Sample Project: Simple Hexapod Walker

1. Introduction 2
2. Materials and Tools 3
3. Construction 5
4. Sequencing the Hexapod Gait 15
5. Using a Script for Obstacle Avoidance 18
6. Suggested Modifications and Improvements 20
7. Conclusion and Community 21

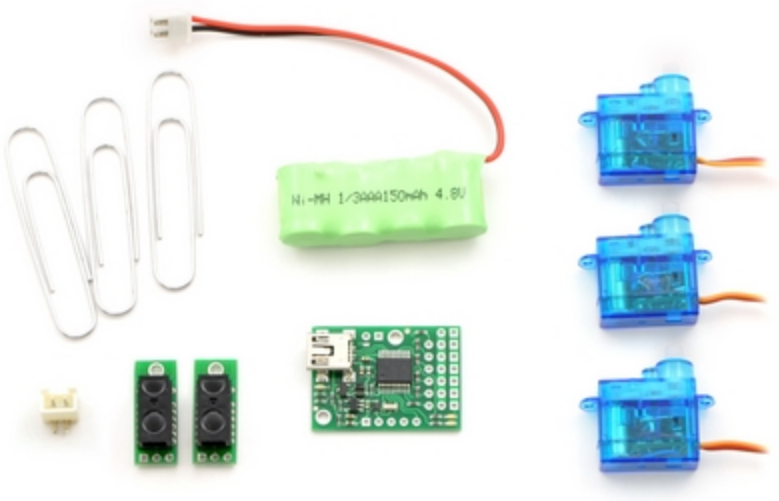
1. Introduction

Six-legged locomotion is a simple, robust system of walking that is very popular both in the animal kingdom and among robotics hobbyists. Robot hexapods range from simple one-motor toys to advanced platforms with 18 or more servos. This tutorial shows you how to build a very simple autonomous hexapod robot using just three servos. The 2"-high hexapod is capable of walking forward and backward, and can turn left and right. Two forward-looking distance sensors provide obstacle avoidance. The brain of the hexapod is the **Pololu Micro Maestro** **[http://www.pololu.com/catalog/product/1351]**, a 6-servo controller that can read inputs and play motion sequences in a stored script.

See the **Micro Maestro User's Guide** **[http://www.pololu.com/docs/0J40]** for complete documentation on the Micro Maestro.

2. Materials and Tools

Parts list:



Parts you will need to build the hexapod robot.

Quantity	Part #	Part	Notes
1	1351	**Pololu Micro Maestro Partial Kit**	Get the kit version so that you can solder in your own wires for the most compact possible robot.
3	1053	**Sub-Micro Servo 3.7g Generic**	These generic servos provide the lowest possible cost and weight, but you may substitute other servos, such as the **Power HD sub-micro servo HD-1440A**, to customize the design.
2	1134	**Pololu Carrier with Sharp GP2Y0D810Z0F Digital Distance Sensor 10cm**	This is a tiny distance sensor with a long enough range to keep your hexapod out of trouble.
1	1171	**Battery Pack: 4.8 V, 150mAh**	This battery pack will provide enough power at about 5 V to power the hexapod for five or ten minutes.
1	1168	**2.5 mm Shrouded Male Connector: 2-Pin, Right Angle**	The polarized connector lets you connect the battery pack safely to the Micro Maestro.
3	–	“Jumbo” paper clip	Used to form the legs of the hexapod. These should be 6" long when unfolded.

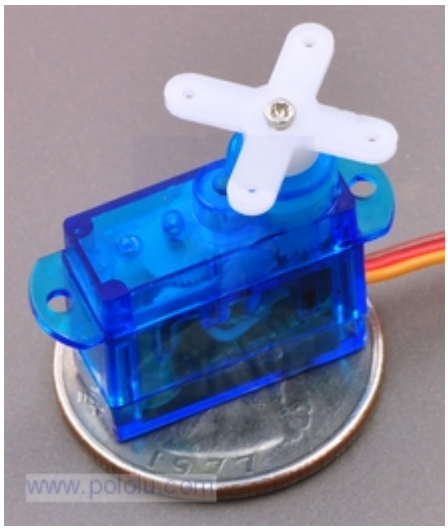
Tools required:

- Soldering iron and solder
- Hot glue gun
- Wire stripper

- Long-nose pliers
- Diagonal cutter
- Some wire for connecting the parts

Most of these parts are available in the **Tools [http://www.pololu.com/catalog/category/5]** section of the Pololu web site. A hot glue gun is available at most craft stores for a few dollars.

