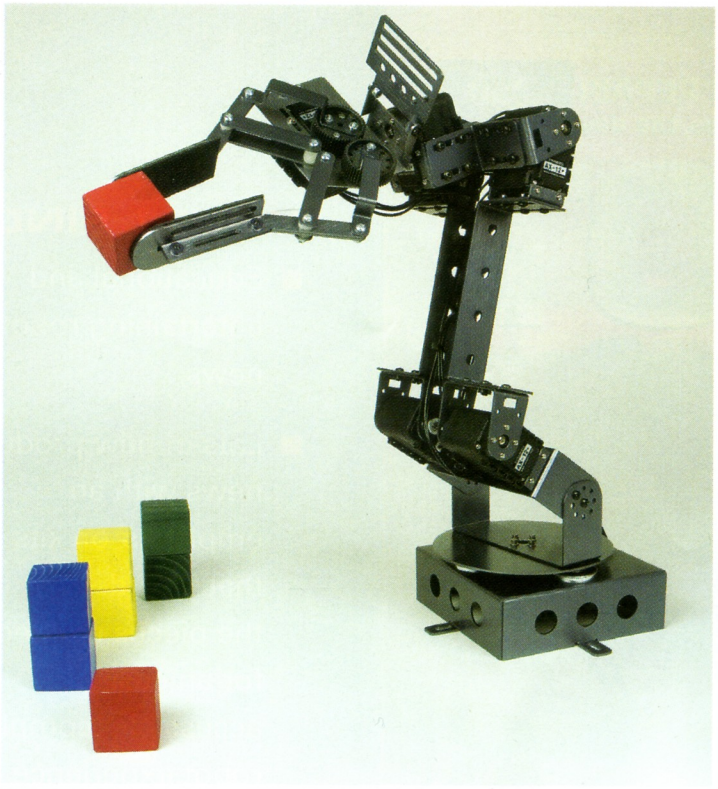
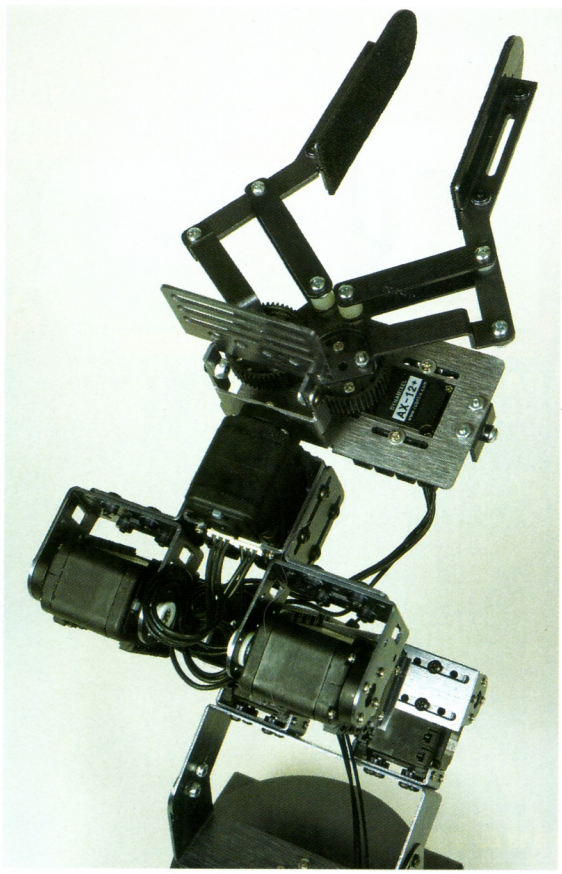




CRUSTCRAWLER AX-12+ Smart Robotic Arm

Powered by the amazing Robotis Dynamixel AX-12 servo

by Steve Norris steven@botmag.com

I built my first robotic arm way back in 1984 as part of the venerable Heathkit Hero-1. Ever since then, I've been interested in its potential applications. When I was asked to review the new CrustCrawler AX-12+ Smart Arm, I jumped at the opportunity. The Smart Arm is unique in its use of a new type of actuator, the Dynamixel AX-12 by Robotis. I've built many applications using servos and stepper motors, so I was curious to see how this new type of actuator would measure up.

