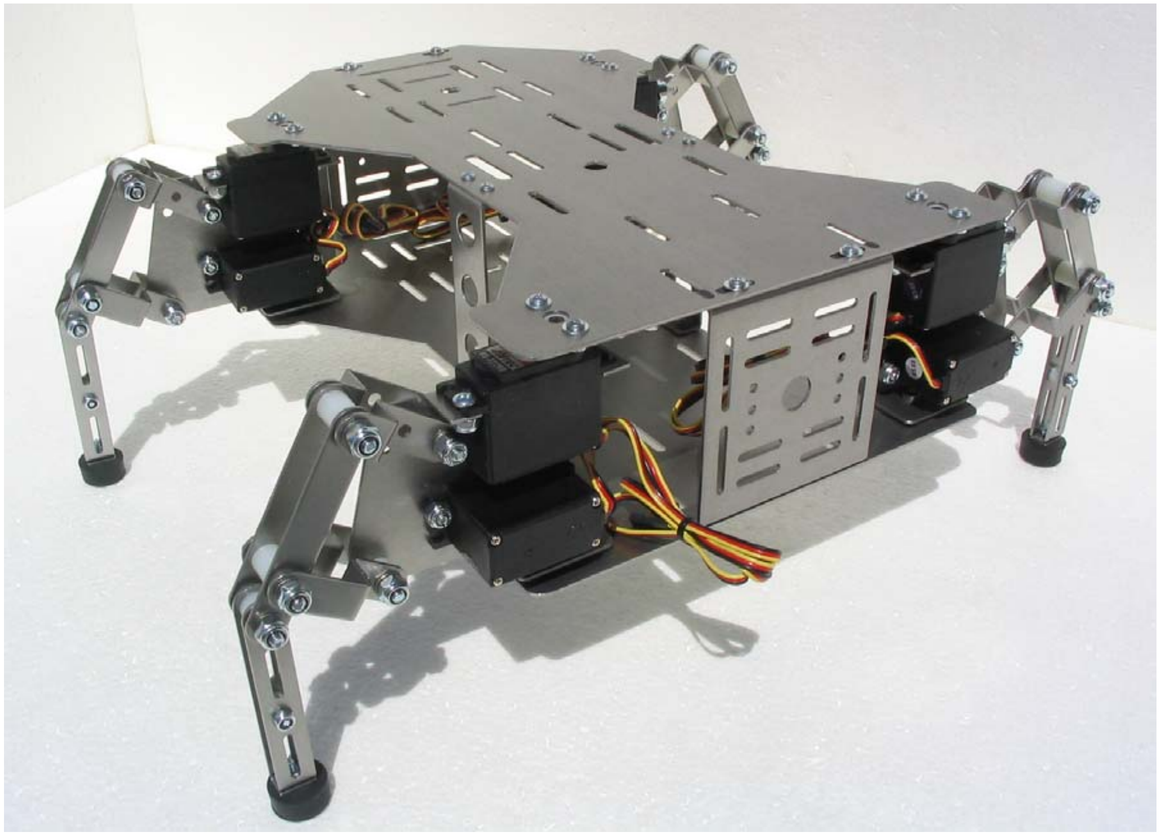
- **Loose Screws -** Move the vertical lift servo so that the leg moves up and down and ensure that the leg moves freely and that there is not too much horizontal play or "rattle" in the leg. If the leg appears loose, hand check each screw and tighten the screw just enough so that it can be turned by hand.

