ROBYTES • ROBOTICS RESOURCES • ROBOTICS ORGANIZATIONS

SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

www.servomagazine.com

MARCH 2007

Bring Your Robot to Life With iRobot Create

• VEX Coming to A School Near You

A Conversation With Dean Kamen

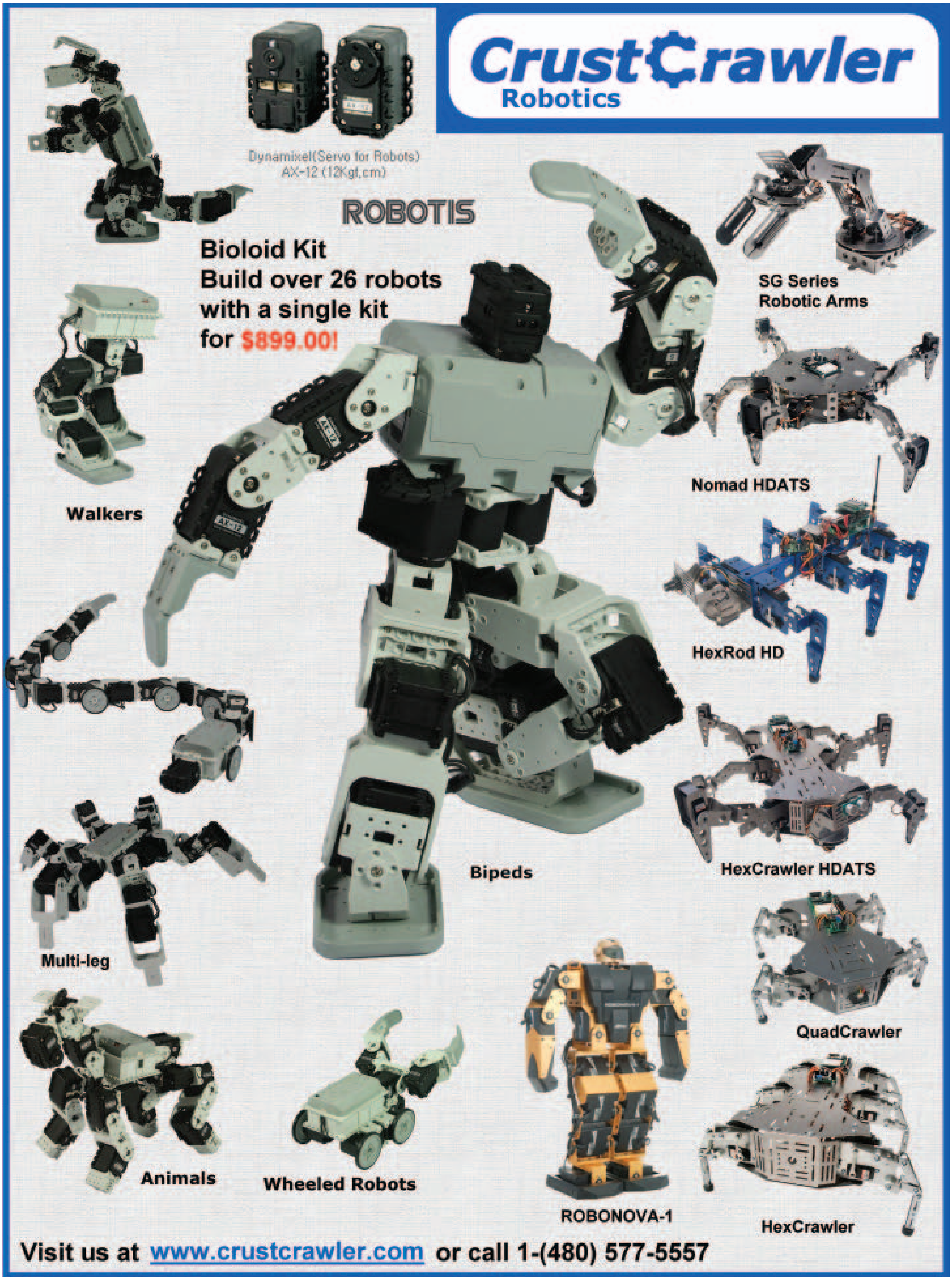
• Seeing With OpenCV

Part 3: Follow That Face!

• Smart Servo:

Understanding The Robotis AX-12

U.S. $5.50 CANADA $7.00



CrustCrawler
Robotics
Dynamixel(Servo for Robots)
AX-12 (12Kgf.cm)
ROBOTIS
Bioloid Kit
Build over 26 robots
with a single kit
for $899.00!
SG Series
Robotic Arms
Walkers
Nomad HDATS
HexRod HD
HexCrawler HDATS
Bipeds
Multi-leg
QuadCrawler
Animals
Wheeled Robots
ROBONOVA-1
HexCrawler
Visit us at www.crustcrawler.com or call 1-(480) 577-5557