Update: The case for **sub-micro servo 3.7g generic [http://www.pololu.com/catalog/product/1053]** has changed slightly since this sample project was written. Versions shipping now have solid black cases instead of transparent blue ones and a portion of the top plane on the opposite side of the output shaft is now slanted rather than completely flat, but they still work as simple hexapod robot actuators as described in this project. The pictures below show the two version side by side, with the old version on the left and the new version on the right.



3. Construction

Step 1: Attach the battery connector.
Using a pair of pliers, flip the leads on the battery connector to the other side.

Bending the leads on the battery connector to the other side.

Verify that this allows you to plug in the battery as shown below, with the **black** wire connected to ground and the **red** wire connected to BAT, then solder in the connector.

Soldering a power connector to the Micro Maestro.

Step 2: Set up the Maestro for self power.
With your battery *disconnected*, attach a wire (red) from the positive terminal of the battery connector to VIN. Take care not to short or damage any of the components on the board. Now, with the battery plugged in, your Maestro should be powered-up and slowly flashing its yellow LED, indicating that it is waiting to detect the baud rate on the serial communication. You will not be using serial communication for this project, so you need to disable baud rate detection in the next step.

Self power and a battery connector on the Micro Maestro.

Step 3: Verify that the Maestro and servos are functioning.
The Maestro Control Center is used for configuration and control of the Micro Maestro, for testing, debugging, scripting, and more. See **Section 4** of the **Micro Maestro User's Guide [http://www.pololu.com/docs/0J40]** for complete instructions on using the Maestro Control Center.

Launch the Maestro Control Center and configure your Maestro for "USB Dual Port" mode. The yellow LED should now be mostly off, flashing very briefly about once per second. On the Status tab, enable servo ports 0, 1, and 2, and the yellow LED will start double-blinking, indicating that some ports are active.

Next, using a piece of male header (included with the Maestro), temporarily connect a servo to port 0. Make sure to connect the wires correctly, with the brown or black wire connected to ground. You should hear a short high-pitched whine as the servo activates, moving to and holding a position in the middle of its range. After a fraction of a second, when the servo has reached its position, it should be silent. Move the slider from 1000 µs to 2000 µs to test the motion of the servo. Test all three ports and all three servos before continuing with the assembly.

Configure each of the servo ports to "Go to" 1500 µs on startup. This will make it easier to align the legs later on.

Configure ports 3, 4, and 5 to be inputs. This is important, since you will be connecting sensors later and want to avoid shorting them out!

Disconnect the battery before continuing.

Step 4: Construct the body by gluing the servos together.
Remove the mounting tabs from all three servos with diagonal cutters. The tabs are not needed for this project and can interfere with the motion of the servos.

A sub-micro servo with mounting tabs clipped off.

Next, join the servos with a few dabs of hot glue as shown below. You do not need much glue to hold them securely! Try to align the corners precisely to make flat surfaces for mounting the other parts.

Try to align the corners.

Clip the servo cables, leaving at least 2" (more if you are less experienced with soldering). Strip a small amount of wire from the end of each cable.

Cut and strip the servo leads, leaving about 2” of wire.

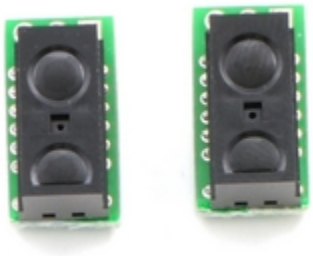
Step 5: Solder the servos and sensors to the Micro Maestro.
This step requires patience and care. A second pair of hands could be very useful.

