BUILD THE ULTIMATE ROBOT

Part 6: An Arm for Megabot

by Michael Simpson

Last month, we used Wi-Fi to control Megabot. One of the problems I had was maneuvering the robot in close quarters. The laptop's built-in webcam worked but it just did not give a good view of what was going on around the robot. This month, we will add an external webcam to a fully articulated robot arm. We then will add buttons on our desktop controller program to allow us to control this arm.

STEP 1 - ADD A SECOND DECK

There is just enough room to mount an arm in front of the laptop, but it's a real tight fit. I had always planned on adding a second deck to the base, so now it's time. It will give us plenty of room to add more components and manipulators later.

I wanted to elevate the platform about 1-1/2" above the cradle. This will allow me easier access to the laptop and give me room to mount things on the bottom of the upper platform. To do this, I simply cut two more pieces of wood at 14" x 3" and attach them to the two sides of the cradle as shown in Figure 2. You only need a single wood screw on each end to hold it in place. Just make sure it sticks up 1-1/2" on both ends.

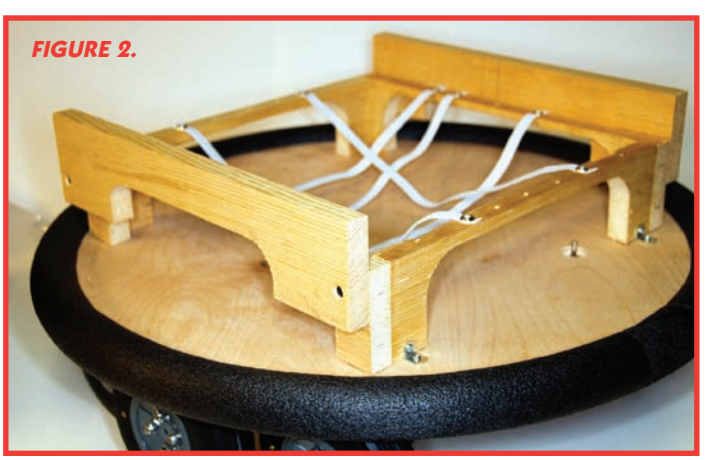


FIGURE 2.

The upper base can be made from any material. You can use 1/4" or 1/8" stock cut to the same diameter as the lower platform. In my case, I used some 1/4" plywood I picked up from my local home center. You don't have to purchase these in full sheets. The one I purchased came in a 2' x 4' section and only cost me about $6.

To cut out the upper base, I removed the cradle and foam from the edge and just traced it. You can also use the procedure I covered in Part 3. I used a handheld jig saw to cut out the platform.

To assemble the base, I first placed the upper platform

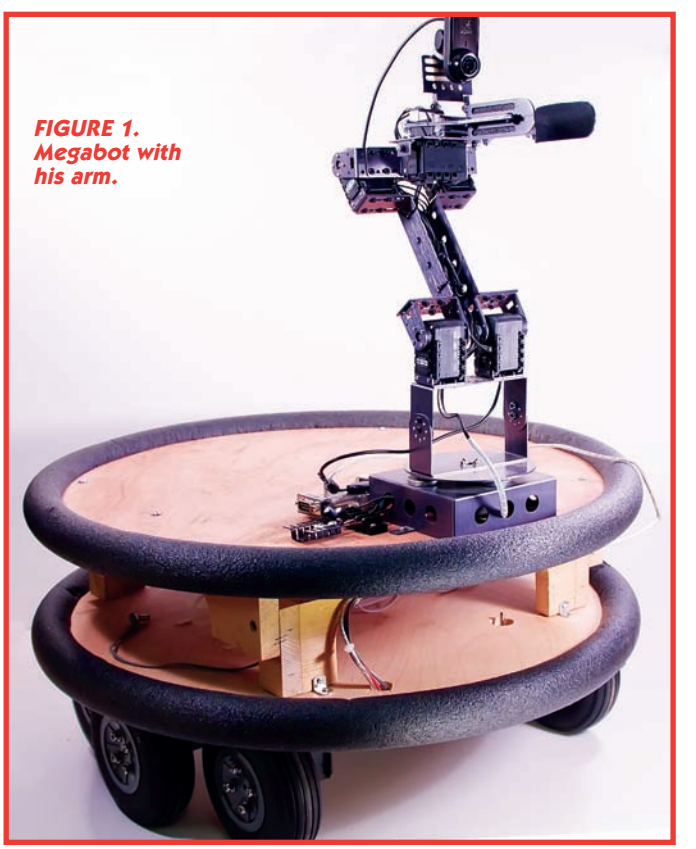


FIGURE 1. Megabot with his arm.