SERVO MAGAZINE

Features & Projects

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

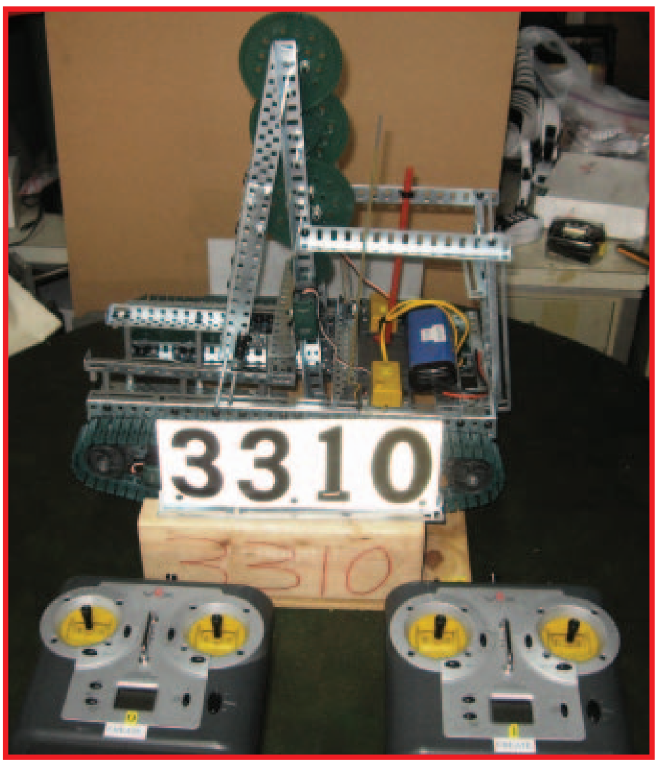
03.2007
VOL. 5 NO. 3

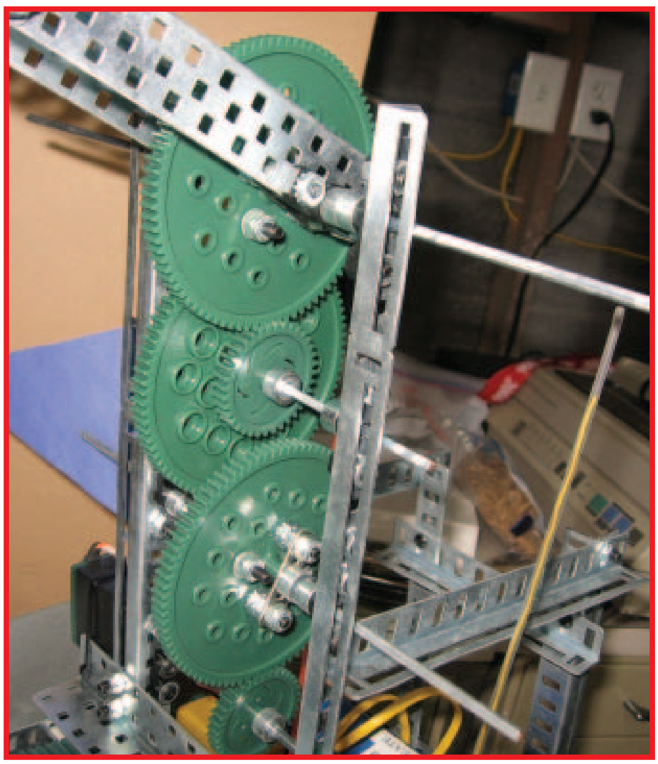
Columns

Departments

ENTER WITH CAUTION!

Page 10





Mind / Iron

by *John Koepke*

Acting as a chaperone for my oldest son during one of his field trips in eighth grade, I had come upon the LEGO Mindstorm System. The Chicago Museum of Science and Industry was hosting a seminar on robotics using this, then new, LEGO Mindstorm System. I was immediately captivated with the genius of this robotic system. A couple of years later, I made a career change to teaching at Joliet Junior College where I immediately employed the LEGO Mindstorm System to teach digital electronics and hold summer seminars for junior high and high school students.

I began to reflect back on some of the competitions I had participated in years earlier regarding robotics and thought these robotic kits would provide a great platform for such a competition. My initial desire was to ensure high schools participating in the competition would not have to purchase these LEGO kits or software in order to participate in the event. Nor did I want teams to have an advantage over others through differences in sponsorship or school funding levels. Teams would arrive with nothing more than something to write with and a calculator. I have been fortunate that Joliet Junior College has supported these goals, as well. Now in our seventh year, The Robotic Engineering Challenge is still free of an entry fee.

The main goal of the Robotic Engineering Challenge (REC) series is to expose students to the engineering and technical fields. REC provides students with real time, hands-on experience acting in the role of an engineer or technician. The LEGO ROBOLAB™ system provided the framework to bring the engineering experience to each student. Each LEGO ROBOLAB system contains a microprocessor-based smart brick that uses sensor information to complete assigned tasks.

The competing student teams must find the right balance between hardware and software interfacing. We can view the software as 'how we want the robot to react' to its changing environment; how it is to think. The software is basically the 'brains' of the robot. The hardware makes up the framework for executing the demands of the program. Much like how our body reacts to our thoughts. The robots that the students build to meet the most challenges are autonomous, that is, the robots think for themselves; no human in the control loop once the run button is pressed.

There are two levels of competition: Expert and Novice. The Novice group receives their challenges in sequence. Their faculty sponsor may assist them. The Expert group performs without the help of their faculty advisor. They receive all challenge tasking at once and need to decide which challenge to tackle first. Both groups are under time constraints to complete as many challenges as possible during the competition period. The expert and novice challenges are shown below. Each challenge also requires the students to answer a set of math, physics, or science questions related to a particular challenge.

EXPERT:

1) Square Route: Follow a square path.
2) Land Navigation: Move across a bridge and return back to the start position.
3) Search-and-Rescue Robot: Deliver a medical supply payload across a vertical barrier.
4) Bomb Disposal Robot: Pick and place a simulated bomb into a bomb dumpster.
5) Hazmat Robot: Clear a building of a simulated hazardous waste container.

Mind/Iron Continued

Published Monthly By
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Product Order Line **1-800-783-4624**
www.servomagazine.com

Subscriptions
Inside US **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

CONTRIBUTING EDITORS
Jeff Eckert, Tom Carroll, Gordon McComb, David Geer, Pete Miles, R. Steven Rainwater, Ted Larson, Bryan Bergeron, David Calkins, Kevin Berry, Bryce Woolley, Robin Hewitt, Peter Best, Evan Woolley, Eddy Wright, John Jellman, Michael Simpson, Helen Greiner, John Koepke, Russ Barrow, Chad New, Christopher Gilleski, Tim Wolter, Robert Wilburn, Dan Kara, Paul Reese

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux
Michele Durant

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

NOVICE:
1). Binary 500: Complete one lap around a small track.
2). Move through a small maze.
3). Sumo-Bot: Survive against the house robot.
4) Pool Table: The student robot acts as a cue ball clearing the pool table of balls.
5) House Navigation: Seek and find point-valued cones placed throughout a scaled-down house layout.

During the Jan. 2007 competition, we had 31 high school teams (over 130 students) participate.