REFERENCE PICTURES

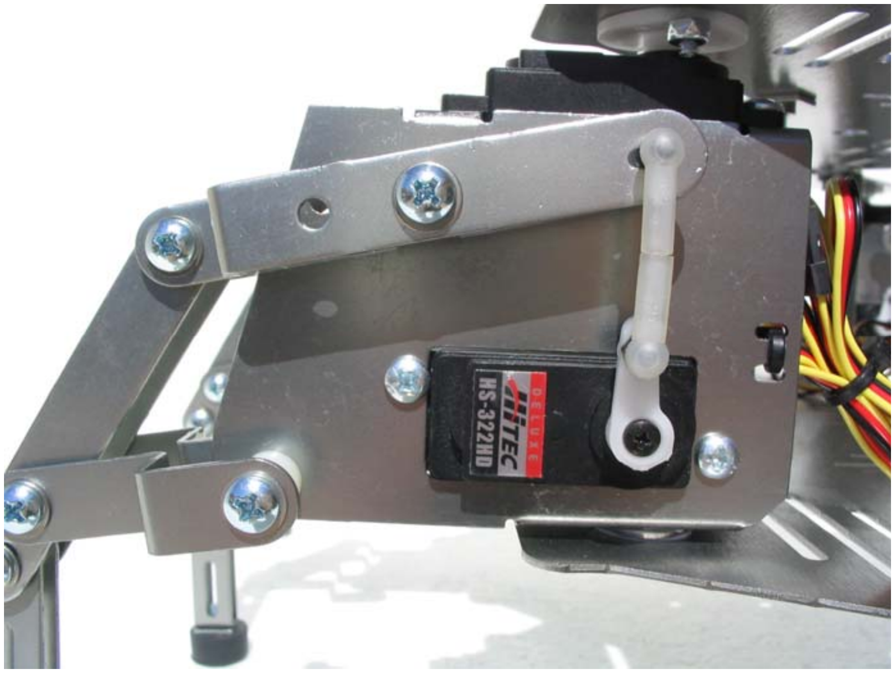

QuadCrawler – Front View- Standing



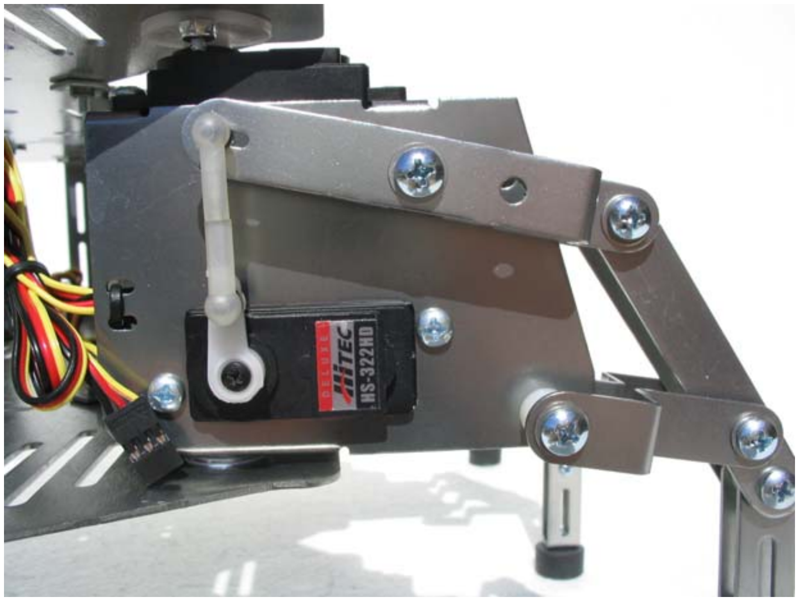
QuadCrawler – Front View- Squatting



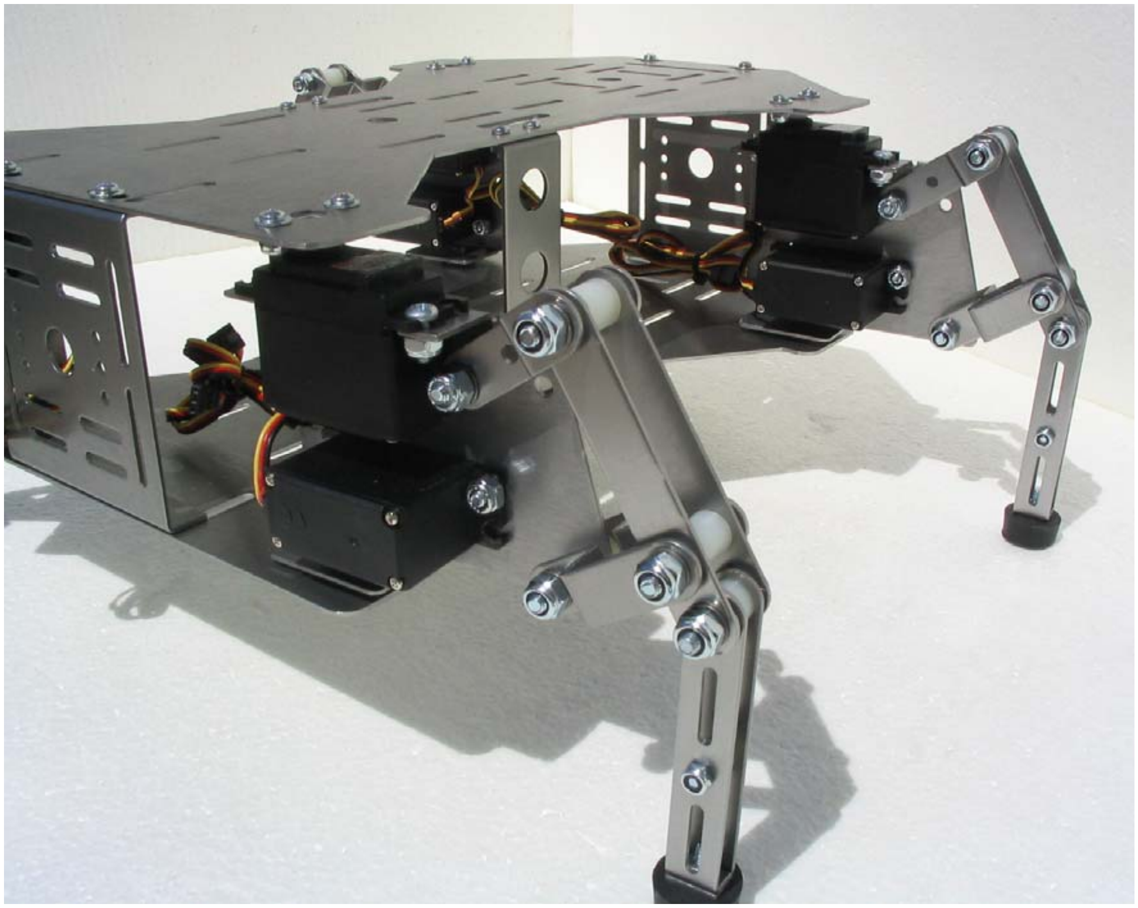
**QuadCrawler - ** Rear View



Left Leg Assembly



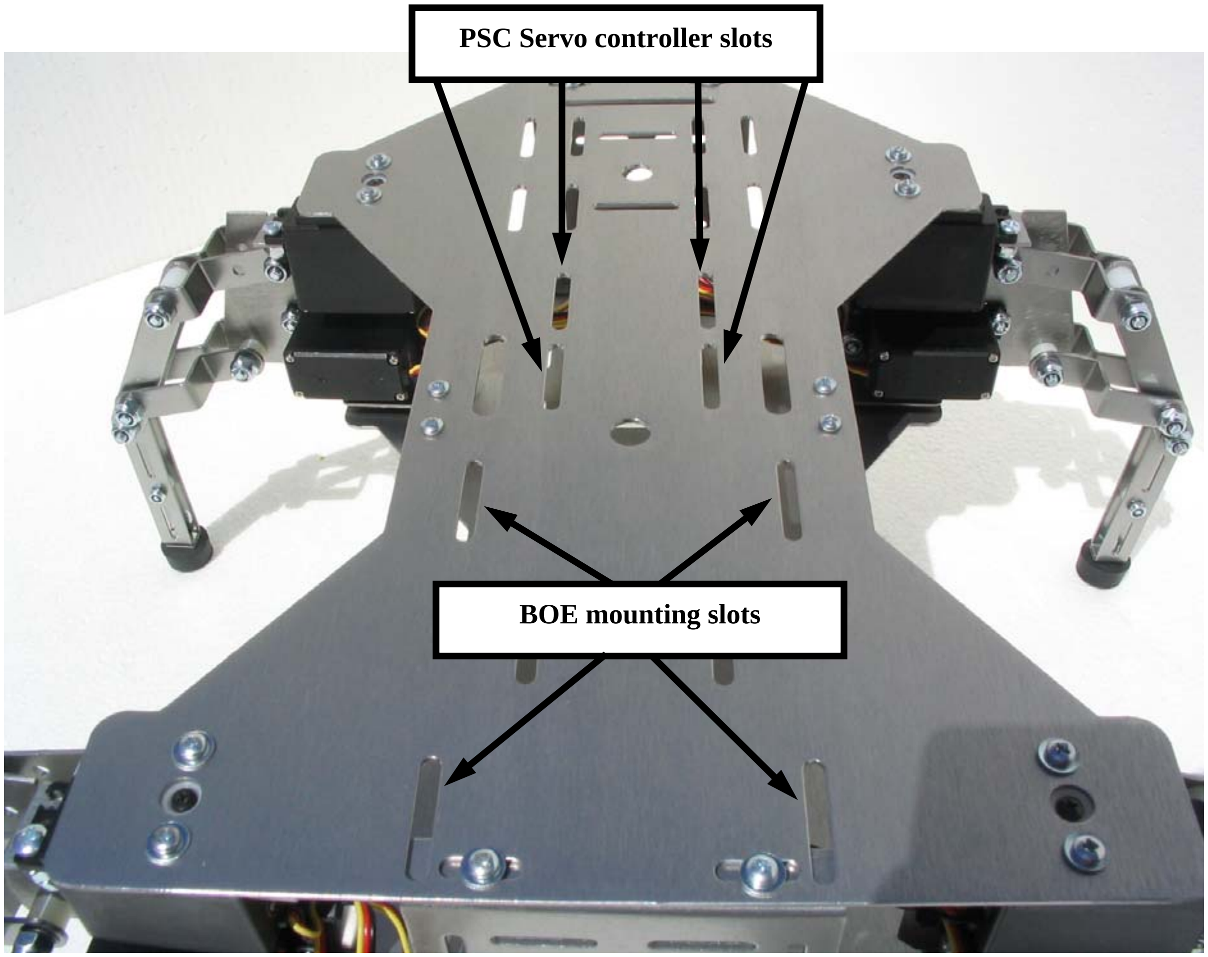
Right Leg assembly

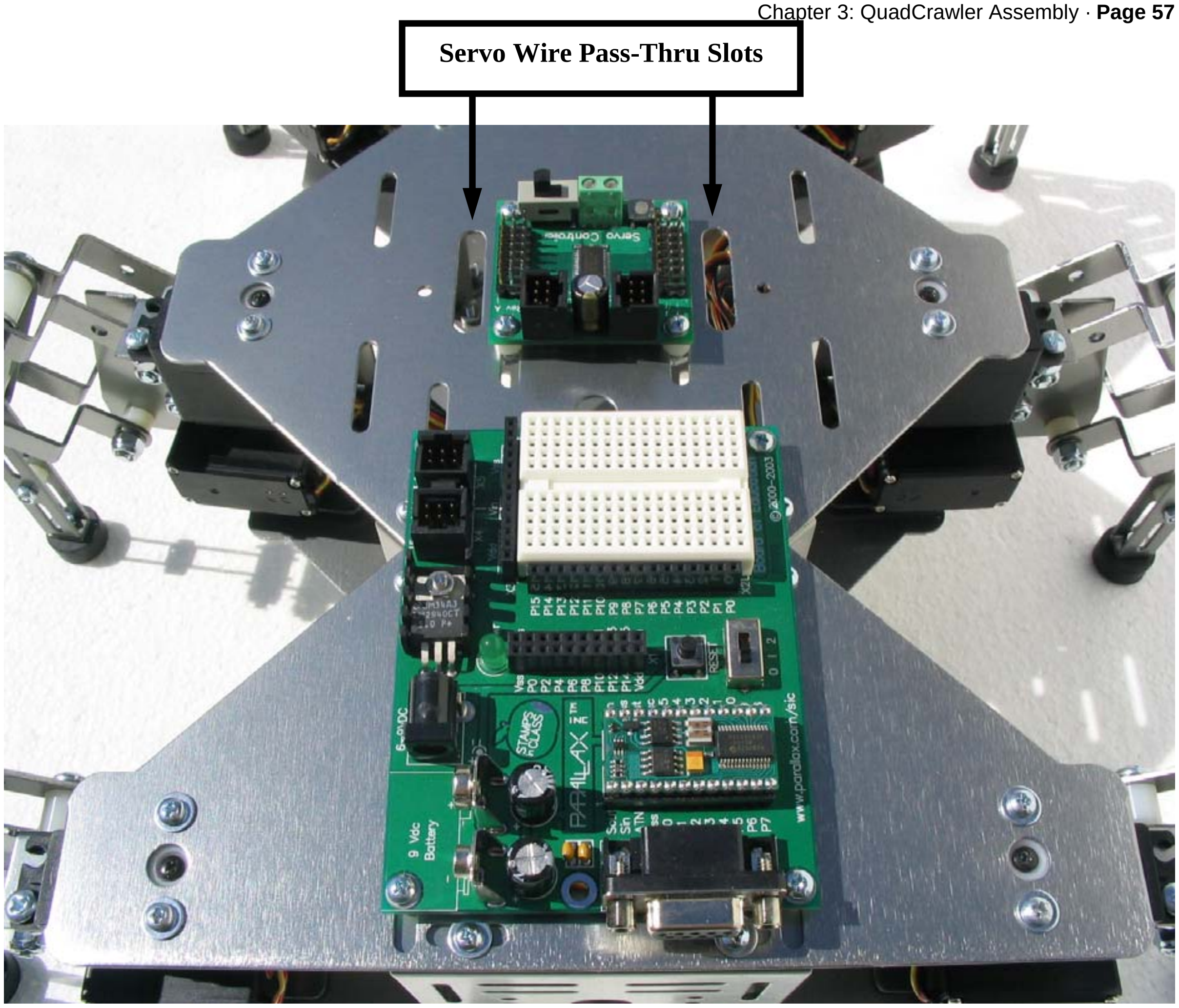


Close Up – Rear View

BOE AND PSC CONTROLLER PLACEMENT

Use the following pictures for the placement of the Parallax Board of Education (BOE) and PSC servo controller. Use the hardware from the "BOE/PSC Hardware Mounting Kit" provided in your QuadCrawler hardware kit.




Servo Wire Pass-Thru Slots

Chapter #4: Wiring and Tuning the QuadCrawler

This section describes how to wire circuits used in the QuadCrawler example programs in the next chapter.

PARTS REQUIRED

(1) BASIC Stamp 2 IC module on a Board of Education carrier board
(1) Parallax Servo Controller (PSC)
(8) Hitec Servos (already installed on the QuadCrawler)
(1) Seven-segment green LED (Fairchild MAN5760)
(7) 1 kΩ resistors
(2) Pushbuttons
(2) 10 kΩ resistors
(2) 220 Ω resistors
(1) 7.2 V battery
(1) 9 V transistor battery (if you intend on powering the BASIC Stamp module separately)
(misc) jumper wires

CONNECT THE PARALLAX SERVO CONTROLLER

Connect the QuadCrawler's servos to the Parallax Servo Controller as shown. Connect the Parallax Servo Controller to the Board of Education carrier board. The diagram below is for both the QuadCrawler and the HexCrawler. You will only need to hook up servos A thru H for the QuadCrawler.

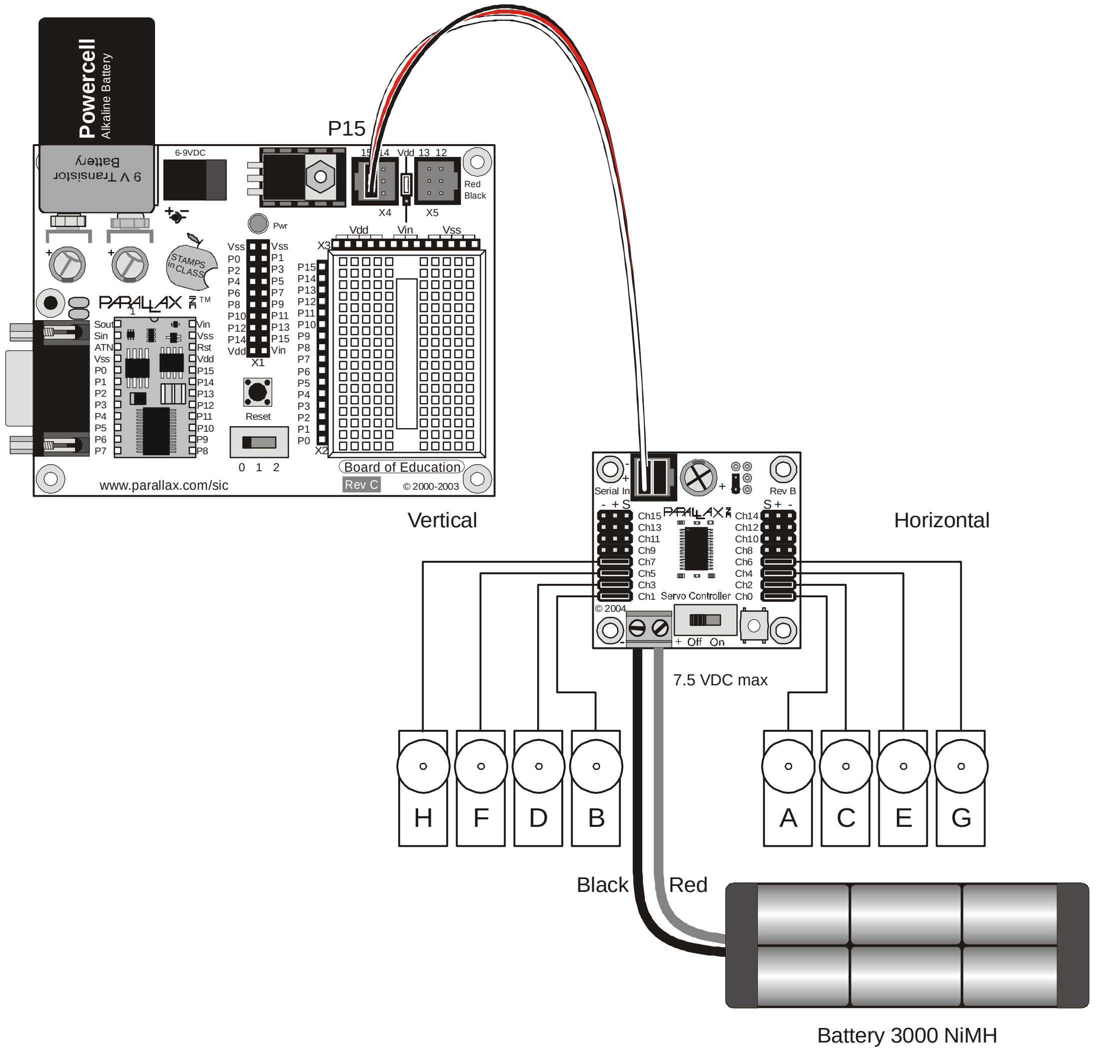
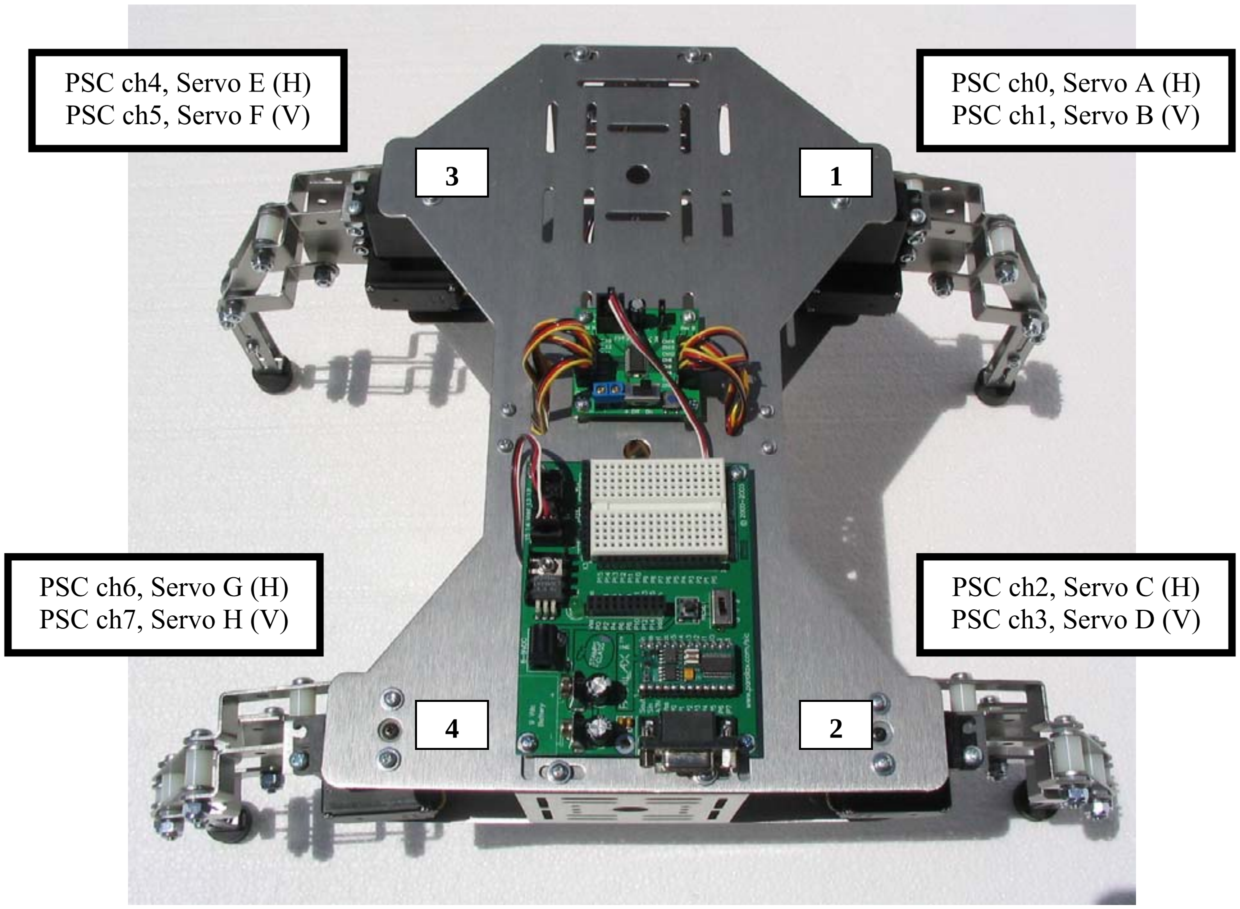