FIGURE 4.

FIGURE 5.

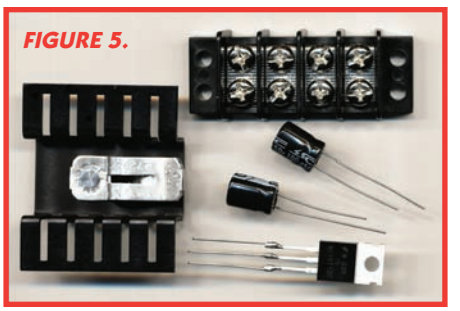


FIGURE 6.

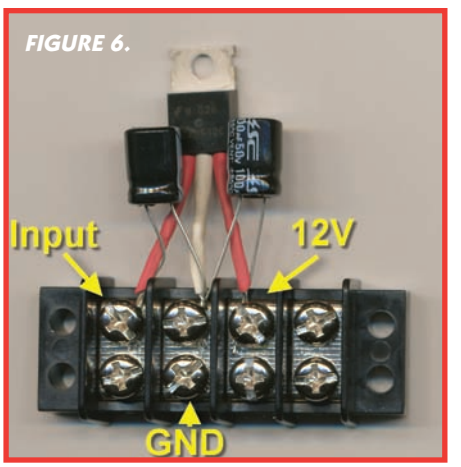
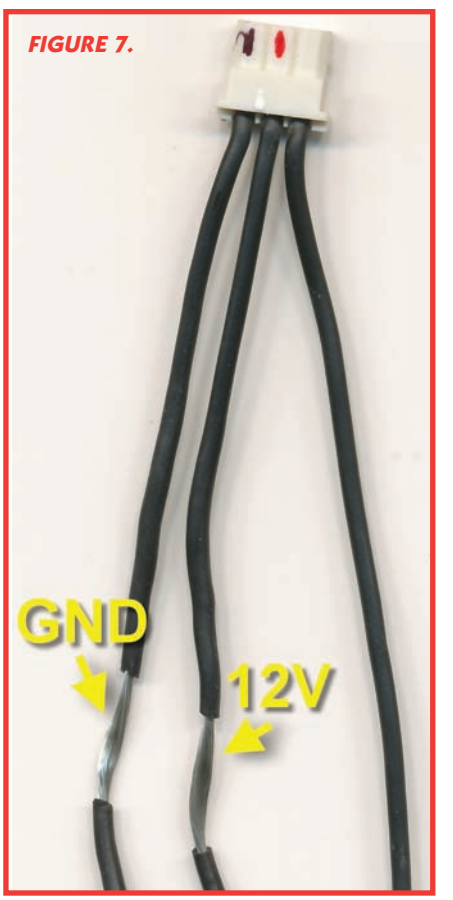


FIGURE 7.

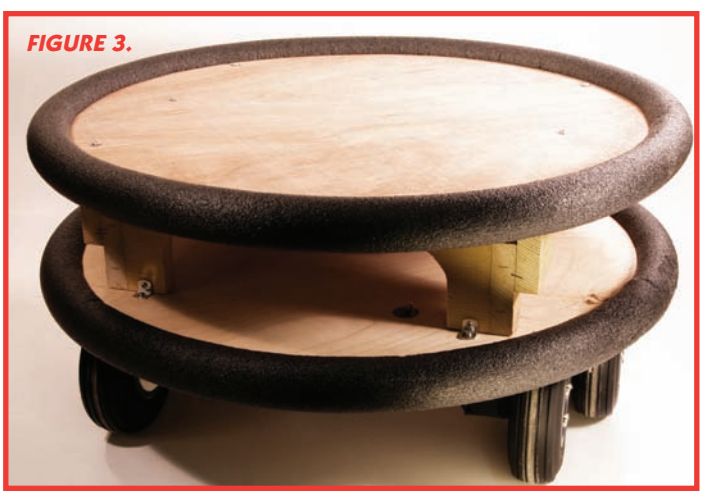


on a flat surface. The surface that will be the top should be face down. I then placed the lower platform — cradle and all — upside down on the upper platform. I used a wood square flat against the table surface and against both the upper and lower platform edges. Just butt the square against the two bases while it's flat on the table surface; first on one side, then the other. Once the two bases are lined up, trace the two 14" x 3" supports where they come in contact with the upper platform. You need to do this for two reasons. First, so that you can place the upper platform back on the supports once you flip the whole thing over. The second reason is so you can drill three pilot holes.