Use solder to tin the leads of the servos so that they can be connected initially without additional solder. Looking at the pictures below for reference, place the Maestro on the back side of the body and place the middle wires over the front of the Maestro and into the holes for channel 1. Holding the wires in these holes, pull the Maestro away from the body, then touch the soldering iron to each connection so that the small amount of solder on the wires melts and holds them in place. You should now be able to add more solder to each of the connections, until the holes are filled and the wires are held securely. Check carefully for loose strands of wire, which could cause shorts.

Continue, soldering the right servo to channel 0 and the left servo to channel 2, so that the servos are arranged in the same relative positions as the ports.

Connecting the servos to ports 0, 1, and 2 on the Micro Maestro.

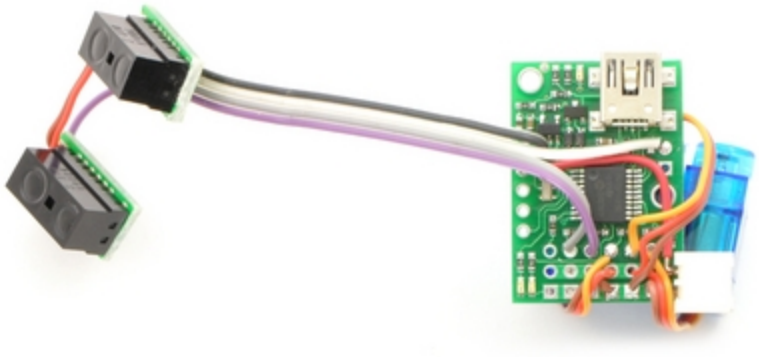
Cut the sensor boards with a rotary tool, grinding wheel, diagonal cutters, or a jeweler's saw, removing the part containing the unneeded mounting hole, so that they are as small as possible. (Make sure you do not cut any traces.) Then solder them to a cable so that you can connect them to the Maestro. The example below uses a 4-wire ribbon cable, sharing the power and ground connections for the two sensors. Ribbon cable makes the assembly relatively clean, but you can use whatever wire you have available. Look ahead in the instructions to see where the sensors are going to go, and make sure that you have a long enough cable. Think about how to keep the wires close to the body and out of the way of the legs and servos.



Digital distance sensors with trimmed carrier boards.

Soldering the sensors to a four-pin cable.

Solder the sensor power and ground to +5V and ground on the Maestro, and connect the outputs of the right and left sensors to channels 3 and 4, respectively. Note that we use +5V instead of the battery voltage so that the Maestro channels will never see higher voltages – and another benefit is that the sensors will work under USB power, without the battery plugged in.



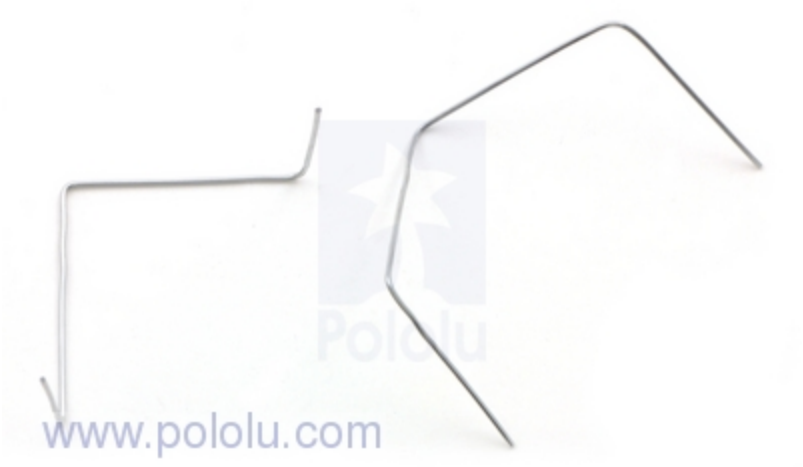
Soldering the sensor cable on to the Maestro.