Hopefully, competitions like the REC will interest students in engineering. I feel students are not going into the engineering field because many may feel "over-challenged" by the math requirement. Mathematics is the language of technology and seeing a practical application of the subject makes it more understandable. Most likely, however, students are not exposed to the engineering or technical subject areas until later in the educational process, if at all. It is hard to make a career choice regarding an area that you may know very little about. It is easier to select a career by relying on your daily experiences and impressions of certain jobs. For instance, most people have some impression of the field of nursing because of their interaction with a nurse when visiting the doctor or hospital. Typically, we don't have daily exposure to engineers (directly that is). However, everything around us is a product of the design work of an engineer.

I hope in some way that students will gain some insight into the field of engineering from this competition and spark an interest to study this subject area. I was told this competition is a form of 'engaged learning' — I want students to get excited about engineering, math, and science. When I watch the students competing, I would agreed that they are totally engaged in learning and problem solving. **SV**

BIO-FEEDBACK

Dear SERVO:

In the January 2007 OpenCV article, there may be an error in the source code listing in Figure 3, page 64. I haven't built the OpenCV library yet, so I am not positive, but line 14 may not be correct. As presented in the article, the cvLoadImage() function is called without assigning the pointer to the returned value. The following test will always be true in this case and the program will always return an error message. It should probably read:

```
pInpImg = cvLoadImage("my_image.jpg",
CV_LOAD_IMAGE_UNCHANGED);
```

Other than that minor problem, the article is very informative. I'd run across OpenCV before, but had been intimidated by the scope of it. The article has convinced me to download it and give it a shot. I'm looking forward to the next installment.

Robert Wood

You're right! Thank you for pointing that out. It's great to hear that the January article motivated you to try out OpenCV. — Robin Hewitt

Robytes

by Jeff Eckert

Are you an avid Internet surfer who came across something cool that we all need to see? Are you on an interesting R&D group and want to share what you're developing? Then send me an email! To submit related press releases and news items, please visit www.jkeckert.com

— Jeff Eckert

ROV Explores Antarctic Floor

The Isis ROV, hanging from its launch and recovery system. Photo courtesy of UK Deep Submergence ROV Facility.

By the time you read this, the Isis explorer — operated by the UK's Deep Submergence ROV Facility — will have completed one of four scheduled plunges to the Antarctic sea bed in an effort to gather information about the effects of glaciers on the ocean floor and to prod whatever animals live there. While there's nothing revolutionary about ROVs, this is reportedly the first time one has been sent into this environment, and it carries an impressive slew of lights, cameras, sonars, and robotic arms for collecting samples and implanting instruments.

Based on the Jason ROV designed at our own Woods Hole Oceanographic Institution (**www.whoi.edu**), the Isis dangles from the mother ship by 10 km (*Wow! - ed.*) of cable so scientists can control and communicate with the vehicle in real time. It can travel under its own power, being fitted with six 3.7 kW thrusters, but its 1.5 knot velocity is not exactly breathtaking. On the other hand, any movement at all is pretty good for something that weighs 3,250 kg (7,165 lb) and has the aerodynamic qualities of a punch press. For updated information, drop in at **www.noc.soton.ac.uk/OED/ROV**.

Bug Joins NATO

The CyberBug UAV recently participated in NATO and other military exercises. Photo courtesy of Cyber Defense Systems, Inc.

Even though it doesn't look like much, the CyberBug™ from Cyber Defense Systems (**www.cduav.com**) is a pretty useful little UAV that is designed for military, law enforcement, and commercial applications. (It would be fun to have one for personal pleasures such as buzzing the neighbors' backyard barbecues, but the $10,000 base price makes that a little impractical.) According to the company, the birds can be assembled in just a few minutes and launched right away to provide aerial surveillance. They can fly for up to an hour while sending video and other data to the ground station, and common tasks include monitoring hazardous events, search and rescue, traffic monitoring, border patrol, and so on.

Late last year, three of them were employed to support navigation warfare scenarios during Trial "Spartan Hammer," a 12 nation NATO collaborative effort conducted in western Greece. A variety of sensor payload configurations were used to collect information used to support signals intelligence and electronic warfare campaigns. Reportedly, a total of 22 sorties were flown with a 100% mission completion rate.

Three sizes are available: the micro (2.5 lb), medium (8.5 lb), and large (14.5 lb). A 42 lb version is under development that will fly with a 12 lb payload for up to two hours, or up to eight hours without the baggage.

A Real Mean Machine

The TALON robot comes in various lethal versions. Photo courtesy of Foster-Miller, Inc.

In case you haven't noticed, Asimov's Laws of Robotics have become pretty antiquated, and no more so than in the case of the TALON® robot from Foster-Miller (**www.foster-miller.com**). The company builds mobile platforms for military, HAZMAT, and SWAT applications, and this model can be configured for different sizes and functions, including stealthy reconnaissance, intruder attack, underwater surveillance, and remote sensing/monitoring.The basic unit is designed to be compatible with a range of hardware, weapons, and sensor systems. More than 80 different payloads have been developed or adapted and mounted on it, including smoke and grenade dropping modules, anti-tank launchers, a 40 mm grenade launcher, and a 12 gauge shotgun.

You can also get mounts for remotely-controlled weapons including the M240 and M249 machine guns, the M16 rifle, and the M82A1 .50 cal. anti-tank/anti-material rifle. A smaller version is available for stealthy reconnaissance operations, and there is even an underwater model.

At present, this is the only mobile platform certified by the Department of Defense for remotely-controlled live firing of lethal weapons, so you probably won't encounter anything meaner.

This One Will Touch Your Heart

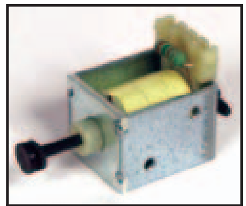
The Sensei Robotic Catheter Manipulator. Photo courtesy of Hansen Medical.

Possibly more disturbing to contemplate than the TALON — for entirely different reasons — is the Sensei™ Robotic Catheter System from Hansen Medical, Inc. (**www.hansenmedical.com**). The company recently completed a 20-patient trial as part of 510(k) premarket submission to the US Food & Drug Administration. The trial was an observational study that used the Sensei system to guide catheters into the heart for mapping heart anatomy.