Figure 51



Servos moving legs vertically connect to odd numbered PSC ports.
Servos moving legs horizontally connect to even numbered PSC ports.

Figure 52

PUSHBUTTON AND LED CIRCUIT

The QuadCrawler's example program is run using a two-pushbutton, seven-segment LED display system feedback. Holding down the two pushbuttons starts/stops the selected gait and each button either toggles the number up or down the LED display.

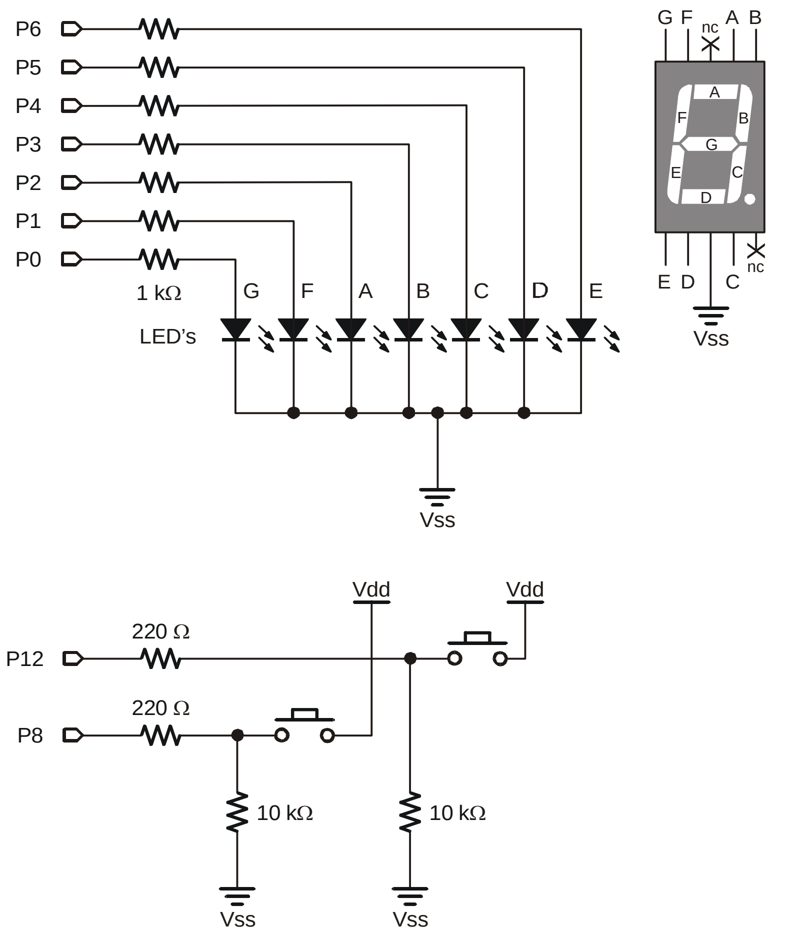


Figure 53: Pushbutton Control System Schematic

SHARP INFRARED AND ADC0831 CIRCUIT

Your QuadCrawler kit is equipped with a Sharp GP2D12 Infrared detector and an analog to digital converter (ADC0831). The circuit in figure 54 converts the analog signal coming from the Sharp infrared detector to a digital signal the BS2 can use. Visit the CrustCrawler web site (www.crustcrawler.com) for the latest code and updates that utilizes this schematic. We have also provided code that utilizes this circuit at the end of this manual.

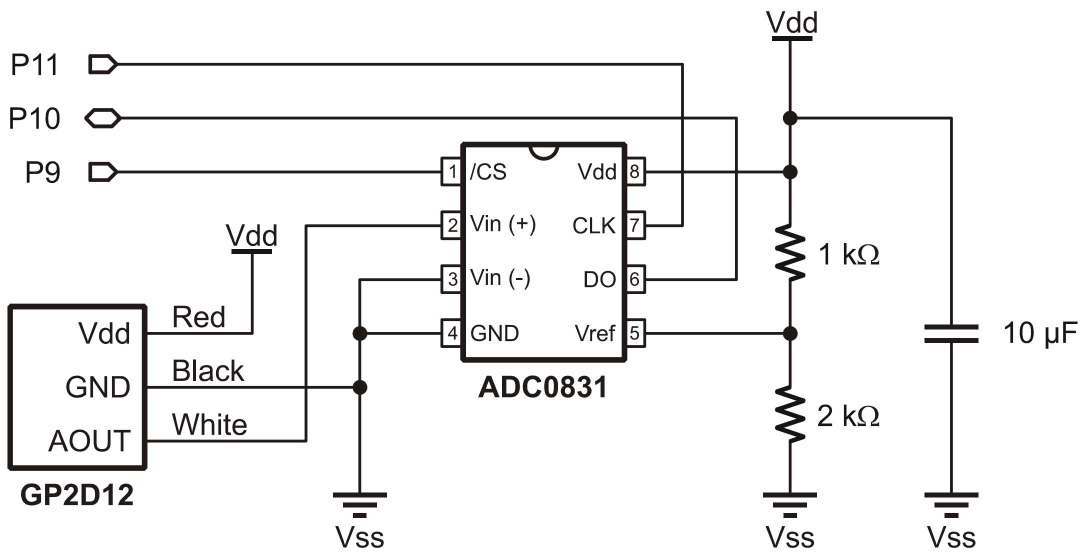
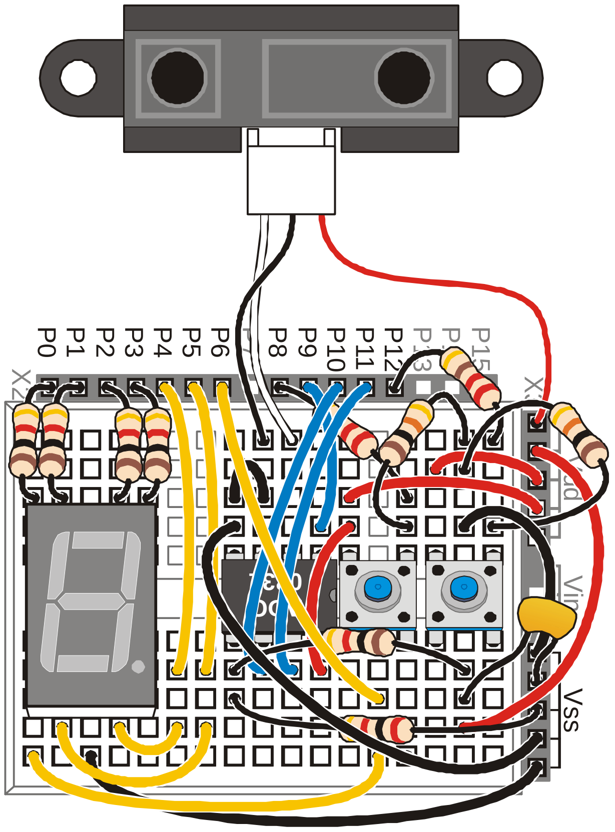


Figure 54



POWER SUPPLY CONNECTIONS

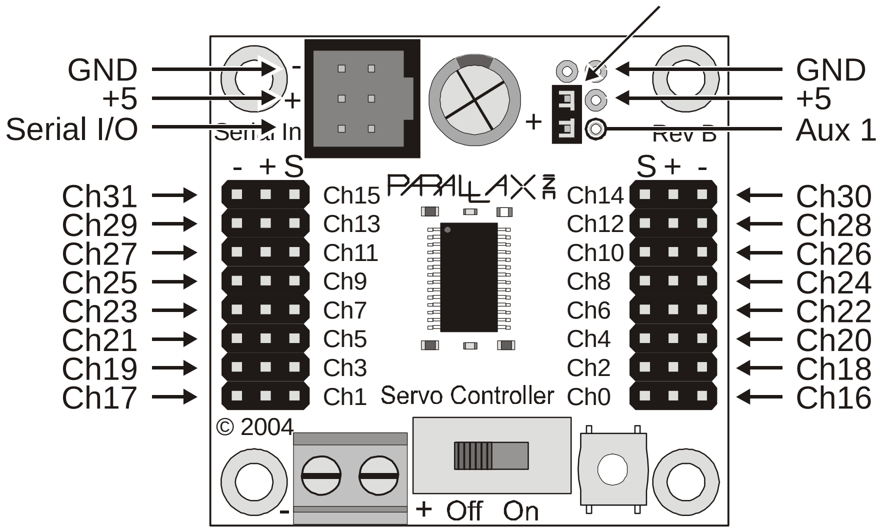
This is the trickiest part of finishing the circuit.

- Connect the Parallax Servo Controller power supply directly to the 7.2V battery. At CrustCrawler, we use a Kyosho battery connector which is available at our web site or can be found at most hobby dealers.

- Connect the Board of Education carrier board to the 9V battery.

NETWORKING ADDITIONAL PARALLAX SERVO CONTROLLERS

One Parallax Servo Controller supports 16 servos. If you add many servos to your QuadCrawler for accessories such as the 3 degree of freedom leg upgrade (3DOF) a robotic arm or the S3 Tilt / Pan system you would simply network another Parallax Servo Controller as shown below.