Remove the lower platform from the upper platform. You should now see two thin rectangles where the supports will be. You may not see a complete rectangle, but you should have an idea where the support will be placed. Drill three pilot holes in each rectangle: one on each end about an inch from the edge and one in the middle.

Now with Megabot upright, place the upper platform on the wooden supports, lining them up with the tracings. You are trying to get the upper platform lined up as it was when you did the tracing. At this point, you can carefully drill a pilot hole on one of the corners through the hole you drilled. Use a #6 washer and flathead screw, and attach the upper base to the platform as shown in Figure 3. Move to the opposite corner and make sure the markings are lined up, and repeat the process. You can now drill the pilot holes for the remaining four holes and attach with washers and screws. Add a foam bumper to the upper platform as outlined back in Part 3.

FIGURE 3.



STEP 2 - ADD A ROBOT ARM

I am going to use a CrustCrawler AX-12 Smart Arm (shown in Figure 4) for our manipulator. This arm uses the AX-12 Dynamixel actuator. The AX-12 uses the same command structure as the RX-64. Unfortunately, it has a different hardware interface; the AX-12 uses an asynchronous TTL level. The USB2Dynamixel controller has a small switch that allows you to connect it to the AX-12.

For the interface, we will use the USB2Dynamixel but we still have one more obstacle, The AX-12 has a much lower operating voltage than the RX-64s we are using. To solve this, I built a 3A, 12V regulator circuit. You will need the following components:

- Two 100 µF radial electrolytic capacitors

- TO-220 Heatsink
- Four position barrier strip
- 78T12CT voltage regulator

The first thing you need to do is to extend some leads on the regulator chip. I did this by soldering some 18 gauge wire to each lead as shown in Figure 5.

The easiest way to assemble the regulator is to take the two capacitors and twist them on to the leads of the regulator chip. Start by twisting the positive side of each capacitor on to the outside leads of the 78T12CT chip. Then, twist the negative leads on the capacitors with the center lead on the 78T12CT chip. I also added some heat shrink to help keep the regulator chip's leads from shorting out against the leads on the capacitors.

Once the leads are all twisted together, all you need to do is loosen the first three terminals and insert the leads and tighten them as shown in Figure 6.

Now, take one of the connectors included with an AX-12 (you should have one left after assembling the Smart Arm) and trim 1/2" of insulation about 2" from one end of the connector. Notice how I have marked the rounded end. This end of the connector will be plugged into the USB2Dynamixel.

Wrap the GND lead on the connector to the terminal marked GND as shown in Figure 7. Wrap the 12V lead on the connector to the terminal marked 12V as shown in Figure 8. Tighten the terminals and insert the marked end into the USB2Dynamixel three-pin connector.

Insert the other end of the connector through one of the holes on the left side of the base on the Smart Arm. Then, attach it to the unused connector on the AX-12 inside the Smart Arm base.

Next, attach the Smart Arm to the front of the upper platform as shown in Figure 9. I used some 3/8" #4 machine screws to attach mine. Lay out the terminal strip and USB2Dynamixel as shown in Figures 8 and 9. I attached the controller with double-sided tape and #4 machine screws on the terminal strip.