THE DYNAMIXEL AX-12

The AX-12 is a robot actuator that contains a precision DC motor, gears and electronics all within a single modular package. It has 300 degrees of movement in 1,024 increments with full control over speed and torque. Unlike traditional RC servos that require each servo to have its own connection to an external PWM controller, the AX-12 has its own internal controller and uses networking functionality to receive commands and send back responses. This greatly reduces the amount of cabling needed because you simply daisy-chain the AX-12 together using three conductor cables (two for power and one for the data).

The network uses half-duplex TTL serial communications configured in a client/server-like architecture. The AX-12 acts as a server receiving commands and returning responses to and from the main application microcontroller or PC. The default baud rate is 1M BPS but can be reduced to allow the use of slower microcontrollers such as the BASIC Stamp. Each AX-12 is assigned a unique ID (more on this later) that identifies it within the network. Using the ID, the main controller addresses and commands the AX-12 by writing to its internal control table. In addition, status and real-time operational information can be read from this table. For the Smart Arm, the AX-12s come from CrustCrawler with their IDs already set.

Having burned out my share of servos, I like that the AX-12 has built-in thermal-shutdown capability and can automatically shut down when heat thresholds are exceeded. With position,

PHOTOS BY STEVE NORRIS

voltage, temperature, speed and load feedback capabilities, it's possible to build applications that can react to overheating, overloading and positioning errors. This is certainly a great advantage over conventional servos.

THE SMART ARM

The Smart Arm is clearly designed to be strong and rugged. It is constructed of anodized 0.063-gauge 5052 brushed finished aluminum components. It uses seven AX-12 actuators to provide 4 degrees of freedom plus the gripper. Four of the AX-12s are paired up for the wrist and shoulder joints, giving each of these joints 440 oz.-in. of torque. The base, wrist rotate and gripper joints use a single AX-12 with 220 oz.-in. of torque. The base has four carbon steel ball bearings to handle heavier loads and an 8-way adjustable angle bracket. The entire arm uses integrated PEM nuts for easy construction. The gripper has a built-in adjustable sensor mount (similar to their S2 mount) for installing cameras or other sensors. It also has slotted gripper paddles for attaching pressure and touch sensors. CrustCrawler also plans to introduce several optional grippers for grasping larger objects.

Depending on the angle slot you choose on the base, the Smart Arm extends up to about 21 inches and the gripper opens up to 2.5 inches. Because of the arm's flexibility and speed, it is necessary to firmly attach the base to a solid foundation. I used a 2-foot-square piece of 0.75-inch plywood to provide a stable base.