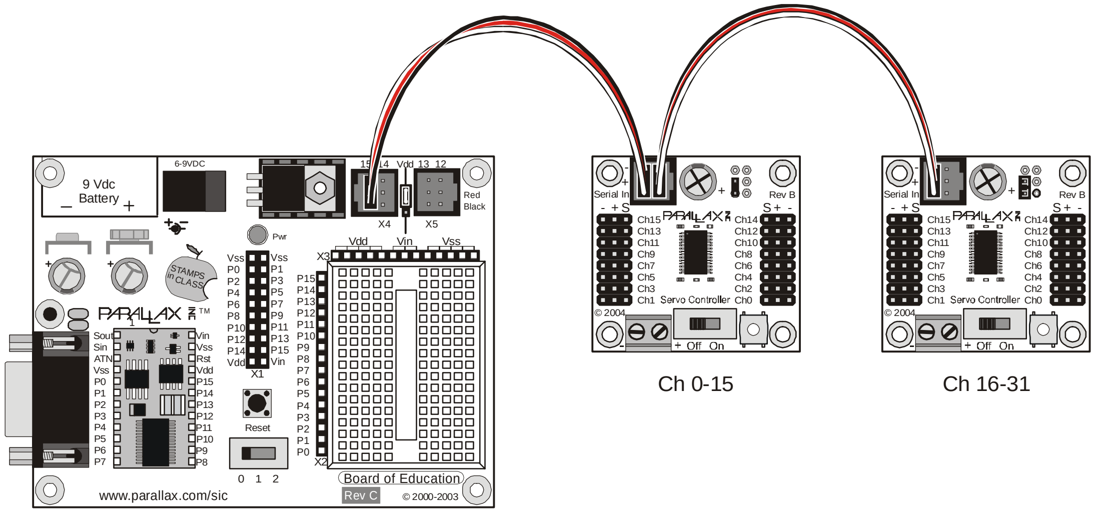


Figure 54

Chapter #5: Programming the QuadCrawler

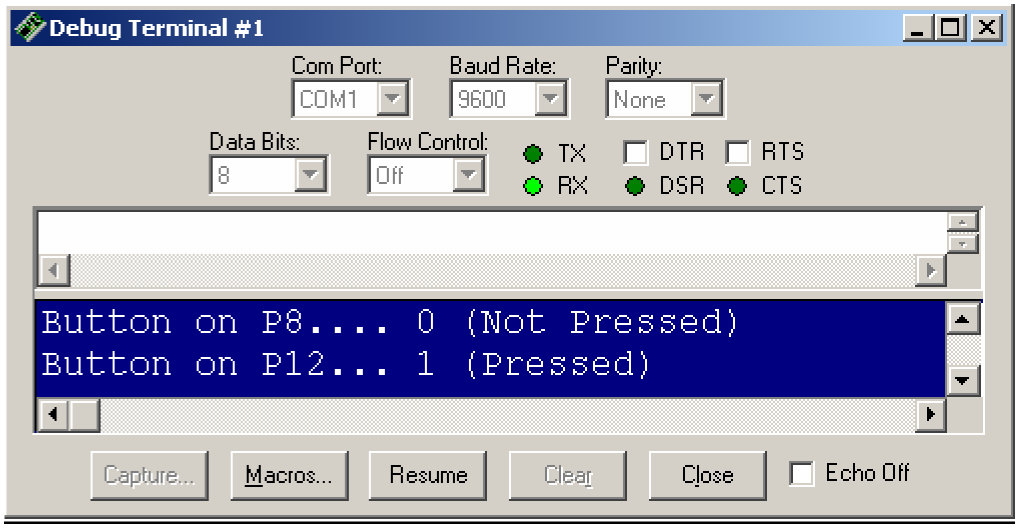
Prior to running the QuadCrawler we will check the individual circuit systems (the pushbuttons, the LED) and then configure the servos for running. All of the programs used in this chapter are available for download from the QuadCrawler page on the CrustCrawler web site. If you need help understanding how to setup the BASIC Stamp Windows editor and how to download programs, see the *What's a Microcontroller?* Version 2.0 text on the Parallax web site.

TESTING THE PUSHBUTTON CONTROLS

Run the following code to check the pushbuttons:

```
' =========================================================================
'
'   File....... QuadCrawler_Pushbutton_Test.BS2
'   Purpose.... Simple checkout of pushbutton circuits
'   Author..... Parallax
'   E-mail..... support@crustcrawler.com
'   Started....
'   Updated.... 11 AUG 2003
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =========================================================================


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

' Uses editor DEBUG screen to check button inputs.


' -----[ Revision History ]------------------------------------------------


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

Button1         PIN     8
Button2         PIN     12


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


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

btnVal          VAR     Bit                     ' state of a button


' -----[ EEPROM Data ]-----------------------------------------------------
' -----[ Initialization ]--------------------------------------------------

Setup:
  DEBUG CLS, "Button Test"


' -----[ Program Code ]----------------------------------------------------

Main:
  DEBUG CRSRXY, 0, 2
  btnVal = Button1                              ' get state of button 1
  DEBUG "Button on P8.... ", BIN1 btnVal        ' display it
  GOSUB Show_State
  btnVal = Button2                              ' get state of button 2
  DEBUG "Button on P12... ", BIN1 btnVal        ' display it
  GOSUB Show_State
  GOTO Main                                     ' do it again


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

Show_State:
  IF (btnVal = 1) THEN
    DEBUG " (Pressed)", CLREOL, CR
  ELSE
    DEBUG " (Not Pressed)", CLREOL, CR
  ENDIF
  RETURN
```

If the circuit is wired correctly you will be able to press either pushbutton and see a "1" appear in the Debug Terminal as shown in Figure 60. If you receive no response from the pushbuttons carefully check the wiring to see that you are connected to the correct BASIC Stamp I/O pins. Once it is working properly proceed to the next section to test the seven-segment LED.



```
Button on P8.... 0 (Not Pressed)
Button on P12... 1 (Pressed)
```

Figure 55: DEBUG Output from Testing Pushbuttons

TESTING THE SEVEN-SEGMENT LED

Run the following code to check the seven-segment LED:

```
' =========================================================================
'
'   File....... QuadCrawler_Seven-Segment_Display_Test.BS2
'   Purpose.... Simple checkout of seven-segment display connections
'   Author..... Parallax
'   E-mail..... support@crustcrawler.com
'   Started....
'   Updated.... 09 AUG 2003
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =========================================================================


' -----[ Program Description ]---------------------------------------------
'
' Displays digits 0 through 9 on seven-segment display to test connections.
'
' Segment map:
'
'       (a)
'      -----
' (f) |     | (b)
'     | (g) |
'      -----
' (e) |     | (c)
'     |     |
'      -----
'       (d)


' -----[ Revision History ]------------------------------------------------


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

Segments        VAR     OUTL                    ' output on pins 0 - 7


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

                        '.edcbafg               ' display segments
Dig0            CON     %01111110               ' segment data for digits
Dig1            CON     %00011000
Dig2            CON     %01101101
Dig3            CON     %00111101
Dig4            CON     %00011011
Dig5            CON     %00110111
Dig6            CON     %01110111
Dig7            CON     %00011100
Dig8            CON     %01111111
Dig9            CON     %00011111
```

```
DigA            CON     %01011111               ' hex A - F (10 - 15)
DigB            CON     %01110011
DigC            CON     %01100110
DigD            CON     %01111001
DigE            CON     %01100111
DigF            CON     %01000111


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

idx             VAR     Nib                     ' digit index


' -----[ EEPROM Data ]-----------------------------------------------------


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

Setup:
  DIRL = %01111111                              ' P0 - P6 are outputs


' -----[ Program Code ]----------------------------------------------------

Main:
  DO
    FOR idx = $0 TO $F                          ' display all digits
      LOOKUP idx, [Dig0, Dig1, Dig2, Dig3,
                   Dig4, Dig5, Dig6, Dig7,
                   Dig8, Dig9, DigA, DigB,
                   DigC, DigD, DigE, DigF], Segments
      PAUSE 500
    NEXT
  LOOP                                          ' loop forever


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

If the circuit is wired correctly the BASIC Stamp 2 module will display digits 0 through E on the seven-segment LED.

USER INTERFACE TEST

Run the following code to check the seven-segment LED with the pushbuttons:

```
' =========================================================================
'
'   File....... QuadCrawler_Interface_Test.BS2
'   Purpose.... Test QuadCrawler buttons and 7-Segment display
'   Author..... Parallax
'   E-mail..... support@crustcrawler.com
'   Started....
'   Updated.... 11 AUG 2003
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =========================================================================


' -----[ Program Description ]---------------------------------------------
'
' Scan QuadCrawler buttons and update 7-segment display.  Button1 increments
' the display, Button2 decrements the display.  This program introduces a
' subroutine for scanning and debouncing both buttons without the use of the
' PBASIC BUTTON command.


' -----[ Revision History ]------------------------------------------------


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

ModeBtn         PIN     8                       ' select robot mode
StartBtn        PIN     12                      ' start/stop robot

Segments        VAR     OUTL                    ' output on pins 0 - 7


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

                        '.edcbafg               ' display segments
Dig0            CON     %01111110               ' segment data for digits
Dig1            CON     %00011000
Dig2            CON     %01101101
Dig3            CON     %00111101
Dig4            CON     %00011011
Dig5            CON     %00110111
Dig6            CON     %01110111
Dig7            CON     %00011100
Dig8            CON     %01111111
Dig9            CON     %00011111

Pressed         CON     1                       ' button states
NotPressed      CON     0


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

```
btns            VAR     Nib                     ' button holder
btn1            VAR     btns.BIT0               ' debounced button value
btn2            VAR     btns.BIT1               ' deboucned button value
idx             VAR     Nib                     ' digit index
counter         VAR     Nib                     ' current digit to display


' -----[ EEPROM Data ]-----------------------------------------------------


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

Setup:
  DIRL = %01111111                              ' P0 - P6 are outputs
  GOSUB Show_Digit                              ' initialize display


' -----[ Program Code ]----------------------------------------------------

Main:
  DEBUG CLS,
        "Button & Display Test", CR, CR,
        "Mode (P8).......... ", CR,
        "Start/Stop (P12)... "

  DO
    GOSUB Get_Buttons                           ' scan buttons
    GOSUB Show_Buttons                          ' show states
    IF (btns > %00) AND (btns < %11) THEN       ' one or the other pressed?
      counter = counter + btn1 // 10            ' increment if Button1 = 1
      counter = counter + (9 * btn2) // 10      ' decrement if Button2 = 1
      GOSUB Show_Digit                          ' update display
      PAUSE 250                                 ' 1/4 sec between changes
    ENDIF
  LOOP                                          ' do forever


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

' Show digit in "counter" on 7-segment display

Show_Digit:
  DEBUG CRSRXY, 0, 5, DEC ?counter              ' update DEBUG screen
  LOOKUP counter, [Dig0, Dig1, Dig2, Dig3, Dig4,
                   Dig5, Dig6, Dig7, Dig8, Dig9], Segments
  RETURN


' Scan and debounce both buttons

Get_Buttons:
  btns = %0011                                  ' assume both pressed
  FOR idx = 1 TO 5
    btns.BIT0 = btns.BIT0 & ModeBtn             ' scan mode button
    btns.BIT1 = btns.BIT1 & StartBtn            ' scan start/stop button
    PAUSE 5                                     ' debounce delay
  NEXT
  RETURN
```

```
' Show current button states

Show_Buttons:
  FOR idx = 0 TO 1
    DEBUG CRSRXY, 20, 2 + idx                   ' move cursor
    IF (btns.LOWBIT(idx) = Pressed) THEN        ' check and display status
      DEBUG "Pressed", CLREOL
    ELSE
      DEBUG "Not Pressed"
    ENDIF
  NEXT
  RETURN
```

TUNING THE QUADCRAWLER'S LEGS

The HomeQuad.bs2 program is a routine designed to assist in centering up the servos to ensure proper walking operation..