The system is designed to provide accurate and stable control of catheter movement in 3D during cardiac electrophysiology procedures. Currently, these procedures are performed using a manual technique that requires physicians to perform complex manipulations at one end of the catheter with inadequate assurance that the inserted tip of the catheter will respond as desired.

The Sensei system consists of three components: the physician workstation, the "instinctive motion controller," and the robotic catheter manipulator (see photo). The gear may look a bit scary, but Hansen says it will "enhance the ease of use and stability of catheter-based procedures by offering physicians better control over catheter placement, as well as potentially decrease procedure times and radiation exposure."

Solenoid Features Quiet Operation



The MagShift solenoids provide power-on noise levels <40 dBA. Photo courtesy of Saia-Burgess.

Returning now to an item that readers might actually find to be of practical use, Saia-Burgess, Inc. (**www.saia-burgess.com**), has introduced the MagShift® line of solenoids to provide actuation in a variety of applications. It is designed to eliminate impact among its moving parts, which results in a noise measurement below 40 dBA, including the end-of-travel stop. There is no impact force at the end-of-travel position, so vibration and noise are minimized. As a result of the elimination of internal components, the MagShift design eliminates residual magnetism and can be configured for either push or pull operation. **SV**

The FIRST (For Inspiration and Recognition of Science and Technology) organization founded by Dean Kamen offers robotics competitions for budding roboticists of all ages and skill levels. Elementary school students can take their first steps into the arenas of science and technology with the Junior FIRST LEGO League. Middle school students can make forays into the exciting world of robotics with the FLL, the FIRST LEGO League. High school students are faced with the challenge of the FIRST Robotics Competition (FRC). But jumping into the FRC is a huge step. Students go from robots they can hold in their hands to robots that are as big as they are. This can be quite an intimidating step up, especially because many teams do not even have the benefit of the FLL as a stepping stone. That's where the Vex Challenge comes in.

The Vex Challenge

The Vex Robotics Design System made its debut a few short years ago, and late 2006 witnessed the first official full-scale Vex Challenge competition. A pilot tournament of the Vex Challenge was held at the 2006 FRC Championships, but the 2006-2007 season witnessed an expansion into a complete competition with regional events and a championship. Regional events across the nation are being held from December 2007 until spring 2007, and the championship is to coincide with the FRC National Championship in April 2007.

The Vex Challenge is a robotics competition that aims to inspire and prepare teams for the big robots of the FRC. The Vex robots, arguably closer to "real" robots than the LEGO Mindstorms kit used in the FLL, are intended as stepping stones for FLL alumni or absolute rookies that feel they need a little more experience before tackling the huge commitments and challenges of a FIRST team.

The Vex Challenge is similar to the FRC in that the game emphasizes cooperation and task completion. Vex Challenge teams also get a kit of parts to work with — as you might have guessed, the Vex kit itself. Teams can buy more than one kit, but they are limited to a certain number of motors and controllers, and they can only add non-Vex parts to their robot if they are exact substitutes of the part in question (like if you needed more collars but didn't want to buy another whole kit to get them). There is no weight restriction for the robots, but the stringent dimensional limit of having to fit into a cube 18 inches to a side ensured a challenge.

One of the most apparent divergences that the Vex Challenge has

THE COMPETITION.

Photo courtesy of Trey Amador.

FVC IN ACTION.

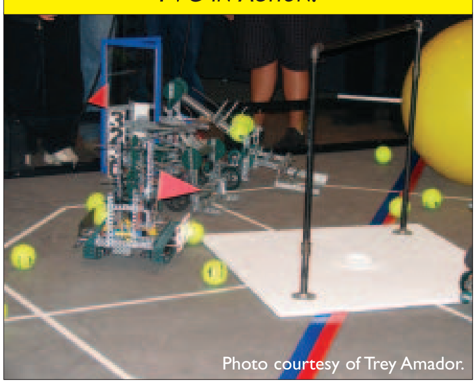


Photo courtesy of Trey Amador.

from the FRC format is that the build time is not a notorious six weeks. For the 2006 Vex Challenge, the game was revealed in September, and the first regional competitions didn't take place until December. This vague build schedule might become more draconian in later years, but the generous timing was much appreciated this time around seeing as how almost every team was essentially a rookie. All of these concessions might lead one to believe that the FIRST Vex Challenge game might be a simple one, but instead, it was most certainly a challenge.

Do You Want to Play a Game?

The 2006-2007 Vex Challenge game was Hangin'-A-Round, and it featured many challenges that would be familiar to veteran FIRST teams. Two teams of two faced on a surprisingly large field measuring 12 feet by 12 feet. In the middle of the field there was a freely rotating platform, and featured prominently there was a tall (33 inches) pull-up bar. At the corners of the field there were two "low goals," and along the sides of the field there were "high goals" in the shape of triangles. Around the field there were pyramids of softballs, and last but not least, directly below the pull-up bar in the middle of the central platform, there was the "atlas ball." The atlas ball was a very large and very yellow inflatable yoga ball, measuring 30 inches in diameter.

Softballs were the main scorable item in this game, and they were worth one point when pushed into the low goals, and three points when dropped in the high goals. Teams were awarded five points for each robot on the central platform at the end of the match, and 15 for each robot hanging from the pull-up bar. Whichever team was lucky enough to have the atlas ball at the end of the match would have their points from scored softballs doubled.

As with the newer FIRST games, the matches started with a period of autonomous play, in this case, 20 seconds. Alliances started with three softballs between the two robots, with a robot only allowed a maximum of two softballs on a robot. As a special incentive to do well in autonomous mode, the team with the most points at the end of the 20 seconds is awarded an extra 10 points.

As Easy As A, B, Easy C

Since there is an autonomous mode, it follows that there must be a way to program the Vex robots for the competition, and indeed there is. Easy C was first introduced as a way to program the big robots of the FRC in 2006, and it is used for the Vex Challenge, as well.