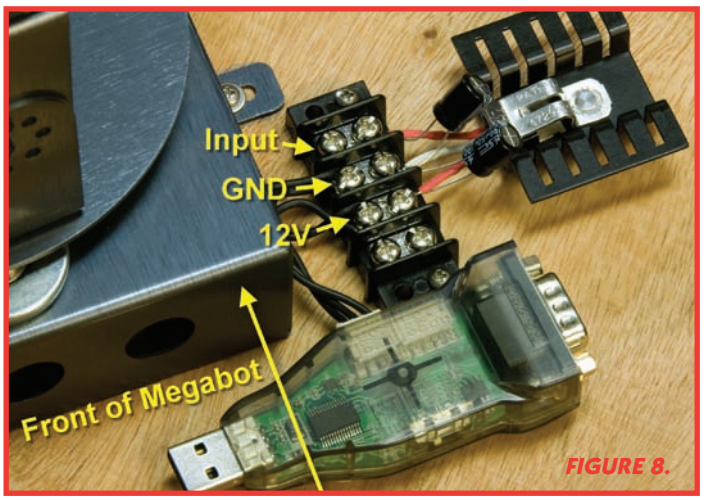


FIGURE 8.

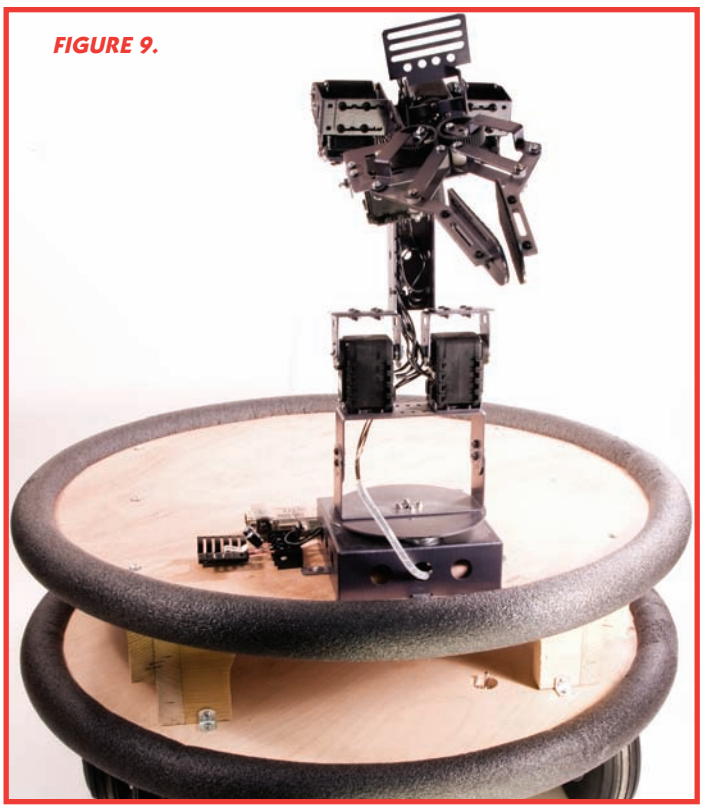


FIGURE 9.

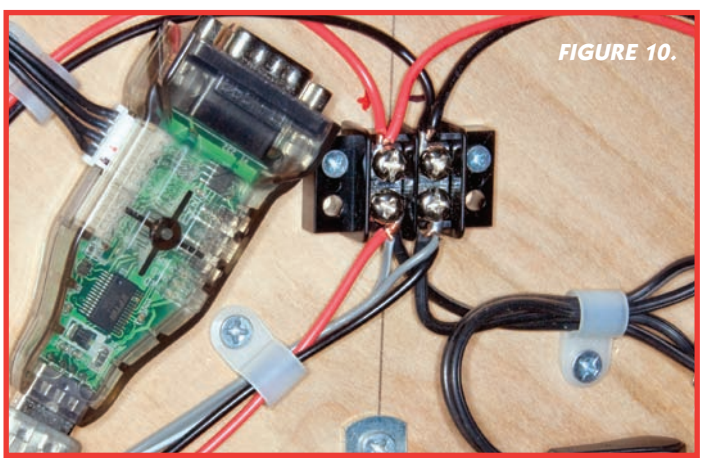


FIGURE 10.

STEP 3 - FINAL HOOKUP

To power the regulator, you need to run two wires from the switched side of the power source as shown in Figure 10, and connect them to the regulator Input and GND leads (shown in Figure 8).

In order to have a single USB interface to the laptop, I added the hub, as shown in Figure 11. A USB cable is also run from the RX-64 Dynamixel controller. The last attachment is a small webcam to the top of the arm as shown in Figure 12. Plug the webcam USB cable into an open port on the hub (shown in Figure 11).