CAUTION: Please ensure that your QuadCrawler is mounted on top of a box or elevated platform with its legs clear of your working surface before you run this program.

```
'---- [HomeQuad.bs2 Version 1.0]--------------------------------
'{$STAMP BS2}
'{$PBASIC 2.5}
'
'
'   File....... HomeQuad.BS2
'   Purpose.... Adjust QuadCrawler Legs to Center and Down
'   Author..... CustCrawler Inc. (Mike Gebhard)
'   E-mail..... support@crustcrawler.com
'   Started.... 16 April 2004
'   Updated.... 24 April 2004
'
' Download HomeQuad to adjust your QuadCrawler's legs
' according to the QuadCrawler assembly guide.
'
'     **Right Side**
' F     |        |
' O    _|_      _|_
' R   / 1 \____/ 2 |
' W  |             |
' A  |     ____    |
' R   \_3_/    \_4_|
' D     |        |
'       |        |
'     **Left Side**
'----------------------------------------------------------------

' -----[ I/O Definitions ]-----------------------------------------------
PSC             PIN     15              ' PSC module
Baud            CON     33164           ' 2400 baud

'---- [Walking Variables] ------------------------------------------------
servoAddr       VAR     Byte            ' Servo addresses
Index           VAR     Word            ' Gait select
servoPosition   VAR     Word            ' Servo Position
newMode         VAR     Word            ' Current mode
Ramp            VAR     Byte            ' Ramp used in SEROUT

'----- [Adjustable Ramp Values] -----------------------------------------------
LiftRamp        CON     $1     ' Very fast ramp for lifting leg
Fast            CON     $F

'               Raise        Center       Lower
Adjust  DATA $01,$26,$01, $00,$EE,$02, $01,$B0,$04 'Leg1
        DATA $03,$26,$01, $02,$EE,$02, $03,$B0,$04 'Leg2
        DATA $05,$B0,$04, $04,$EE,$02, $05,$26,$01 'Leg3
        DATA $07,$B0,$04, $06,$EE,$02, $07,$26,$01,'Leg4
        $FF ' end of adjust legs


'---- [Center and lower all legs] -----------------
Leg_Adjust:
  DEBUG CLS, "Center and Lower Legs",CR,
           "====================",CR,CR
  READ Adjust, servoAddr
  'DEBUG HEX ?Index, HEX ?servoAddr, HEX ?newMode
  DO WHILE ServoAddr <> $FF
    IF (ServoAddr // 2) = 1 THEN
      Ramp = LiftRamp
    ELSE
      Ramp = Fast
    ENDIF
    GOSUB Send_EEPROM_Info_to_PSC
    PAUSE 50
    'DEBUG HEX ?servoAddr
  LOOP
  DEBUG "Ready to Adjust Legs!!!",CR
END

'----- [Serial Out EEPROM values to PSC] -------------------------------
Send_EEPROM_Info_to_PSC:
  ' Read EEPROM data
  READ Index, servoAddr,servoPosition.LOWBYTE,servoPosition.HIGHBYTE
  ' Send EEPROM data to PSC
  SEROUT PSC,33164,["!SC",ServoAddr, Ramp,
                         servoPosition.LOWBYTE, servoPosition.HIGHBYTE, CR]
  'DEBUG HEX ?servoAddr, HEX ?servoPosition.LOWBYTE, HEX ?servoPosition.HIGHBYTE
  Index = Index + 3
  READ Index,servoAddr
  PAUSE 500
RETURN
```

ADJUSTING THE QUADCRAWLER'S LEGS

Not all servos are created equal. Therefore, you might have to fine-tune the legs. After the program executes, it leaves the robot in the home position: centered horizontally and standing. When looking down at the legs they should appear perpendicular to the body. Don't worry if you can't get all the legs perfectly perpendicular. Being off a degree or two will not adversely affect the robot's gait. Refer to the horizontal and vertical adjustment information on the next two pages to adjust the legs.

Horizontal Servo Adjustment

1. Loosen (do not remove) the side support screws on the same side as the leg as well as the upper front support screws. (see Figure 56)

2. Remove the top horizontal servo screw.

3. Gently lift the lip of the top half of the servo body to pop the servo spindle out of its holder and rotate it so that the leg is as close to straight out from the body as possible.

4. Re-install the top horizontal servo screw.

5. Perform steps 1 – 4 for each leg that is not centered.

6. Re-tighten the side and front support screws.

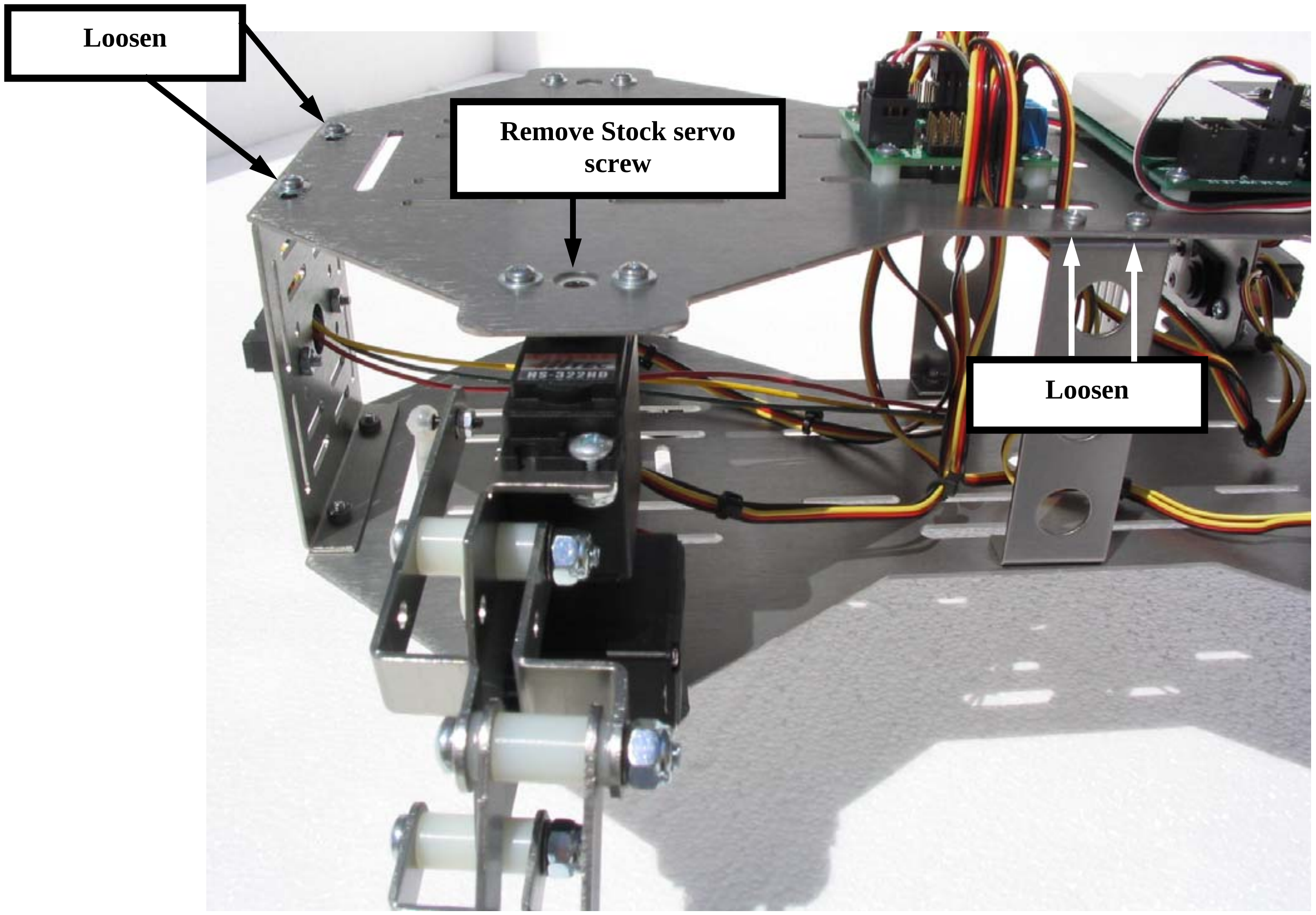


Figure 56: Horizontal Leg Adjustment

Vertical Servo Adjustment

1. Loosen the servo screw on the servo control horn and remove the servo control horn. (Figure 57)
2. Turn the servo control horn until it is aligned as close to 90 degrees vertical as possible.
3. Re-install the servo screw
4. Repeat steps 1- 3 for every leg that needs adjustment.

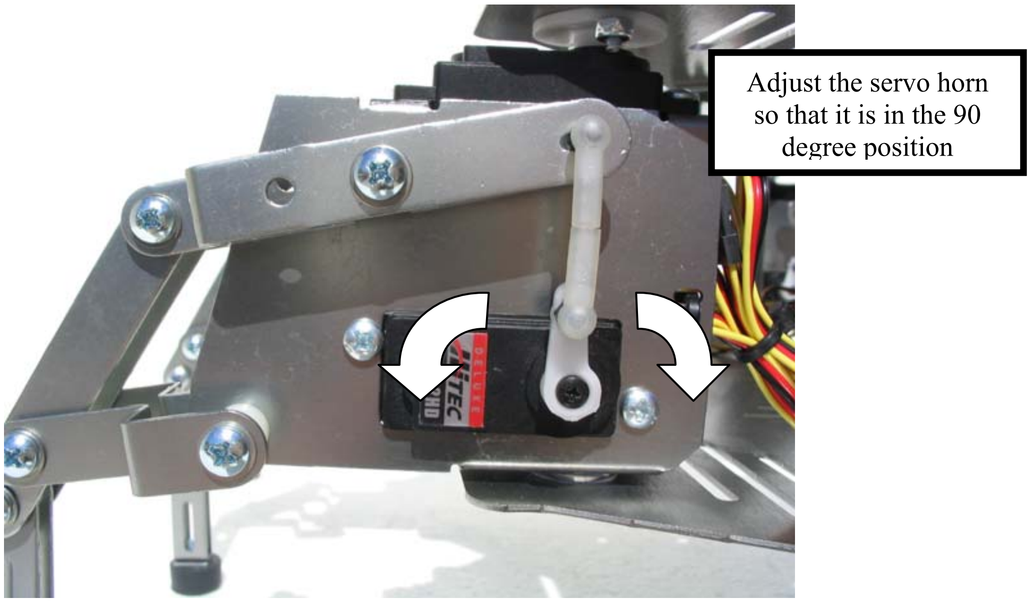

Adjust the servo horn so that it is in the 90 degree position

Figure 57

The next procedure will ensure that all of the legs are evenly sharing the weight of the QuadCrawler

Once all of the servo arms have been adjusted as vertical as possible, place your QuadCrawler on a smooth surface in a standing position. One leg at a time swing the legs forward and backwards and note any differences in resistance between the leg and the floor.

For legs that appear to be bearing more of the weight than the other legs on the QuadCrawler - Remove the dog bones from the leg and twist the dog bones closer together. Re-install the dog bone and re-test.

For leg that appear to be bearing less of the weight than the other leg of the QuadCrawler - Remove the dog bones from the leg and twist the dog bones further apart - Re-install the dog bone and re-test.

Once all of the legs have been adjusted, you are ready to load and run the walking programs on the next pages. Remember, all of the programs included in this manual as well as new programs that we develop are always available as a free download from our web site at www.CrustCrawler.com.