You now have a complete electrical assembly. Plug in the batteries, and the sensors should become active, turning on their red LEDs whenever they detect an object within 10 cm. With the Maestro control center, you should be able to see the input value change from 255, when no object is present, to a low value of 40 or so, when an object is detected. If the LEDs are always on, you probably forgot to set the ports to inputs in Step 3.

Step 6: Construct the legs.
Unfold the paper clips into straight pieces of wire. Pliers make ugly dents in the metal, so try to use your fingers and the edge of a table to do this.

Straighten the paper clips as much as possible.

The wires should be six inches long. To make the front and back legs, fold two of them into 1.5" sections, with 90° angles between the sections, like this:



The front and back legs of the hexapod. The segments at the end should be 1.5” in length.

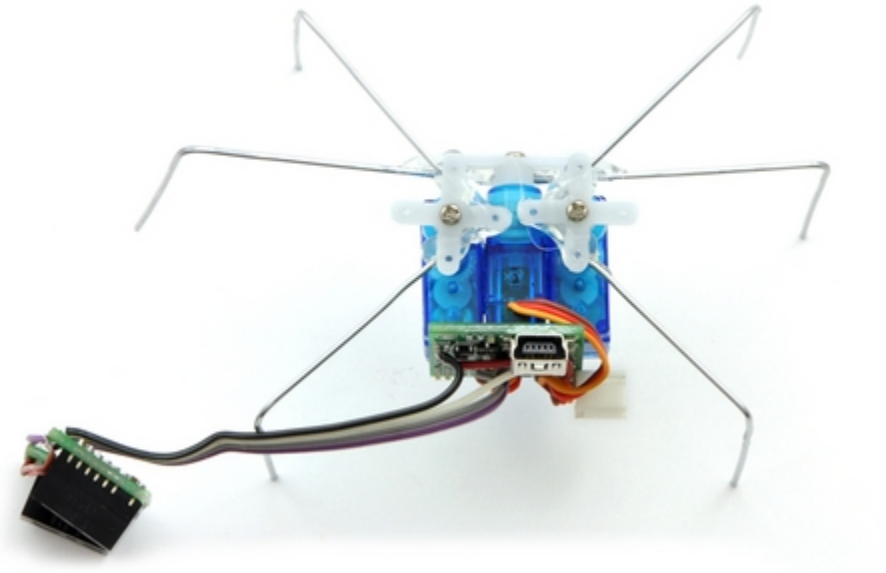
Fold the third piece into an M shape, with sections of length 1.25", 1.75", 1.75", and 1.25", like this:

The middle legs of the hexapod. The segments at the end should be 1.25” in length.

Hot-glue the legs onto servo horns. Use a straight horn for the middle legs and cross-shaped or round horns for the front and back legs.

Gluing the hexapod legs to servo horns.

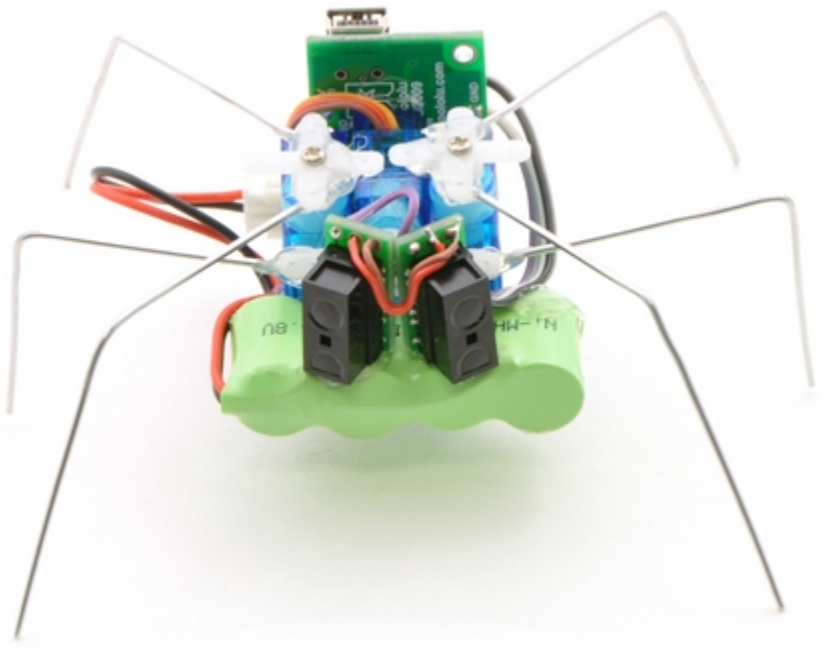
With battery power connected, so that the servos hold their neutral positions, put the horns onto the servos so that the legs are as close as possible to neutral positions, as shown in the picture. Fix them in place with the included screws, holding the legs as you tighten them so that you do not apply torque to the fragile servo gears. Glue the Maestro to the back side of the servos, flush with the bottom.