PROGRAMMING THE BRAIN

Step 1

Let's start with a simple program that will let you

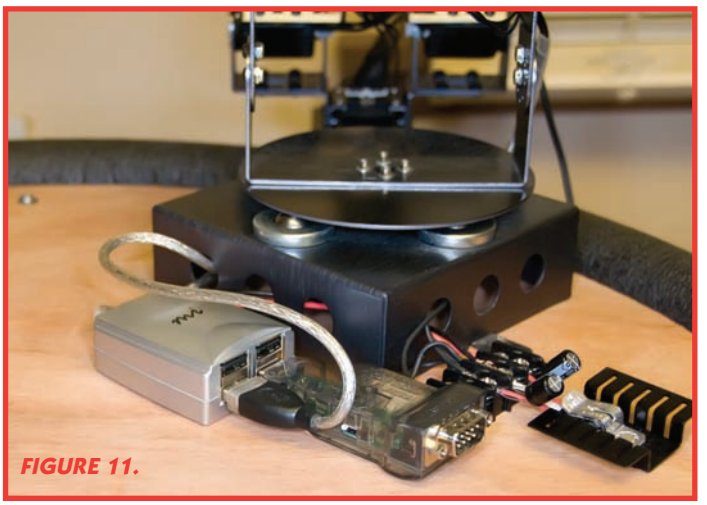


FIGURE 11.

exercise both the RX-64 and AX-12 actuators. The program is called Megabot_Actuator1b_DT.exe (available at **www.kronosrobotics.com**).

When you run this program on the PC that is connected to the USB2Dynamixel, it will display the output screen shown in Figure 13.

If the program is properly connected to the USB2Dynamixel, the status should return 1 and 1. If you don't get this result, you need to create a file named Port and place the number for both com ports connected to the USB2Dyanmixel controllers in this file. The first line in the file should be the com port connected to the RX-64. The second line in the file should be the com port connected to the AX-12. Once you make the changes, restart the program. If all the actuators are connected properly, you should get a voltage and temperature for each one. If you don't, go back and connect each actuator individually to the PC and program the IDs, as outlined in Part 5.

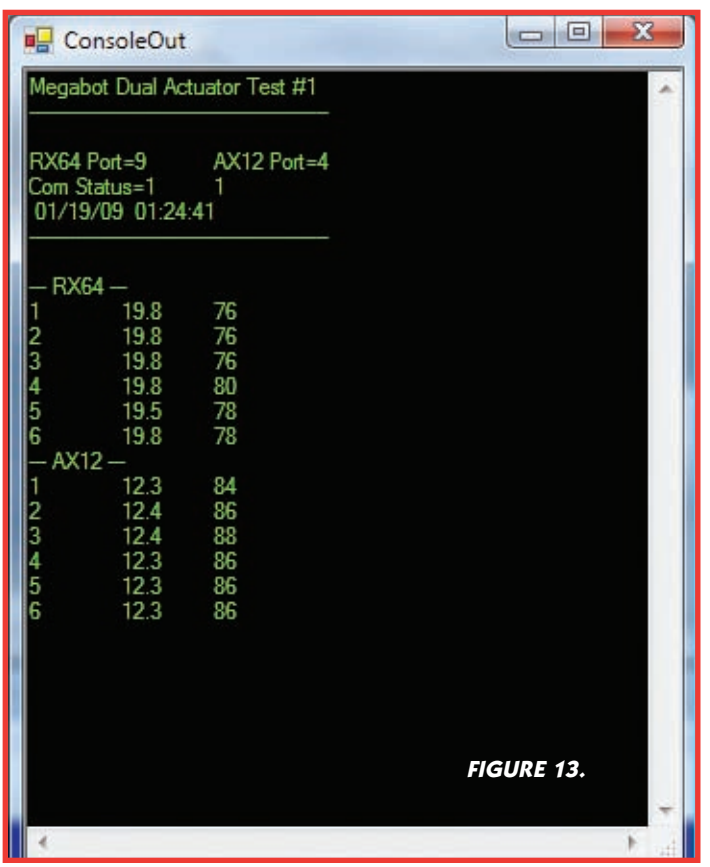


FIGURE 13.