QUADCRAWLER WALKING PROGRAM – BUTTON CODE

```
'------- [QuadWalker_Table_PSC_Button Version 1.0] -----------------------------------------
-----------
'{$STAMP BS2}
'{$PBASIC 2.5}
'
'   File....... QuadWalker_Table_PSC_Button.BS2
'   Purpose.... Use EEPROM tables and a button interface to
'               produce 15 different gaits
'   Author..... CustCrawler Inc. (Mike Gebhard)
'   E-mail..... support@crustcrawler.com
'   Started.... 16 April 2004
'   Updated.... 24 April 2004
'
' QuadWalker_Table_PSC_Button.bs2 contains 15 different gait settings.
'
' Button operation:
' Press both buttons during program execution to enter selection mode.
' Press both buttons to accept selection
' Press the up and down buttons to select gaits.
'
' If Adjust Legs is selected (0) the robot will center and lower it's legs
' for adjustment according to the quadCrawler assembly guide.
' Press the BOE reset Button to restart.
'
' Code Sections:
' The Button Code Sections can be replaced with your own code for use
' with devices like RC units or various sensors.
' Sub routines include:
'  -> Update_Button_Index
'  -> Show_Buttons - GUI this routine updates the debug screen
'  -> Get_Buttons
'
' Use the selectedMode variable to pass gait information
' to the Select_Mode sub routine.  Values list below:
'
' LED Display
' (0)   $00 - Home
' (1)   $01 - Spin Left
' (2)   $02 - Spin Right
'
' Display                         Display
' (3)   $10 - Forward Fast         (6)   $20 - Forward
' (4)   $11 - Fast Forward Left    (7)   $21 - Forward Left
' (5)   $12 - Fast Forward Right   (8)   $22 - Forward Right
'
' Display                         Display
' (9)   $20 - Backward             (C)   $40 - Fast Backward
' (A)   $21 - Backward Left        (d)   $41 - Fast Backward Left
' (b)   $22 - Backward Right       (E)   $42 - Fast Backward Right
'                                  (F)   Open
'
' Adjustable values:
' [Adjustable Ramp Value section]
' Ramp is a prameter passed to the PSC with the SEROUT command.
' See your PSC instruction manual for more information.
' Ramp is the speed/time it takes for a servo
' to move to a new position.  Small values, 1-7,
' move fast while larger values, 8-15($F), move
' slow.
'
' LiftRamp        CON     $A  -> Used to lift legs
' VeryFast        CON     $7  -> Used in walk fast
' Fast            CON     $A  -> Used in turn
' Medium          CON     $D  -> Used in walk slow
' Slow            CON     $F  -> Used in gradual turning
'
' Adjust the these values to increase/decrease
' servo speeds.  Experiment to find the values
' that work best for your robot.
'
' [Adjustable horizontal servo positions]
' Adjust the constants below to modify horizontal servo positions.
'
' Pos1            CON     650 -> Right back/Left forawrd
' Pos2            CON     750 -> Center servo
' Pos3            CON     850 -> Right forward/Left back
'
' All servos move from Pos1 or Pos3 to Pos2 or from
' Pos2 to Pos1 or Pos3 depending on the the mode selected.
'
' Walk Forward Mode Example:
'
'      **Right Side**
'      Pos2      Pos2
'      750       750
' F     |        |          Next Move:
' O    _|_      _|_         Leg1 lifts and moves to Pos3(850)
' R   / 1 \____/ 2 |        Leg2 moves to Pos1(650)
' W  |             |        Leg3 moves to Pos2(750)
' A  |     ____    |        Leg4 lifts and moves to Pos2(750)
' R   \_3_/    \_4_|
' D     /        \
'      /          \
'     650         850
'     Pos1        Pos3
'==========================
'     Pos3        Pos1
'     850         650
'      \          /         Next Move:
'      _\_      _/_         Leg1 moves to Pos2(750)
'     / 1 \____/ 2 |        Les2 lifts and moves to Pos2(750)
'    |             |        Leg3 lifts and moves to Pos1(650)
'    |     ____    |        Leg4 moves to Pos3(850)
'     \_3_/    \_4_|
'       |        |
'       |        |
'      750      750
'      Pos2     Pos2
'      **Left Side**
'
' Example of changing horizontal servo positions:
' Increase stride:    Decrease Stride:
' Pos1 = 600          Pos1 = 700
' Pos2 = 750          Pos2 = 750
' Pos3 = 900          Pos3 = 800
```

```
'
' Becareful changing Pos2 though.  Pos2 is servo center (750).
' Adding to Pos2 (i.e. 750+50=800) will cause the right side to
' adjust forward while the left side adjusts further back.
'
'
' 7 Segement LED Display:
'    Segment map:      .edc bafg  HEX      .edc bafg  HEX
'       (a)         0 %0111 1110  $7E   8 %0111 1111  $7F
'      -----        1 %0001 1000  $18   9 %0001 1111  $1F
' (f) |     | (b)   2 %0110 1101  $6D   A %0101 1111  $5F
'     | (g) |       3 %0011 1101  $3D   B %0111 0011  $73
'      -----        4 %0001 1011  $1B   C %0110 0110  $66
' (e) |     | (c)   5 %0011 0111  $37   D %0111 1001  $79
'     |     |       6 %0111 0111  $77   E %0110 0111  $67
'      -----        7 %0001 1100  $1C   F %0100 0111  $47 - Open
'
'
' EEPROM Sections:
' This program uses EEPROM to store servo address
' servo positions, gait selections, and 7 segment LED Information.
' EEPROM address are Read by the
' Send_EEPROM_Info_to_PSC and Walker sub routines.
' The result is a walking gait.
' Modify this section at your own
' risk.  You can find more information in the Stamp Editor's
' Help under the DATA or READ commands.
' The Forward DATA section is commented to help understand
' what's happening.
'
' Programing note:
' Hex and decimal values used in horizontal motion
' Low Byte to High Byte as stored in EEPROM
' Pos1 = 650 = $8A,$02,
' Pos2 = 750 = $EE,$02,
' Pos3 = 850 = $52,$03,
'
' Vertical positions
' $B0,$04, = Lower Right - Raise Left
' $26,$01, = Raise Right - Lower Left
'
'
'-------------------------------------------------------------------------------------

' -----[ I/O Definitions ]-----------------------------------------------
PSC             PIN     15              ' PSC module
ModeBtn         PIN     8               ' select robot mode up
StartBtn        PIN     12              ' select robot mode down
Segments        VAR     OUTL            ' output on pins 0 - 7
Baud            CON     33164           ' 2400 baud

'---- [Button Variables]-------------------------------------------------
btns            VAR     Nib             ' button holder
btn1            VAR     btns.BIT0       ' debounced button value
btn2            VAR     btns.BIT1       ' deboucned button value
idx             VAR     Nib             ' digit index
btnIndx         VAR     Nib             ' current digit to display
flag            VAR     Bit
```

```
'---- [button states] -----------------------------------------------------
Pressed         CON     1
NotPressed      CON     0


'---- [Walking Variables] -------------------------------------------------
servoAddr       VAR     Byte            ' Servo addresses
Index           VAR     Word            ' Gait select
servoPosition   VAR     Word            ' Servo Position
selectedMode    VAR     Byte            ' Mode selected
newMode         VAR     Word            ' Current mode
Ramp            VAR     Byte             ' Ramp used in SEROUT
RightRamp       VAR     Byte             ' Right side ramp values
LeftRamp        VAR     Byte             ' Left side ramp values

'----- [Adjustable Ramp Values] ---------------------------------------------
' $1(1) = Very Fast
' $F(15) = Very Slow
LiftRamp        CON     $6      ' Very fast ramp for lifting leg
VeryFast        CON     $7
Fast            CON     $A
Medium          CON     $B
'Slow            CON     $A

'----- [Adjustable horizontal servo positions] ------------------------------
' Adjust the values below to change leg
' positions.  All legs move from Pos1 or Pos3
' to Pos2.  You might find it useful to adjust these values
' depending on loads added to your QuadCrawler.
Pos1            CON     650   ' Right back/Left forawrd
Pos2            CON     750   ' Center
Pos3            CON     850   ' Right forward/Left back




'---- [7 Segment LED and Gait EEPROM Data] -----
' Hex #    0   1   2   3   4   5   6   7   8   9   A   B   C   D   E    F
LED  DATA $7E,$18,$6D,$3D,$1B,$37,$77,$1C,$7F,$1F,$5F,$73,$66,$79,$67',$47
Gait DATA $00,$01,$02,$10,$11,$12,$20,$21,$22,$30,$31,$32,$40,$41,$42

'---- [Servo Address and Position EEPROM data] -----------------
' Walk Forward
Forward DATA $01,$26,$01, $07,$B0,$04, 'Lift legs 1 and 4
             'Move legs 1,2,3,&4
             $00,Word Pos3, $02,Word Pos1, $04,Word Pos2, $06,Word Pos2,
             $01,$B0,$04, $07,$26,$01, 'Lower legs 1 and 4
             $03,$26,$01, $05,$B0,$04, 'Lift legs 2 and 3
             'Move legs 1,2,3,&4
             $00,Word Pos2, $02,Word Pos2, $04,Word Pos1, $06,Word Pos3,
             $03,$B0,$04, $05,$26,$01,  'Lower legs 2 and 3
             $FF ' end of forward

' Walk Backward
Back    DATA $01,$26,$01, $07,$B0,$04,
             $00,Word Pos2, $02,Word Pos2, $04,Word Pos1, $06,Word Pos3,
             $01,$B0,$04, $07,$26,$01,
             $03,$26,$01, $05,$B0,$04,
             $00,Word Pos3, $02,Word Pos1, $04,Word Pos2, $06,Word Pos2,
             $03,$B0,$04, $05,$26,$01,
             $FF 'end of back

' Left Turn EEPROM values
LTurn   DATA $01,$26,$01, $07,$B0,$04,
             $00,Word Pos3, $06,Word Pos3, $04,Word Pos2, $02,Word Pos2,
             $01,$B0,$04, $07,$26,$01,
             $03,$26,$01, $05,$B0,$04,
             $00,Word Pos2, $06,Word Pos2, $04,Word Pos3, $02,Word Pos3,
             $03,$B0,$04, $05,$26,$01,
             $FF ' end of left turn

' Right turn EEPROM Values
RTurn   DATA $01,$26,$01, $07,$B0,$04,
             $00,Word Pos2, $06,Word Pos2, $04,Word Pos3, $02,Word Pos3,
             $01,$B0,$04, $07,$26,$01,
             $03,$26,$01, $05,$B0,$04,
             $00,Word Pos3, $06,Word Pos3, $04,Word Pos2, $02,Word Pos2,
             $03,$B0,$04, $05,$26,$01,
             $FF  ' end of right turn

'               Raise        Center       Lower
Adjust  DATA $01,$26,$01, $00,$EE,$02, $01,$B0,$04 'Leg1
        DATA $03,$26,$01, $02,$EE,$02, $03,$B0,$04 'Leg2
        DATA $05,$B0,$04, $04,$EE,$02, $05,$26,$01 'Leg3
        DATA $07,$B0,$04, $06,$EE,$02, $07,$26,$01,'Leg4
        $FF ' end of adjust legs
'---- [End EEPROM Data] -----------------

'---- [Troubleshoot section] ---------------
' Remove comments to run related
' walking gaits without first entering
' button interface.
'Index = Adjust
'DEBUG HEX? Adjust, HEX ?Index
'END
'Index = Forward
'Index = Back
'Index = LTurn
'Index = RTurn

'RightRamp = $7
'LeftRamp = $7

'GOTO Walker
'GOTO Leg_Adjust

'---- [setup] -----------------
DIRL = %01111111              ' P0 - P6 are outputs
' Setup debug screen interface
Main:
  DEBUG CLS,
        "Button, Display, and Mode Settings", CR, CR,
        "Up Button (P8).... ", CR,
        "Down Button (P12). ", CR,CR,CR,
        "Selected Mode.....",CR,
        "LED Display.......",CR
  GOSUB Show_Buttons
  READ btnIndx, Segments                      ' display current gait
  GOSUB Update_Button_Index

'---- [Gait Selection] -------------------------------------------------
Select_Mode:
  ' Turnning gaits
  SELECT selectedMode
    CASE $00
      newMode = Adjust
      GOTO Leg_Adjust
    CASE $01
      newMode = LTurn
      RightRamp = Fast
      LeftRamp = Fast
      GOTO Walker
    CASE $02
      newMode = RTurn
      RightRamp = Fast
      LeftRamp = Fast
      GOTO Walker
  ENDSELECT

' Check HIGHNIB to select
' EEPROM starting address
  IF selectedMode.HIGHNIB <= $2 THEN
    newMode = Forward
  ELSE
    newMode = Back
  ENDIF

' Check HIGHNIB to select
' Ramp speed
  IF (selectedMode.HIGHNIB = $1) OR (selectedMode.HIGHNIB = $4) THEN
    Ramp = VeryFast
  ELSE
    Ramp = Medium
  ENDIF

' Check LowNib to determine
' left and right leg ramps
' 0 = straight
' 1 = slow left side
' 2 = slow right side
  SELECT selectedMode.LOWNIB
    CASE $0
      'DEBUG "Straight ", CR
      RightRamp = Ramp
      LeftRamp = RightRamp
    CASE $1
      'DEBUG "left ", CR
      RightRamp = Ramp
      LeftRamp = (Ramp-4)
    CASE $2
      'DEBUG "Right ", CR
      LeftRamp = Ramp
      RightRamp = (Ramp-4)
  ENDSELECT
  'DEBUG HEX ?RightRamp, HEX ?LeftRamp

DEBUG CLS, "Executing Walker",CR,CR, HEX ?selectedMode
```