Attaching the legs and Maestro to the servos.

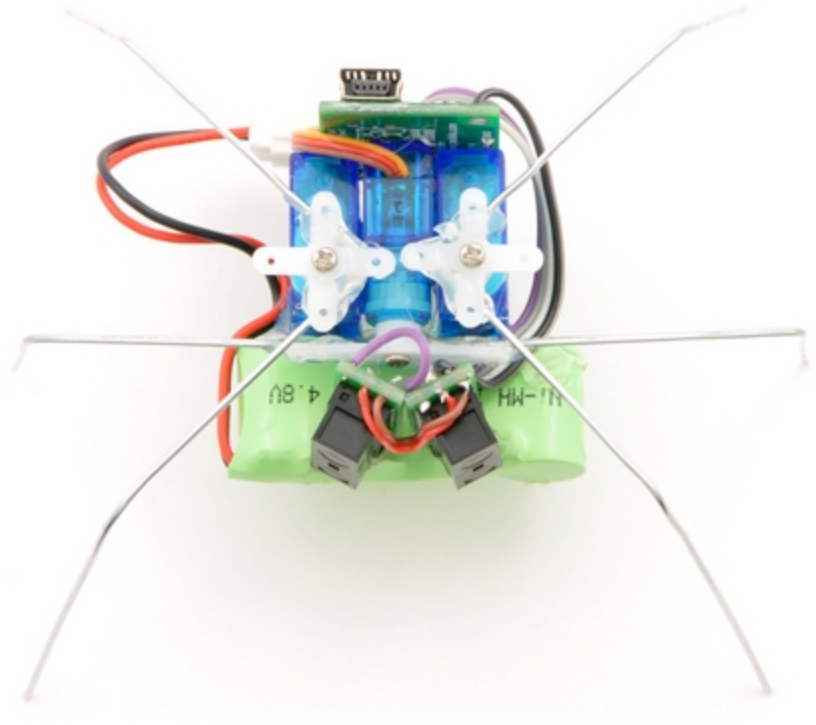
Important: Never apply torque to the legs with your hands, attempt to prevent them from moving, or backdrive them. Servo gears can be easily broken, so they should only ever move under their own power. Use the Maestro Control Center to experiment with different positions, instead of forcing the servos.

Step 7: Attaching the battery and sensors.
Glue the battery to the front of the hexapod, flush with the bottom, so that there will be as much clearance as possible. Make sure that the middle legs have plenty of room to turn left and right without hitting the battery. Glue the sensors to the top of the battery, angled to the left and right.