Easy C is an object-oriented version of C that has simple drag and drop commands for constructing a program. It contains all of the commands you need for an effective autonomous program, and Easy C has several features that make it a great way to learn programming. As users create a program with dragged and dropped blocks, the program is simultaneously written out in the actual syntax of C. Even the most inexperienced of programmers can be coding in a matter of minutes with the accessible interface.

Novices can do much more than just drag and drop the C commands. Being able to see the actual syntax of the C program helps new users understand what a command does and how it is actually written out. Inquisitive Vex challengers should be able to figure out what all those semicolons, ifs, and for loops mean in no time, and they will be well equipped to take on programming classes in college when they pursue a degree in engineering.

Atlas Ball Shrugged

The FIRST Vex Challenge was certainly designed with newcomers in mind, but it also does a great service for existing FIRST teams. The hectic build season of FIRST season may only last six weeks, and the competition season a few months more, but FIRST is really a year round commitment. The Vex Challenge, by preceding the FRC by a few months, is a way to engage team members that might have become bored with fundraising or other inter-season activities. Also, as senior members of FIRST teams graduate and head off to college each year, robot rookies come in to take their place. The FVC is a great way to introduce rookie members of established FIRST teams to robotics, and it is still a great way for veteran FIRST participants to continue their education in robotics. That's why our old FIRST team — Team 1079 — jumped at the chance to get involved with the FVC. All of the founding members of the team had gone onto to college, and the new team members needed a solid foundation in robotics. So Team 3310 was formed, ready to take on the challenge of Hangin'-A-Round.

The design process for the Vex Challenge was just as frenetic and energetic as that of the FRC. After an introduction to the Vex kit, the team members were brainstorming and building prototypes. Rapid prototyping was a

FVC FIELD.

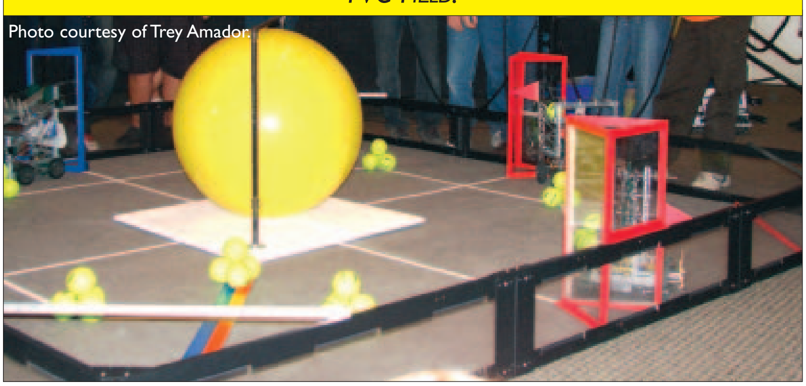
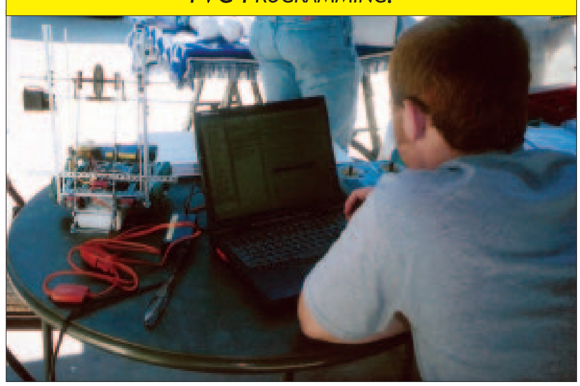


Photo courtesy of Trey Amador.

FVC PROGRAMMING.



great way for the students to test their ideas, and such tangible experimentation is difficult to do on the larger scale of FIRST robots. The limited materials, difficult construction, and harsh time limit of FIRST make extensive experimentation difficult and largely unrealistic for smaller teams. Trial and error is a great way to test ideas, but the fast pace of the FRC doesn't allow for too much of either.

The generous time limit of the FVC and the simple construction and deconstruction of the Vex kit make prototyping a very realistic pursuit, and building test devices is much more appealing to most people than Newton's Laws and number crunching. Those are important techniques as well, but Vex teams will a get a healthy portion of both when they tackle the FRC.

The game was difficult, and it seemed excessively ambitious to design a robot to do everything. The team members brainstormed to decide what tasks had the greatest point potential and what seemed like the best strategy, and it was decided that hanging on the pull-up bar was too difficult for just 15 points. Instead, Team 3310 chose to concentrate on building a maneuverable robot that could provide an effective autonomous mode, easily manipulate the softballs, and control the atlas ball.

When You Give a Team a Robot Kit

The Vex starter kit was a great launch pad for building a robot capable of tackling the Vex Challenge, but Team 3310 wanted to do more than the simple starter kit would allow. Thankfully, a wide range of expansion kits are available to FVC teams, and the budding roboteers of Team 3310 were immediately drawn to the treads kit. The treads expansion kit certainly had a high cool factor, but it also fit the bill for the desired qualities of maneuverability and traction that the team wanted in their bot.

The first order of business was to construct a solid drive train. The highly adjustable nature of the Vex kit came in handy when determining the proper gear train and placement of critical bits, and soon the team had a platform able to support the other necessary mechanisms. The main mechanism would be an arm capable of scoring softballs in the top goal, which was no small feat considering the disparity between the height restriction of the robot (18 inches) and the height of the goal (two feet).

Some initial forays into building test mechanisms revealed that the Vex motors are stronger than they look, even with only a few stages of reduction. Durability and reliability were also critical issues, though, so the mechanism would most definitely benefit from some re-engineering. After some finessing of the gear train, Team 3310 had a robot that was ready to compete. As long as it was under human control, that is.

The team wanted to finish the mechanical skeleton of the robot as soon as possible so they had plenty of time to create a great autonomous mode. While many of the robot team members from 1079 were hesitant to tackle the challenge of regular C syntax, Easy C was much less intimidating and many more team members were able to be a part of the programming process.

The Vex kit comes with a variety of sensors, and lines that spanned the playing field encouraged strategies like line following. The roboticists of Team 3310, however, subscribed to the philosophy of KISS, so they concentrated on finding the simplest way to score as many softballs in the high goals as possible during the autonomous mode. By far, the simplest strategy seemed to be to start directly adjacent to one of the high goals and to begin the match with two softballs on the robot. All the robot would need to do would be to lift its arm and deposit the balls for a cool six points.