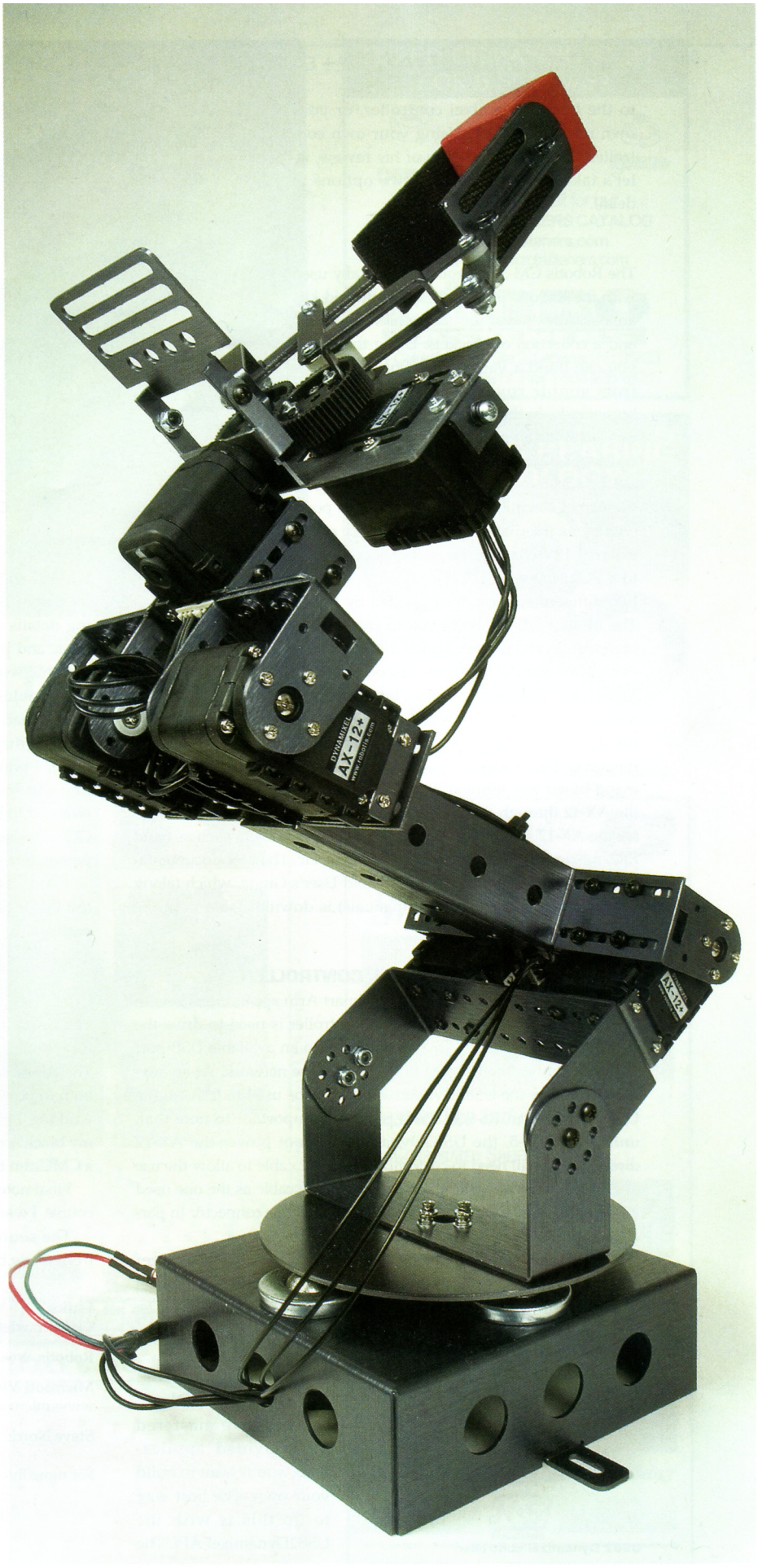
CONSTRUCTION

The Smart Arm comes with a 62-page printed assembly guide. This is a new product, so my guide was version 1. It has clear illustrations and is well written. All the parts are organized into logical assembly units. No parts were missing, but it would have been nice to have a few spares, since I always seem to lose at least one screw to that quantum singularly located under my bench. Loctite is recommended for many of the screw attachment points, so have some on hand before you begin. The trickiest part of the assembly is the gripper, so pay close attention to the text and illustrations. I had one problem with Figure 49, which appeared to be reversed. I reported the issue and got a quick response from CrustCrawler. There are also some good reference pictures at the end of the manual to confirm that you assembled the arm correctly.

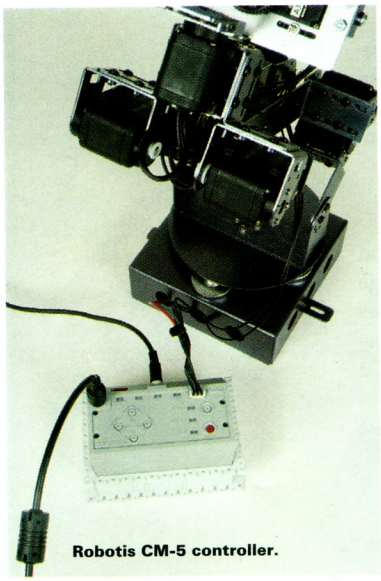
Electrically, the assembly is quite simple. Each AX-12 has two 3-pin connectors. The input cable plugs into one and the output cable, which goes to the next AX-12 in line, plugs into the other. This creates a single daisy-chained cable that runs up the spine of the arm. With just a few zip-ties, you end up with a clean wiring harness.

CONTROLLING THE ARM

You have three ways to control the Smart Arm. You can use the CM-5 Bioloid controller, a PC connected



to the USB2Dynamixel controller, or your own microcontroller. Using your own controller is beyond the scope of his review, so let's take a look at the first two options in detail.



Robotis CM-5 controller.

CM-5

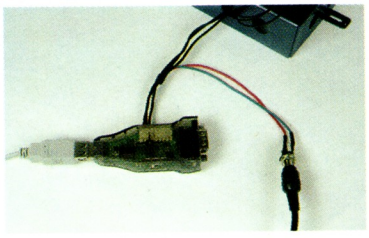
The Robotis CM-5 controller is generally used with the Robotis Bioloid Kits. The Bioloid kits are basically erector sets that consist of AX-12s and a collection of brackets. Using these kits, you can build a variety of robots that range from simple rovers to complex walking humanoids. In this case, we can use the CM-5 to control the Smart Arm since it is also based on the AX-12.