5

```
'-----[ Main Walking Routine ]-----------------------------------------
Walker:
' Get EEPROM Address
Index = newMode
' Read EEPROM value into servoAddr
READ Index,ServoAddr
'DEBUG "Entered Walker",CR
' Loop while servoAddr is not equal to $FF
' the end of Forawrd EEPROM data
DO WHILE servoAddr <> $FF
'DEBUG "Looping",CR
' If servoAddr is odd lift the leg fast
' If even assign left and right ramp
  IF (servoAddr // 2) = 1 THEN
    Ramp = LiftRamp
  ELSE
    IF (servoAddr = $00) OR (servoAddr = $02) THEN
      Ramp = RightRamp
    ENDIF
    IF(servoAddr = $04) OR (servoAddr = $06) THEN
      Ramp = LeftRamp
    ENDIF
  ENDIF
  GOSUB Send_EEPROM_Info_to_PSC
  'PAUSE 50
LOOP
  GOSUB Get_Buttons
  IF (btns.BIT0=Pressed) AND (btns.BIT1=Pressed) THEN
    DEBUG "Reset",CR
    PAUSE 500
    GOTO Main
  ELSE
    'DEBUG "Goto Walker",CR
    GOTO Walker
  ENDIF

'---- [Center and lower all legs] -----------------
Leg_Adjust:
DEBUG CLS, "Center and Lower Legs",CR,
           "=====================",CR,CR
Index = newMode
READ Index, servoAddr
'DEBUG HEX ?Index, HEX ?servoAddr, HEX ?newMode
DO WHILE ServoAddr <> $FF
  IF (ServoAddr // 2) = 1 THEN
    Ramp = LiftRamp
  ELSE
    Ramp = Fast
  ENDIF
  GOSUB Send_EEPROM_Info_to_PSC
  PAUSE 50
  'DEBUG HEX ?servoAddr
LOOP
DEBUG "Ready to Adjust Legs!!!",CR,
      "Press the reset button to exit leg adjustment mode."
END

'----- [Serial Out EEPROM values to PSC] -------------------------------
```

```
Send_EEPROM_Info_to_PSC:
  ' Read EEPROM data
  READ Index, servoAddr,servoPosition.LOWBYTE,servoPosition.HIGHBYTE
  ' Send EEPROM data to PSC
  SEROUT PSC,33164,["!SC",ServoAddr, Ramp,
                       servoPosition.LOWBYTE, servoPosition.HIGHBYTE, CR]
  'DEBUG HEX ?servoAddr, HEX ?servoPosition.LOWBYTE, HEX ?servoPosition.HIGHBYTE
  Index = Index + 3
  READ Index,servoAddr
  'PAUSE 2
RETURN

'---- [Button Code Section] ------
Update_Button_Index:
'DEBUG "Select Gait",CR
PAUSE 10
    DEBUG CRSRXY, 0, 10,"Gait selection = Up/Down Buttons",CR,
                      "Enter/New Gait = Both"
  DO
    GOSUB Get_Buttons                          ' scan buttons
    GOSUB Show_Buttons                         ' Update debug screen
    IF (btns > %00) AND (btns < %11) THEN      ' one or the other pressed?
      btnIndx = btnIndx + btn1 // 15           ' increment if Button1 = 1
      btnIndx = btnIndx + (14 * btn2) // 15    ' decrement if Button2 = 1
      ' Read EEPROM
      READ btnIndx, Segments                   ' Update LED display
      READ gait+btnIndx,selectedMode           ' Update Selected gait Mode
      PAUSE 250                                ' 1/4 sec between changes
      GOSUB Show_Buttons
    ENDIF
    GOSUB Get_Buttons
    ' Change gaits if both buttons are pressed
    IF (btns.BIT0=Pressed) AND (btns.BIT1=Pressed) THEN
      GOSUB Show_Buttons
      PAUSE 1000
      GOTO Select_Mode
    ENDIF
  LOOP

'---- [Display Button and gait information to Debug Screen] -----
Show_Buttons:
  FOR idx = 0 TO 1
    DEBUG CRSRXY, 18, 2 + idx                  ' move cursor
    IF (btns.LOWBIT(idx) = Pressed) THEN       ' check and display status
      DEBUG "Pressed", CLREOL
    ELSE
      DEBUG "Not Pressed"
    ENDIF
  NEXT
    DEBUG CRSRXY, 18, 6, HEX selectedMode.HIGHNIB, HEX selectedMode.LOWNIB
    DEBUG CRSRXY, 18, 7, HEX btnIndx
  RETURN

' ----- [Read Buttons] -----------------------------
Get_Buttons:
  btns = %0011                                 ' assume both pressed
  FOR idx = 1 TO 5
    btns.BIT0 = btns.BIT0 & ModeBtn            ' scan mode button
    btns.BIT1 = btns.BIT1 & StartBtn           ' scan start/stop button
    PAUSE 5                                     ' debounce delay
  NEXT
RETURN
```

QUADCRAWLER WALKING PROGRAM – RADIO CONTROL (R/C) CODE

```
'------- [QuadWalker_Table_PSC_RC Version 1.0] ----------------------------------------------
-------
'{$STAMP BS2}
'{$PBASIC 2.5}
'
'   File....... QuadWalker_Table_PSC_Button.BS2
'   Purpose.... Use EEPROM tables and R/C control
'               produce 15 different gaits (we use a Tower Hobbies
'   "System 3000" radio with matched receiver to control the QuadCrawler)
'   Author..... CustCrawler Inc. (Mike Gebhard)
'   E-mail..... support@crustcrawler.com
'   Started.... 16 April 2004
'   Updated.... 24 April 2004
'
' QuadWalker_Table_PSC_RC.bs2 contains 15 different gait settings
' controlled by an RC Tx/Rx.
'
' RC Operation:
' Left Stick positions
'                            Up
'           ______________________________________
'          |            |            |            |
' Fast -> |   Left     |  Forward   |  Right     | <- Top
'          |------------|------------|------------|
' Slow -> |   Left     |  Forward   |  Right     | <- Top Mid
'          |------------|------------|------------|
' Spin -> |   Left     |  Stop      |  Right     | <- Center
'          |------------|------------|------------|
' Slow -> |   Left     |  Backward  |  Right     | <- Bottom Mid
'          |------------|------------|------------|
' Fast -> |   Left     |  Backward  |  Right     | <- Bottom
'          |____________|____________|____________|
'                           Down
'
' Use the selectedMode variable to pass gait information
' to the Select_Mode sub routine.  Values list below:
'
' LED Display
' (0)   $00 - Home
' (1)   $01 - Spin Left
' (2)   $02 - Spin Right
'
' Display                         Display
' (3)   $10 - Forward Fast         (6)   $20 - Forward
' (4)   $11 - Fast Forward Left    (7)   $21 - Forward Left
' (5)   $12 - Fast Forward Right   (8)   $22 - Forward Right
'
```

```
' Display                          Display
' (9)   $20 - Backward             (C)   $40 - Fast Backward
' (A)   $21 - Backward Left        (d)   $41 - Fast Backward Left
' (b)   $22 - Backward Right       (E)   $42 - Fast Backward Right
'                                  (F)   Open
'
' Adjustable values:
' [Adjustable Ramp Value section]
' Ramp is a prameter passed to the PSC with the SEROUT command.
' See your PSC instruction manual for more information.
' Ramp is the speed/time it takes for a servo
' to move to a new position.  Small values, 1-7,
' move fast while larger values, 8-15($F), move
' slow.
'
' LiftRamp        CON     $1  -> Used to lift legs
' VeryFast        CON     $7  -> Used in walk fast
' Fast            CON     $A  -> Used in turn
' Medium          CON     $D  -> Used in walk slow
' Slow            CON     $F  -> Used in gradual turning
'
' Adjust the these values to increase/decrease
' servo speeds.  Experiment to find the values
' that work best for your robot.
'
' [Adjustable horizontal servo positions]
' Adjust the constants below to modify horizontal servo positions.
'
' Pos1            CON     650 -> Right back/Left forawrd
' Pos2            CON     750 -> Center servo
' Pos3            CON     850 -> Right forward/Left back
'
' All servos move from Pos1 or Pos3 to Pos2 or from
' Pos2 to Pos1 or Pos3 depending on the the mode selected.
'
' Walk Forward Mode Example:
'
'      **Right Side**
'      Pos2      Pos2
'      750       750
' F     |        |          Next Move:
' O    _|_      _|_         Leg1 lifts and moves to Pos3(850)
' R   / 1 \____/ 2 |        Leg2 moves to Pos1(650)
' W  |             |        Leg3 moves to Pos2(750)
' A  |    ____     |        Leg4 lifts and moves to Pos2(750)
' R   \_3_/    \_4_|
' D     /        \
'      /          \
'     650         850
'     Pos1        Pos3
'==========================
'     Pos3        Pos1
'     850         650
'      \          /         Next Move:
'      _\_      _/_         Leg1 moves to Pos2(750)
'     / 1 \____/ 2 |        Les2 lifts and moves to Pos2(750)
'    |             |        Leg3 lifts and moves to Pos1(650)
'    |    ____     |        Leg4 moves to Pos3(850)
'     \_3_/    \_4_|
'       |        |
'       |        |
'      750      750
'      Pos2     Pos2
'      **Left Side**
'
' Example of changing horizontal servo positions:
' Increase stride:    Decrease Stride:
' Pos1 = 600          Pos1 = 700
' Pos2 = 750          Pos2 = 750
' Pos3 = 900          Pos3 = 800
'
' Becareful changing Pos2 though.  Pos2 is servo center (750).
' Adding to Pos2 (i.e. 750+50=800) will cause the right side to
' adjust forward while the left side adjusts further back.
'
'
' 7 Segement LED Display:
'    Segment map:      .edc bafg  HEX      .edc bafg  HEX
'       (a)         0 %0111 1110  $7E   8 %0111 1111  $7F
'      -----        1 %0001 1000  $18   9 %0001 1111  $1F
' (f) |     | (b)   2 %0110 1101  $6D   A %0101 1111  $5F
'     | (g) |       3 %0011 1101  $3D   B %0111 0011  $73
'      -----        4 %0001 1011  $1B   C %0110 0110  $66
' (e) |     | (c)   5 %0011 0111  $37   D %0111 1001  $79
'     |     |       6 %0111 0111  $77   E %0110 0111  $67
'      -----        7 %0001 1100  $1C   F %0100 0111  $47 - Open
'
'
' EEPROM Sections:
' This program uses EEPROM to store servo address
' servo positions, gait selections, and 7 segment LED Information.
' EEPROM address are Read by the
' Send_EEPROM_Info_to_PSC and Walker sub routines.
' The result is a walking gait.
' Modify this section at your own
' risk.  You can find more information in the Stamp Editor's
' Help under the DATA or READ commands.
' The Forward DATA section is commented to help understand
' what's happening.
'
' Programming note:
' Hex and decimal values used in horizontal motion
' Low Byte to High Byte as stored in EEPROM
' Pos1 = 650 = $8A,$02,
' Pos2 = 750 = $EE,$02,
' Pos3 = 850 = $52,$03,
'
' Vertical positions
' $B0,$04, = Lower Right - Raise Left
' $26,$01, = Raise Right - Lower Left
'
'
'-----------------------------------------------------------------------------------------

' -----[ I/O Definitions ]---------------------------------------------------
PSC             PIN     15              ' PSC module
StickXPin       PIN     13              ' Left/Right(X) joystick Rx Ch4
StickYPin       PIN     12              ' Up/Down(Y) joystick Rx Ch3
```

```
Segments        VAR     OUTL            ' output on pins 0 - 7
Baud            CON     33164           ' 2400 baud


'---- [Walking Variables] ------------------------------------------------
servoAddr       VAR     Byte            ' Servo addresses
stickXPos       VAR     Word            ' Left/Right (X) joystick posn
stickYPos       VAR     Word            ' Up/Down(Y) joystick position
Index           VAR     Word            ' Gait select
servoPosition   VAR     Word            ' Servo Position
newMode         VAR     Word
selectedMode    VAR     Byte            ' Mode selected
currentMode     VAR     Word            ' Current mode
Ramp            VAR     Byte            ' Ramp used in SEROUT
RightRamp       VAR     Byte            ' Right side ramp values
LeftRamp        VAR     Byte            ' Left side ramp values
Counter         VAR     Nib
GaitValue       VAR     Byte

'----- [Adjustable Ramp Values] ------------------------------------------------
' $1(1) = Very Fast
' $F(15) = Very Slow
LiftRamp        CON     $6     ' fast ramp for lifting leg
VeryFast        CON     $6
Fast            CON     $8

'----- [Adjustable horizontal servo positions] --------------------------------
' Adjust the values below to change leg
' positions.  All legs move from Pos1 or Pos3
' to Pos2.  You might find it useful to adjust these values
' depending on loads added to your QuadCrawler.
Pos1            CON     650   ' Right back/Left forawrd
Pos2            CON     750   ' Center
Pos3            CON     850   ' Right forward/Left back


'---- [7 Segment LED and Gait EEPROM Data] -----
' Hex #    0   1   2   3   4   5   6   7   8   9   A   B   C   D   E    F
LED  DATA $7E,$18,$6D,$3D,$1B,$37,$77,$1C,$7F,$1F,$5F,$73,$66,$79,$67',$47
Gait DATA $00,$01,$02,$10,$11,$12,$20,$21,$22,$30,$31,$32,$40,$41,$42

'---- [Servo Address and Position EEPROM data] -----------------
' Walk Forward
Forward DATA $01,$26,$01, $07,$B0,$04, 'Lift legs 1 and 4
             'Move legs 1,2,3,&4
             $00,Word Pos3, $02,Word Pos1, $04,Word Pos2, $06,Word Pos2,
             $01,$B0,$04, $07,$26,$01, 'Lower legs 1 and 4
             $03,$26,$01, $05,$B0,$04, 'Lift legs 2 and 3
             'Move legs 1,2,3,&4
             $00,Word Pos2, $02,Word Pos2, $04,Word Pos1, $06,Word Pos3,
             $03,$B0,$04, $05,$26,$01,  'Lower legs 2 and 3
             $FF ' end of forward

' Walk Backward
Back    DATA $01,$26,$01, $07,$B0,$04,
             $00,Word Pos2, $02,Word Pos2, $04,Word Pos1, $06,Word Pos3,
             $01,$B0,$04, $07,$26,$01,
             $03,$26,$01, $05,$B0,$04,
             $00,Word Pos3, $02,Word Pos1, $04,Word Pos2, $06,Word Pos2,
             $03,$B0,$04, $05,$26,$01,
             $FF 'end of back

' Left Turn EEPROM values
LTurn   DATA $01,$26,$01, $07,$B0,$04,
             $00,Word Pos3, $06,Word Pos3, $04,Word Pos2, $02,Word Pos2,
             $01,$B0,$04, $07,$26,$01,
             $03,$26,$01, $05,$B0,$04,
             $00,Word Pos2, $06,Word Pos2, $04,Word Pos3, $02,Word Pos3,
             $03,$B0,$04, $05,$26,$01,
             $FF ' end of left turn

' Right turn EEPROM Values
RTurn   DATA $01,$26,$01, $07,$B0,$04,
             $00,Word Pos2, $06,Word Pos2, $04,Word Pos3, $02,Word Pos3,
             $01,$B0,$04, $07,$26,$01,
             $03,$26,$01, $05,$B0,$04,
             $00,Word Pos3, $06,Word Pos3, $04,Word Pos2, $02,Word Pos2,
             $03,$B0,$04, $05,$26,$01,
             $FF  ' end of right turn

'                Raise        Center       Lower
Adjust  DATA $01,$26,$01, $00,$EE,$02, $01,$B0,$04 'Leg1
        DATA $03,$26,$01, $02,$EE,$02, $03,$B0,$04 'Leg2
        DATA $05,$B0,$04, $04,$EE,$02, $05,$26,$01 'Leg3
        DATA $07,$B0,$04, $06,$EE,$02, $07,$26,$01,'Leg4
        $FF ' end of adjust legs
'---- [End EEPROM Data] -----------------
DIRL = %01111111              ' P0 - P6 are outputs
'---- [Troubleshoot section] ---------------
' Remove comments to run related
' walking gaits without first entering
' button interface.
'Index = Adjust
'DEBUG HEX? Adjust, HEX ?Index
'END
'Index = Forward
'Index = Back
'Index = LTurn
'Index = RTurn

'RightRamp = $7
'LeftRamp = $7

'GOTO Walker
'GOTO Leg_Adjust

'GOTO Get_LED
'---- [setup] -----------------


Get_Stick:
PULSIN StickXPin, 1, stickXPos        ' Read joystick positions
PULSIN StickYPin, 1, stickYPos        ' from transmitter P13, P12
  'DEBUG ?stickYPos, ?stickXPos
  'PAUSE 1000
' Convert stickYPos position to HighNib of selectedMode
  SELECT stickYPos
    CASE 580 TO 650
```

```
      selectedMode = $10
    CASE 651 TO 730
      selectedMode = $20
    CASE 731 TO 800
      selectedMode = $00
    CASE 801 TO 870
      selectedMode = $30
    CASE 871 TO 960
      selectedMode = $40
  ENDSELECT

'DEBUG HEX ?selectedMode.HIGHNIB, HEX ?currentMode.HIGHNIB

' Convert stickXPos position to LowNib of selectedMode
  SELECT stickXPos
    CASE 500 TO 700
      selectedMode = selectedMode | $01
    CASE 701 TO 820
      selectedMode = selectedMode | $00
    CASE 821 TO 999
      selectedMode = selectedMode | $02
  ENDSELECT

'---- [Troubleshooting Section] ---------
'currentMode = $01
'selectedMode = $42
'DEBUG HEX ?selectedMode,CR
'GOTO Get_Stick
'---- [End TS] --------------------------
  ' Do nothing if stick is in
  ' the center Position
  IF selectedMode = $00 THEN
    Segments = $7E      'Display Zero
    GOTO Get_Stick
  ENDIF
  ' Continue walking if stick has not
  ' changed position
  IF currentMode = selectedMode THEN
    GOTO Continue
  ENDIF
'DEBUG HEX ?selectedMode, HEX ?currentMode ,CR

'---- [Gait Selection] -------------------------------------------------
Select_Mode:
  currentMode = selectedMode
  GOSUB Get_LED               ' Get LED display value
  ' Turnning gaits
  SELECT selectedMode
    CASE $01
      newMode = LTurn         ' Turn Left
      RightRamp = Fast
      LeftRamp = Fast
      GOTO Walker
    CASE $02
      newMode = RTurn         ' Turn Right
      RightRamp = Fast
      LeftRamp = Fast
      GOTO Walker
  ENDSELECT
```