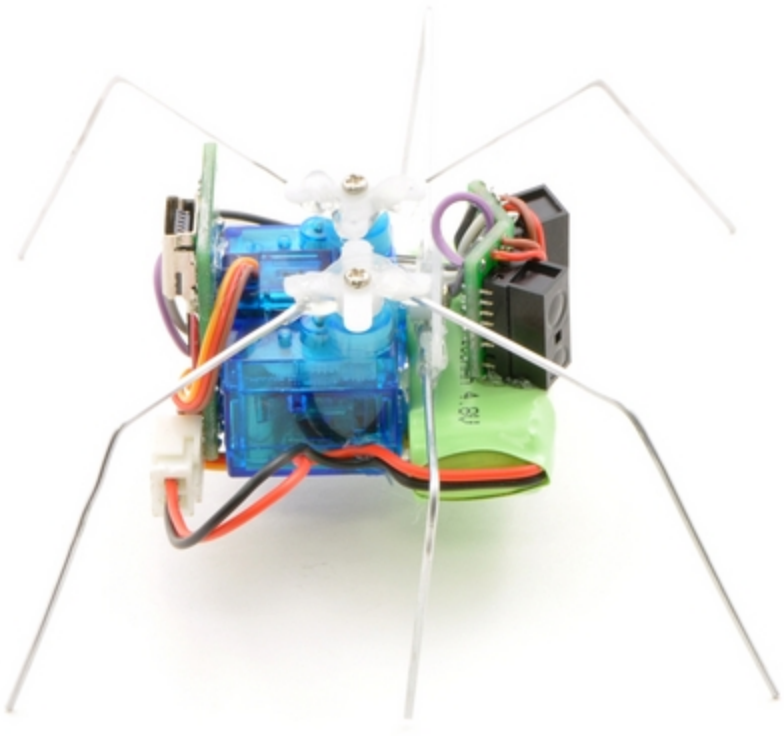
The assembled hexapod, front view.

Take care that the wiring does interfere with the motion of the middle legs. You might want to route yours differently. Use small drops of hot glue as necessary to hold the wires in place.



The assembled hexapod, top view.

Step 8: Final touches.
Use the Maestro Control Center to find the neutral positions (where the legs are as symmetrically arranged as possible) as well as their safe minimums and maximums. Set the neutral positions as the "Go to" values for each servo, and set *Min* and *Max* values so that your hexapod will never destroy itself. Adjust the angles of the wires so that all six feet touch the ground in the neutral position, probably by bending the front and back legs to more than 90°.



The assembled hexapod, side view.

One final, optional thing that you might want to do is to add a dab of hot glue to each foot, so that the metal is less likely to scratch up your work surface. Your hexapod is now ready to be programmed!

4. Sequencing the Hexapod Gait

Gaits

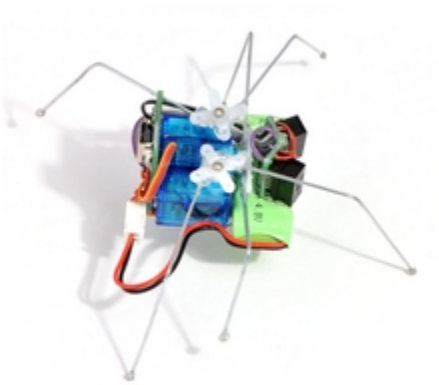
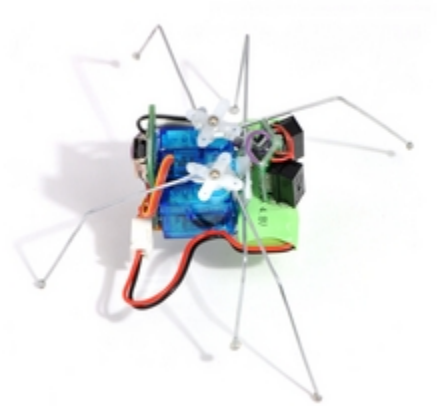
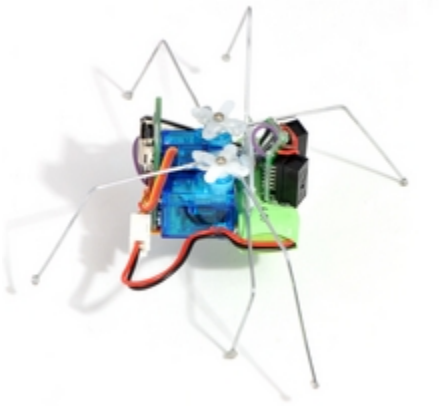
Now that you have constructed your hexapod, it is time to make it walk. A method of walking forward with legs is called a *gait*, and animals or robots with many degrees of freedom have a variety of gaits available – humans can walk, run, hop, or skip; horses can walk, trot, canter, or gallop, and so on. Your hexapod is so simple that it has just one possible gait for forward motion, called the *tripod gait*.