Build for Today, Learn for Tomorrow

The importance of the Vex Challenge goes far beyond the FIRST organization, and people like Dr. Ralph Mills of the Small Manufacturers Institute recognize that. Dr. Mills and the folks at SMI know that events like the FVC are training and inspiring the next generation of engineers, so they sponsored several workshops and even an unofficial competition at Glendale Community College.

The Glendale competition was a great way for teams to test their robots before the official regional event, and Team 3310 jumped at the opportunity to give MO Jr. a chance to play. The promise of practice also drew many other teams, and the Glendale competition drew the phenomenal crowd of 44 teams. Thanks to some expert planning, a couple official FVC fields, and the monumental volunteer efforts of FIRST teams like Team 599, the Glendale competition proved that the Vex Challenge was well on its way to becoming a mainstay of the FIRST organization. And despite the tinier robots, the matches were just as action-packed and exciting as any FIRST Competition. The variety of robots and strategies was as diverse as ever.

Even though there is no wrong way to play a Vex Challenge game, dominant strategies and robots did arise. The atlas ball really seemed to be the deciding factor in many matches, and Team 3310's hard work on a solid drive train really paid off. Even though

MO JR.

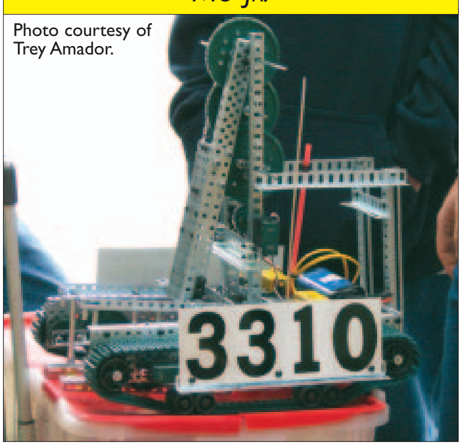
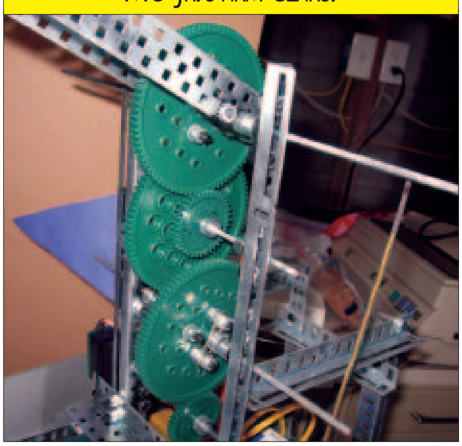


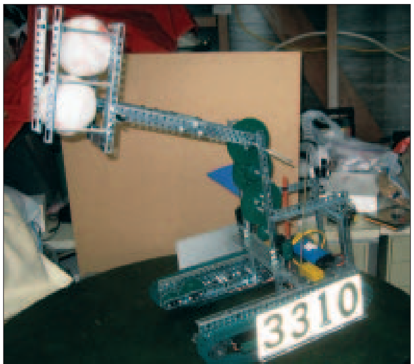
Photo courtesy of Trey Amador.

MO JR.'S ARM GEARS.



MO Jr. did not have the ability to hang, many other robots rose to the challenge to prove that even mechanical arms can benefit from a few pull-ups.

A reliable autonomous mode is always a winning attribute in the FRC, and such was also the case for the FVC. Team 3310's autonomous mode turned out to be reliable and high scoring, and coupled with MO Jr.'s heavy duty drive train and aggressive pursuit of the atlas ball, it was a winning combination. Team 3310 eventually placed 2nd in the Glendale competition, and the team was ready to tackle the official regional event.



MO JR. LIFTING.

NICK PRICE, STEVEN KESSLER, AND RYAN POTTS.

The Thrill of Competition

The Los Angeles Regional competition was held at California State University, Northridge in mid-December. One might expect that with such small robots that the energy and excitement of the competition might be scaled down accordingly, but such was not the case. The spirit of camaraderie and gracious professionalism is also very much intact, and teams are just as interested in helping out and admiring other teams as in the FRC.

High production values, spot on scheduling, and fun times galore are all hallmarks of FRC events, and the regional FVC event had all of them in abundance. About 40 teams attended the regional event, and while many teams were formed from established FIRST Teams, there were also a considerable number of teams that were using the Vex Challenge as their first experience in robotics.

With multiple fields set up, the action at the regional competition was nonstop. Many of the same strategies from the Glendale competition resurfaced at the regional, with the atlas ball almost always playing a deciding role in the matches. Competition over the atlas ball was fierce, and Team 3310 was part of the match with the interesting distinction of being the only one where the big yellow atlas ball was actually ejected from the field. Unexpected indeed, but nobody gets points for an ejected atlas ball.

MO Jr.'s effective autonomous mode and obsession with the atlas ball once again served Team 3310 very well in competition, eventually landing the team in 3rd place.

Portal to Education and Beyond

The goal of the FIRST Vex Challenge is exactly that of the FRC — to inspire students to pursue an education and a career in science and technology. While the FRC may provide an experience that is much more like what the students would face as engineers with the strict deadlines and complicated kit of parts, the FVC certainly fulfills an important role of its own.

The FRC is a huge undertaking — teams have to raise a significant sum of money that goes beyond the $6,000 entry fee to include extra parts and transportation to competitions. Teams need to be super organized to finish such a big robot in six weeks, they need dedicated mentors to teach them the ins and outs of complicated robot design, and they need to be diligent spokespeople all year round to retain the interest of sponsors and team members. Of course, all of these things are great skills that are important for the students to learn, but it can also be terribly intimidating for prospective teams with meager resources.

The Vex Challenge is a lot more manageable — a sensible budget for a team that includes the registration fee, extra kits, and a few travel expenses can land at about $1,000. This would still demand some fundraising, but it would be more along the lines of a few school fundraising events and some local sponsors, not giant corporations that can be daunting to approach. The smaller, simpler kit with accompanying curriculum is also a lot more accessible to an inexperienced team that might not necessarily have an engineering mentor.