5

```
' Check HIGHNIB to select
' EEPROM starting address
  IF selectedMode.HIGHNIB <= $2 THEN
    newMode = Forward
  ELSE
    newMode = Back
  ENDIF

' Check HIGHNIB to select
' Ramp speed
  IF (selectedMode.HIGHNIB = $1) OR (selectedMode.HIGHNIB = $4) THEN
    Ramp = VeryFast
  ELSE
    Ramp = Fast
  ENDIF


' Check LowNib to determine
' left and right leg ramps
' 0 = straight
' 1 = slow left side
' 2 = slow right side
  SELECT selectedMode.LOWNIB
    CASE $0
      'DEBUG "Straight ", CR
      RightRamp = Ramp
      LeftRamp = RightRamp
    CASE $1
      'DEBUG "left ", CR
      RightRamp = Ramp
      LeftRamp = (Ramp+2)
    CASE $2
      'DEBUG "Right ", CR
      LeftRamp = Ramp
      RightRamp = (Ramp+2)
  ENDSELECT
  'DEBUG HEX ?RightRamp, HEX ?LeftRamp

'DEBUG CLS, "Executing Walker",CR,CR, HEX ?selectedMode
'-----[ Main Walking Routine ]-----------------------------------------
Walker:
  ' Get EEPROM Address
  Index = newMode
  ' Read EEPROM value into servoAddr
  READ Index,ServoAddr
  'DEBUG "Entered Walker",CR
  ' Loop while servoAddr is not equal to $FF
  ' the end of Forawrd EEPROM data
  DO WHILE servoAddr <> $FF
  'DEBUG "Looping",CR
  ' If servoAddr is odd lift the leg fast
  ' If even assign left and right ramp
    IF (servoAddr // 2) = 1 THEN
      Ramp = LiftRamp
    ELSE
      IF (servoAddr = $00) OR (servoAddr = $02) THEN
        Ramp = RightRamp
      ENDIF
      IF(servoAddr = $04) OR (servoAddr = $06) THEN
        Ramp = LeftRamp
      ENDIF
    ENDIF
    GOSUB Send_EEPROM_Info_to_PSC
    'PAUSE 50
  LOOP
  GOTO Get_Stick
  Continue:
  GOTO Walker

'----- [Serial Out EEPROM values to PSC] ------------------------------
Send_EEPROM_Info_to_PSC:
  ' Read EEPROM data
  READ Index, servoAddr,servoPosition.LOWBYTE,servoPosition.HIGHBYTE
  ' Send EEPROM data to PSC
  SEROUT PSC,33164,["!SC",ServoAddr, Ramp,
                         servoPosition.LOWBYTE, servoPosition.HIGHBYTE, CR]
  'DEBUG HEX ?servoAddr, HEX ?servoPosition.LOWBYTE, HEX ?servoPosition.HIGHBYTE
  Index = Index + 3
  READ Index,servoAddr
  'PAUSE 2
RETURN

Get_LED:
GaitValue = 0                  ' Initialize
Counter = 0
READ counter+gait, GaitValue   ' Read gait value into GaitValue
DO
  READ counter+gait, GaitValue
  counter = counter + 1
LOOP  WHILE GaitValue <> selectedMode

  READ counter-1,Segments
  DEBUG HEX ?counter, HEX ? GaitValue, HEX ?Segments
  'END
RETURN
```

SHARP INFRARED DISTANCE READING CODE

This code is used to obtain distance readings from the Sharp Infrared attached to the front of the QuadCrawler.

```
'----[sharp_ir_demo.bs2]-----------------
'{$STAMP BS2}
'{$PBASIC 2.5}
'
'   File....... sharp_ir_demo.BS2
'   Purpose.... Distance readings using a Sharp GP2D12
'               IR sensor and ADC0831 8-bit Analog
'               to Digital Converter
'   Author..... CustCrawler Inc. (Mike Gebhard)
'   E-mail..... support@crustcrawler.com
'   Started.... 31 July 2004
'   Updated.... 31 July 2004
'
'
'
' Schematics:
'
' LTC1298 to BOE and GP2D12 connections
'                       Vdd(+5V)
'          ____________  |
'         |1           | |   3.3 micro F
' pin 9  -|CS   A  Vcc |-|------||-------|
'         |     D      |        Vdd(+5V) |
' GP2D12 -|Vin+ C  CLK |- pin 11   |     |
'         |     0      |           /    Vss
'     |---|Vin= 8  Dout|- pin 10   \
'     |   |     3      |           / 1k resistor
'     |---|GND  1  Vref|-----------|
'     |   |____________|           /
'     |                            \ 1k resistor
'   Vss                            /
'                                  |
'                                 Vss
'
' GP2D12 to BOE and LTC1289
'          _________________
'         |     GP2D12      |
'         |     Front       |
'         |_Vo____GND___Vcc_|
'            |     |     |
'           Vin+  Vss   Vdd
'         LTC1298 BOE   BOE
'
'---------------------------------------------------------

' Pin Assignments
CS      CON   9        ' Chip Select
DIO_10  CON   10       ' Data I/O pin 10
CLK     CON   11       ' Clock

ADResult  VAR   Byte   ' Variable to hold 8-bit AD result.
```

```
HIGH CS                 ' Deactivate ADC to begin.

Main:
  GOSUB Get_Average_AD_Reading               ' Get data from ADC.
  DEBUG CLS
  DEBUG "Raw 8-Bit data:  ",DEC ADResult,CR
  PAUSE 500                                  ' Wait a half second.
GOTO Main                                    ' Endless loop.

Get_Average_AD_Reading:
    LOW CS                                   ' Activate the ADC.
    SHIFTIN DIO_10,CLK,MSBPOST,[ADResult\8]  ' Get data bits.
    HIGH CS                                  ' Deactivate the ADC.
RETURN
```