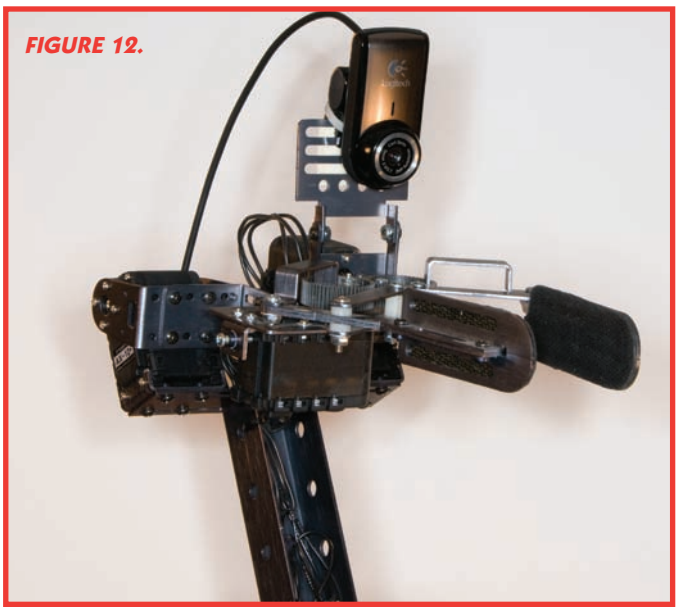


FIGURE 12.

Before moving on, let me say a few things about the USB2Dynamixel interface using Zeus. To open a communication channel, you use the USB2AXinit command. You supply a channel number so that Zeus can keep track of the com settings you supply. The problem is the USB2Dynamixel library was designed so that only one controller is connected to the PC at a time.

You can initialize multiple USB2Dynamixel controllers by issuing separate USB2AXinit commands. Just supply them with different channel numbers (First Parameter). When you issue one or more commands — like the USB2AXwriteword — you need to set up the correct channel. This is done by setting the AXchan variable with the correct channel. To help clarify this, take a look at the source code for the various code examples.

Step 2

Now that we have verified all the actuators are connected properly, it's time to make them move. Load up the program called Megbot_Actuator2b_DT.exe. This program will cause the wheels to move in one direction, then reverse. At the same time, the arm will articulate as well. If you don't see the wheels and arm moving, go back and check your power connections.

There are a couple functions in the source code that I have provided as shortcuts in case you want to dive in and start programming the Megabot. Figure 14 shows the actual commands and what they control on the arm.

The commands are as follows:

- armgrip

- armwrist
- armarm
- armknee
- armspin

In addition, the armopen and armclose functions will open and close the gripper without having to supply the gripper position.

Step 3

Now it's time for some fun stuff. Load the program called Megabot_Wifibotb_DT.exe on the laptop used in the Megabot. This is the server program shown in Figure 15. Its job is to listen on the network and wait for commands from the client program. When a command is received, it takes an action. Notice how the server program displays the IP address at the top of the form. In this case, it's 192.168.1.201. Keep this in mind, as we will need it later.

In Part 5, we used the built-in webcam on the Aspire One laptop. In this article, we are going to use a small Logitech webcam attached to the robot arm. Load up the program called Megabot_Remote1b_DT.exe. This program is the client and it will connect to the server. It has been modified to include the ability to control the arm as shown in Figure 16.

Before doing anything fancy, I really recommend that you test everything out using a robot stand. First, take the IP address shown on the top of the server (bot) form and place it in the Remote IP: field on the client (remote1) form.

Next, click the Start button at the bottom of the form. This will connect the client to the server. Once connected, both forms will enter state 4 — Receive State Pending. In this state, you can issue commands. Finally, you can now hit the buttons to issue the various commands. Try each to make sure they are all working. Once you are satisfied, you can place the Megabot on the floor without the stand. Give yourself plenty of room and start slow.

Once you know everything is working, it's time to set Megabot free. Remove it from the stand and go for it. I have also added a button labeled "Center." This button will center the arm with the camera facing forward. I recommend you use this position when moving Megabot or you could get disoriented. Notice in Figure 16 how the grip is slightly visible. This will allow you to see items as you pick them up.