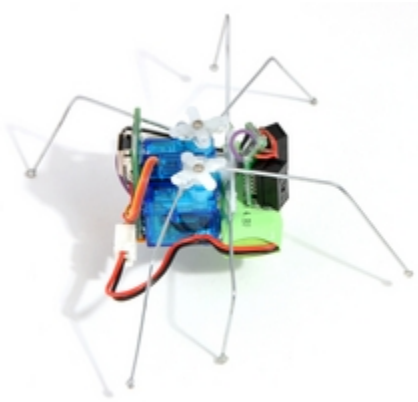
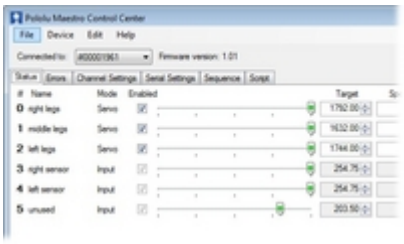
In the tripod gait, your hexapod has three feet in contact with the ground at all times: the front and back legs on one side and the middle leg on the other side. The angle of the middle servo determines which side is up and which side is down. It achieves forward motion by pushing those feet backwards against the ground while the other feet move forward through the air. Then the hexapod shifts its weight to the other three feet and moves forward in the same way. By continuously shifting its weight using the middle legs, then moving the raised feet forward, it walks forward.

Walking forward with the Maestro Control Center

You can easily assemble motion sequences using the Maestro Control Center's sequencer feature. For documentation on the sequencer, see **Section 4.c** of the **Micro Maestro User's Guide [http://www.pololu.com/docs/0J40]**.

Use the controls on the Status tab of the Maestro Control center to move your hexapod to each of these four positions, pressing the "Save Frame" button after each, to save a sequence.

Frame 0 right front & back, left middle legs touching the ground		
Frame 1 down legs move back, front legs move forward		
Frame 2 weight shifted to the other tripod		

Frame 3 down legs move back, front legs move forward		

In the screenshots of the Maestro Control Center, you can see that the servos are always either at their minimum or maximum values, which you should have configured to be safe values that do not cause the servo to strain. Your numbers might be slightly different from the ones shown here. If you have assembled the servos in a different configuration or connected them to different ports, you will, of course, have different very settings for each frame.

On the Sequence tab, you should now have four frames saved. Select “Play in a loop” and play the sequence to see your hexapod walk. Rename your sequence to “forward” before continuing.

Walking forward autonomously
Click the button “Copy Sequence to Script” to convert your sequence into a script that can be saved on the Maestro. If you select the “Run script on startup” option on the Script tab and apply settings, your hexapod will automatically start to walk. You can now disconnect it from USB and allow it to walk on its own.

Reconnect USB and click “Stop Script” or disable the “Run script on startup” option to get it to stop walking.

Backwards and turning gaits
On the status tab, start again with Frame 0 but go through the sequence of motions in reverse to get backwards walking: Frame 0, Frame 3, Frame 2, then Frame 1. Save this sequence under the name “backwards”, and test that it moves your hexapod backwards.

Turning is different. To create turning sequences, you will need to move all front and back legs forward or backward together, instead of moving the two sides in opposite directions. Try it out, and save two more sequences: “right” and “left”, verifying that they turn the hexapod right and left.

You are now ready to program your hexapod to avoid obstacles.

5. Using a Script for Obstacle Avoidance

The Micro Maestro has an internal scripting language that can store sequences, read sensors, and link everything together to form intelligent behaviors, making your hexapod truly autonomous. For complete documentation on the scripting language, see **Section 6** of the **Micro Maestro User's Guide [http://www.pololu.com/docs/0J40]**.

Once you have set up all of your basic gaits, erase your script, then click the "Copy All Sequences to Script" button on the Sequence tab. This button adds a set of subrountines that you can use to access the various gaits from within a script.