The generous time limit also gives teams time to experiment, and it allows them a little slack if they aren't run like well-oiled machines. The smaller size of the robot makes transportation a snap — teams can put their robot in the trunk or seat of a car instead of having to build or procure a crate for shipping. And even with all the scaling down, the Vex Challenge doesn't lose any of the major lessons of the FRC.

Students learn how to work as a team to solve a difficult problem. They learn the engineering process that involves brainstorming, risk reduction, construction, and revision. And the Vex Challenge events are just as much havens of excitement, sportsmanship, and gracious professionalism as any FRC event.

The FIRST Vex Challenge turned out to be a great experience for Team 3310. Doing well in competition is always nice, but the knowledge and inspiration taken away from the competition means much more than any medal or award.

The FIRST Vex challenge is sure to become a staple of the FIRST organization; it is an accessible competition that imparts inspiration to its participants that is not at all diminished by the smaller size of the robots. SV

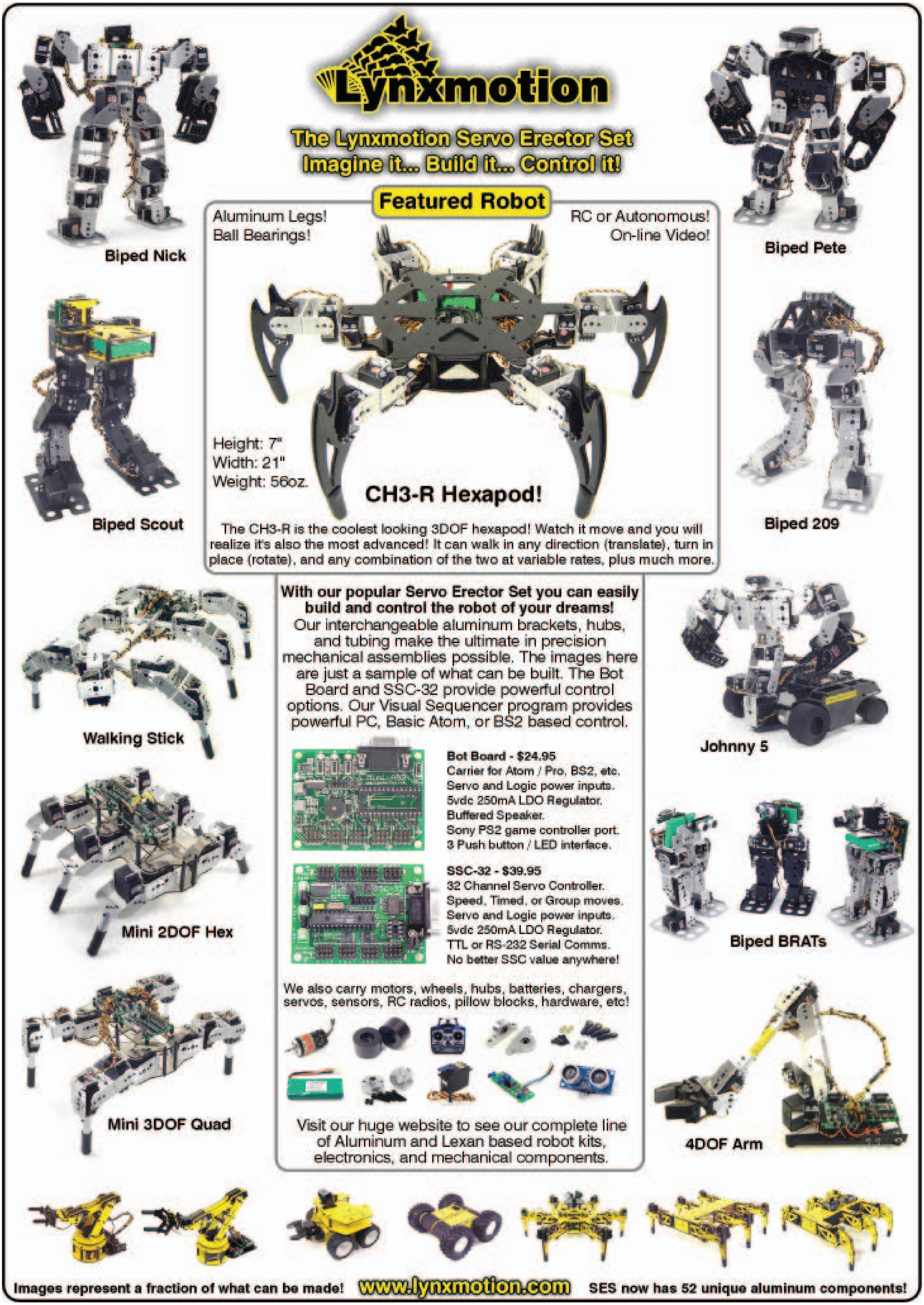
RECOMMENDED WEBSITES

For more information, go to:

www.usfirst.org

www.vexlabs.com

Rhode Island Science and Technology Advisory Council
www.stac.ri.gov


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Featured Robot
Aluminum Legs!
Ball Bearings!
RC or Autonomous!
On-line Video!
Height: 7"
Width: 21"
Weight: 56oz.
CH3-R Hexapod!
The CH3-R is the coolest looking 3DOF hexapod! Watch it move and you will realize it's also the most advanced! It can walk in any direction (translate), turn in place (rotate), and any combination of the two at variable rates, plus much more.
Biped Nick
Biped Pete
Biped Scout
Biped 209
With our popular Servo Erector Set you can easily build and control the robot of your dreams!
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible. The images here are just a sample of what can be built. The Bot Board and SSC-32 provide powerful control options. Our Visual Sequencer program provides powerful PC, Basic Atom, or BS2 based control.
Walking Stick
Johnny 5
Bot Board - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!
Mini 2DOF Hex
Biped BRATs
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Mini 3DOF Quad
4DOF Arm
Visit our huge website to see our complete line of Aluminum and Lexan based robot kits, electronics, and mechanical components.
Images represent a fraction of what can be made!
www.lynxmotion.com
SES now has 52 unique aluminum components!