GOING FURTHER

I didn't get a chance to add any sensors to the project, but it would be a simple matter to add a microcontroller like the Dios Pro. The Dios Pro has a slave library that will allow you to attach it directly to the AX-12 bus as a device. Once done, you can add almost any kind of sensor to help you automate some of Megabot's actions. A sonar range sensor would be really cool and could transmit telemetry back to your Wi-Fi remote program. If you added a sonar sensor to the rear of the robot, you could tell when an animal or person is sneaking up on your bot.

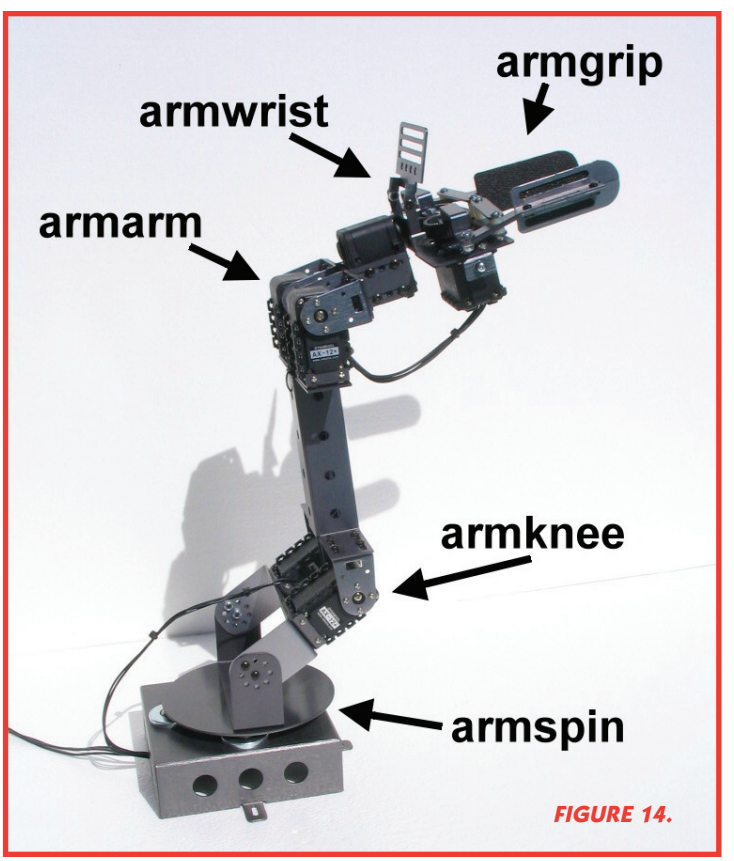


FIGURE 14.

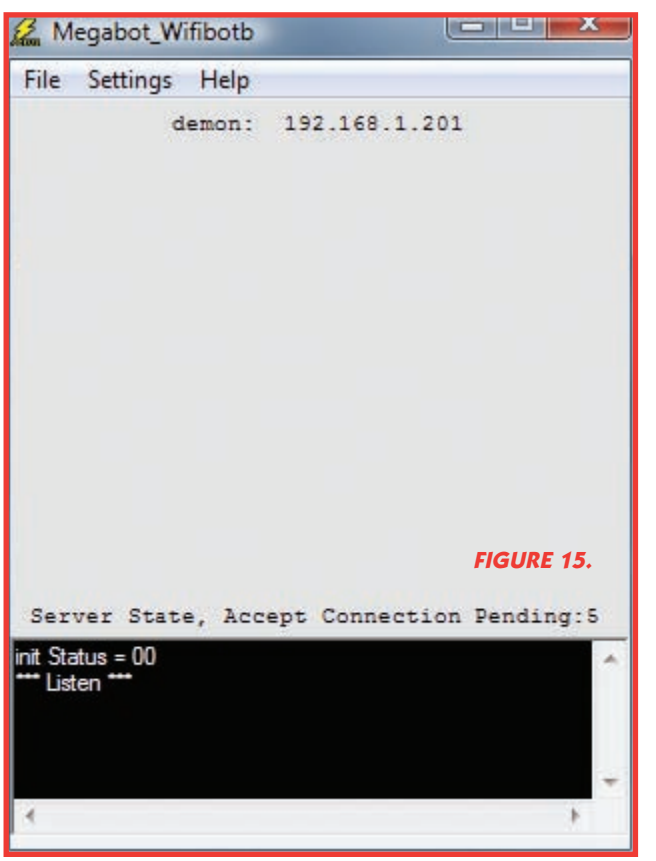


FIGURE 15.