The CM-5 is a self-contained unit based on the Atmel 128 microcontroller. It can be powered by an internal rechargeable battery or the external 12V charger. The CM-5 is connected to a PC using a serial cable. You can interact with the CM-5 using two different applications provided by Robotis that run on the PC. The Motion Editor allows you to create a sequence of AX-12 commands that can be downloaded into the CM-5 and then played back even after the PC has been disconnected. Although I found the Motion Editor's user interface a bit cumbersome, you can easily create sequences by manually positioning the Smart Arm and then capturing the positions of the AX-12s and saving them in the CM-5's memory. The second application, the Robot Terminal, is a command-based text terminal that allows you to interact directly with the AX-12 through the CM-5. This is the application that you use to set the AX-12's unique ID and to set other parameters such as baud rate, maximum torque, temperature range, etc. The best documentation to use with the CM-5 is the Bioloid User's Guide, which (along with all the other Smart Arm manuals) is downloadable from the CrustCrawler site.

USB2DYNAMIXEL CONTROLLER

The best way to build PC controlled Smart Arm applications is with the USB2Dynamixel controller. This controller is used to drive the AX-12 directly from a PC by connecting it to an available USB port and then connecting it to the first AX-12 in the network. As an auxiliary function, the USB2Dynamixel can also be used to transform a USB port into an RS-232 serial port. It is important to note that, unlike the CM-5, the USB2Dynamixel cannot power the AX-12 directly. You will need to customize an AX-12 cable to allow the use of a 7- to 10V power adapter. I used the same cable as the one used for the CM-5 and simply soldered a 2.1mm power connector in parallel with the power lines.

The USB2Dynamixel is supported by the Dynamixel Manager software. Like the Robot Terminal for the CM-5, the Dynamixel Manager allows you to set and display parameters in the AX-12s. It is a Windows GUI-based interface instead of the text-based for the CM-5.



USB2 Dynamixel controller.

THE .NET API

Once you've mastered the provided applications, you'll want to build your own. The best way to do this is with the USB2Dynamixel API. The API uses Microsoft .NET technology. The VB.NET source code is available free on the CrustCrawler site. It was developed by Mike Gebhard, and the source code is fully commented and compatible with Microsoft's free Visual Studio Express. You can use the API with your favorite .NET language such as VB.NET, C#, C++ or J#. I used the 2005 version of the Visual Studio Express. As I write this, Microsoft has just released the 2008 version. I have not yet tested the API code with this version.

Side note: Mike Gebhard is also working on a Spin object library for the Parallax Propeller chip. A beta version should be obtainable soon. Check the CrustCrawler site for availability.

MY APPLICATION

To test the API and get my Smart Arm up and running, I developed a couple of simple VB.NET applications that control Smart Arm. For reusability, I built a Smart Arm wrapper object that encapsulates the USB2Dynamixel API and hides all the low-level interfacing details of the arm. It provides functions to move each of the joints and properties to access the AX-12 parameters. As an example, this wrapper object hides and handles the fact that the wrist and shoulder joints have dual AX-12s that face each other. This means that any requested position must be transposed for one of them or they will fight each other and cause an over-torque shutdown. I bring this up because it took some debugging before I realized this simple mechanical fact. Having debugged it, I used this wrapper to build the Smart Arm Executive application. It has a full GUI interface that allows you to move the arm and read position parameters.

I also built a simple test program that is used to stack blocks and test the dexterity and repeatability of the arm. I must say that I was particularly impressed by the arm's speed, smoothness and repeatable position accuracy as it moved the blocks around.

CONCLUSIONS

The CrustCrawler AX-12+ Smart Arm is a sophisticated and powerful robotic arm and a welcome addition to my collection of robots. The AX-12 actuators have proven to be superior to standard servos both in power and accuracy. Now that I have the arm and software working, I plan to start a few more projects. I will probably duplicate my block sorter application and also mate this arm with a rover and a CMUcam to create an interesting find-and-retrieve application.

Final note: way back in 1984 when I built the Heathkit Hero-1, of course, I was very, very young.

The source code for my VB applications can be downloaded from www.botmag.com/issue11. ◎

Links

CrustCrawler, www.crustcrawler.com, (480) 577-5557

Robotis, www.robotis.com

Microsoft Visual Studio Express 2005, www.microsoft.com/express/2005/

Steve Norris website, www.norrislabs.com

For more information, please see our source guide on page 89.