You can then call these subroutines from your script. For example,

```
begin
  forward
  left
repeat
```

will cause the hexapod to repeatedly step forward then turn left, over, and over. You will probably want to take into account the sensor readings, which can be accessed using the GET_POSITION command. Here is an example of a very simple program that uses the sensor readings to try to avoid objects – customize this to get the behavior you want!

```
start:
  # back up if both sensors see an object
  left_sensor right_sensor logical_and
  if back back back goto start endif

  # back up and turn right if the left sensor sees an object
  left_sensor if back right right right goto start endif

  # back up and turn left if the right sensor sees an object
  right_sensor if back left left left goto start endif

  # otherwise, if there is nothing ahead, walk forward
  forward
  goto start

# returns true if the left sensor sees an object
sub left_sensor
  4 get_position 512 less_than
  return

# returns true if the right sensor sees an object
sub right_sensor
  3 get_position 512 less_than
  return

### Sequence subroutines: ###

# back
sub back
  100 4992 5312 5056 frame_0_1_2 # Frame 0
  120 7168 6976 frame_0_2 # Frame 1
  100 6528 frame_1 # Frame 2
  120 4992 5056 frame_0_2 # Frame 3
  return
# forward
sub forward
  100 7168 5312 6976 frame_0_1_2 # Frame 1
  120 4992 5056 frame_0_2 # Frame 2
  100 6528 frame_1 # Frame 3
  120 7168 6976 frame_0_2 # Frame 0
  return
# left
sub left
  100 7168 5312 5056 frame_0_1_2 # Frame 0
  150 4992 6976 frame_0_2 # Frame 1
```

```
  100 6528 frame_1 # Frame 2
  150 7168 5056 frame_0_2 # Frame 3
  return
# right
sub right
  100 4992 5312 6976 frame_0_1_2 # Frame 1
  120 7168 5056 frame_0_2 # Frame 2
  100 6528 frame_1 # Frame 3
  120 4992 6976 frame_0_2 # Frame 0
  return

sub frame_0_1_2
  2 servo 1 servo 0 servo delay
  return

sub frame_0_2
  2 servo 0 servo delay
  return

sub frame_1
  1 servo delay
  return
```

6. Suggested Modifications and Improvements

Here are some ideas for improvements or modifications that could be made to the hexapod design:

- More complicated scripted behaviors – help your hexapod get out of stuck situations more reliably.
- More robust sensor readings. The example code only read the sensors once after each sequence – can you do better than that, and detect obstacles sooner?
- Alternative servos – the entire design could be scaled up.
- A power switch so that the battery does not have to be unplugged over and over.
- **Boost regulation** **[http://www.pololu.com/catalog/product/791]** for consistent power from a more compact (lithium?) battery.
- **QTR sensors** **[http://www.pololu.com/catalog/product/958]** for line following or table-edge detection.
- Light sensors for light-seeking or dark-seeking behaviors.

7. Conclusion and Community

The Micro Maestro can act as the brain for a simple hexapod robot. Three of the Maestro's six channels are used for the servos, so there are three channels available for sensors such as distance sensors, which the hexapod can use to react to its environment. The Maestro's scripting functionality allows simple behaviors and motion sequences to be programmed onto the device, making the hexapod autonomous.

Did you manage to build the hexapod or something like it? Please join us on our **robotics forum [http://forum.pololu.com/]** to ask questions, give feedback, or share your projects. We would love to hear about your experiences, and we would be delighted to see any improvements or alterations you make!

We will post hexapod projects submitted by others below:

Hexapod clone (my first bot) [http://letsmakerobots.com/node/16683]
A hexapod built by letsmakerobots.com user Bruud using the Micro Maestro and HD-1440a servos. He used the assembled version of the Micro Maestro, plugging wires in instead of soldering them, so that he could re-use all of the parts later.

ShC v1 (Simple hexapod Clone version 1) [http://letsmakerobots.com/node/15458]
This is a simple hexapod built by a letsmakerobots.com user who was inspired by the Pololu Simple Hexapod Walker.