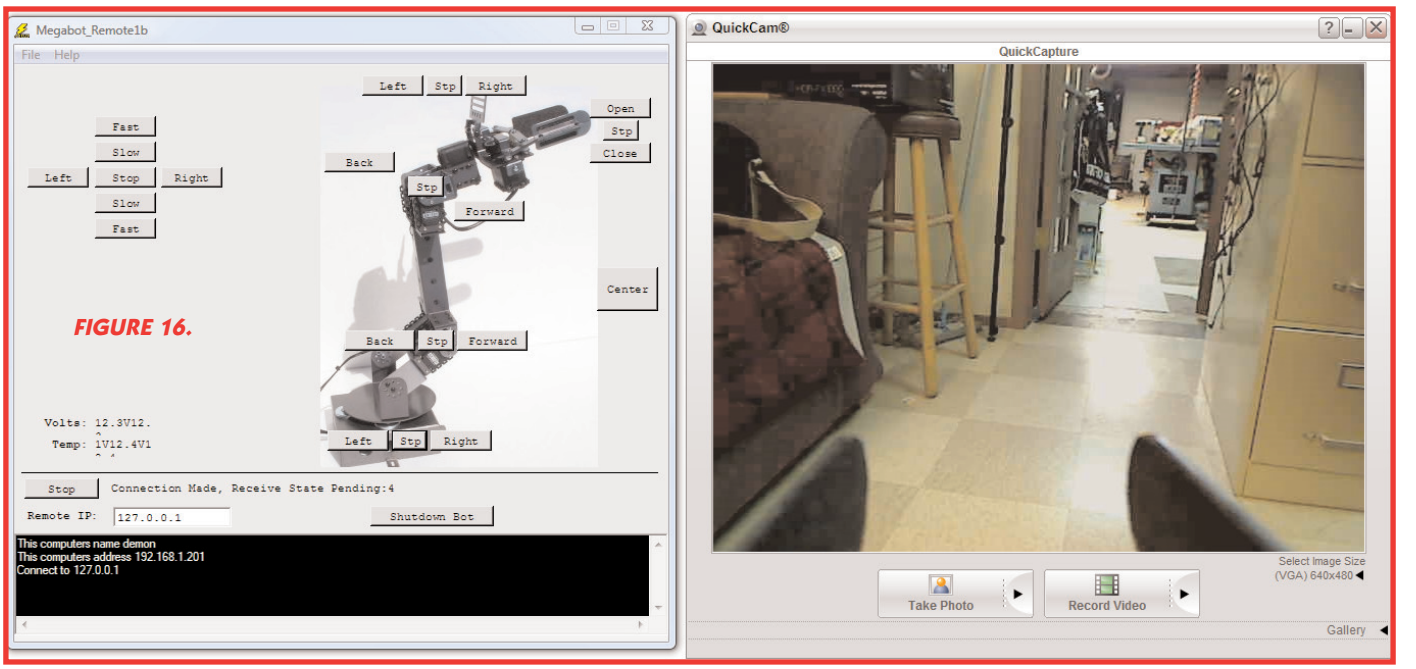


FIGURE 16.

If you added a GPS logger to the robot, you could use the recorded information to help automate navigation. You could place this navigational aid on the laptop, remote, or even both.

I know I had great aspirations when I first started this project. The problem is I just ran out of space and time. That said, enough information has been provided in this series to help you with your own robot projects.

Be sure to check out the Kronos Robotics website for updates, source code, and executables at: **www.kronosrobotics.com/Projects/megabot.shtml** SV

PARTS LIST

Crustcrawler

RX-64
www.crustcrawler.com/motors/RX64/index.php?prod=67

AX-12 Smart Arm
www.crustcrawler.com/products/smartarm/index.php?prod=12

Treaded Wheels
www.crustcrawler.com/products/rover/wheels.php?prod=28

Dynamixel Configurator
www.crustcrawler.com/electronics/USB2Dynamixel/software/Dynamixel_Configurator/DXCONFINST1.2.1.0.exe

Kronos Robotics

ZeusPro Development Environment
www.krmicros.com/Development/ZeusPro/ZeusPro.htm

RadioShack

Four-Position Barrier Strip
#274-658