GEERHEAD

by David Geer
Contact the author at geercom@alltel.net

Explorer I and Explorer Generation II

The First and Second Wireless, Robotic, Pipeline-snaking Inspection Systems!

Natural gas pipelines have traditionally been inspected by tethered, push-pull equipment with more limitations and fewer capabilities than were time or cost-effective (the technology makes for short range inspection and much excavation for the many pipeline entry points it requires).

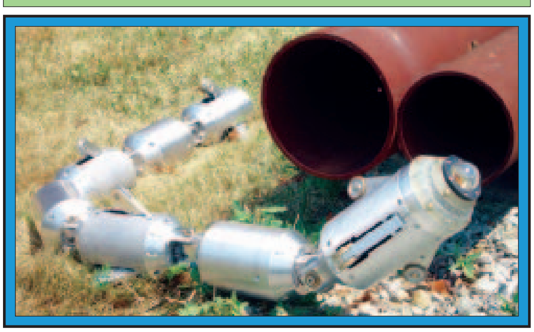
Now, the Explorer — through funding from the New York Gas Group (NYGAS), the Department of Energy (DoE), and NASA — has been created by Dr. Hagen Schempf and the Carnegie-Mellon Robotics Institute (CMU RI).

The Explorer is a "highly-articulated robot with dozens of processors and individual joints," says Dr. Schempf. "It is deployed wirelessly into a totally hazardous and inaccessible place so it better work and work well ... and it uses the latest in battery and wireless technology. In essence, it is a novel, integrated system!" he emphasizes. Each of the many architectures of the multiple modules that make up the snake robot were custom-designed by the CMU RI.

The Explorer can inspect many miles of pipeline infrastructure over the course of several hours under the wireless control of a human operator. Data captured using its front and rear cameras is communicated back to its operator wirelessly in real-time. The cameras can "image, de-warp, and mosaic" a pipeline's innards "at frame rates with a combo of edge finding [locating the pixels that belong to the edge of an image object] and Laplace [using the Laplacian operator in image processing] operations."

The Explorer is sealed and impenetrable by natural gas inside the live six- to eight-inch pipelines it traverses while investigating them for weaknesses. The robot crawls any pipe configuration including T, Y, and elbow joints. It can retract its arms to crawl on the bottom of pipes or extend them to center itself inside the six- or eight-inch pipes.

The Explorer can traverse either six- or eight-inch pipes and pipeline systems like those shown in the background here.



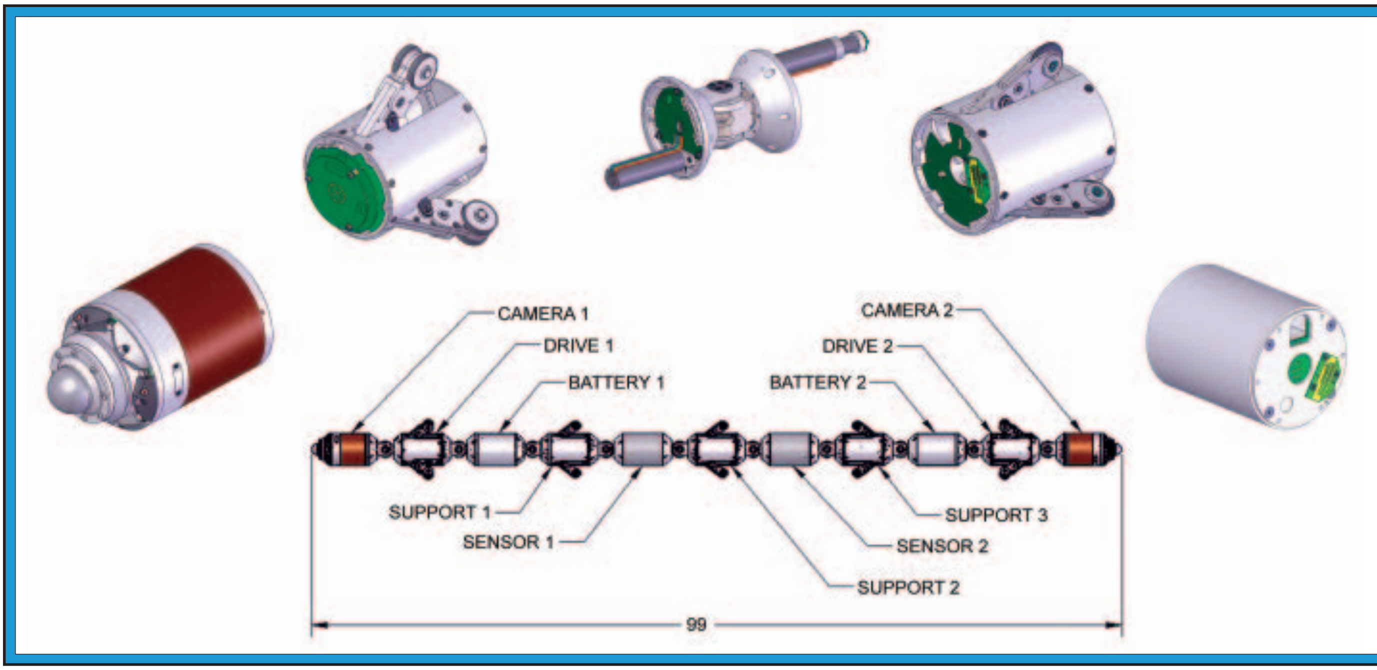
The bot (Explorer I as opposed to Explorer II, referenced elsewhere) is a snaking robot, currently made up of seven segments, with a camera eye at each end that takes and forwards images from inside the piping. Its "eyes" also help the human operator see where the robot is going. The first and seventh segments are hinged/jointed to the others with "pitch-roll" joints that aid in steering the robot via servos. The second through seventh segments connect via pitch-joints.

This configuration allows either end of the robot to lead while the attached segments follow in train-like fashion. The first and seventh segments (modules) are fitted with miniature cameras, lenses, and lighting. The modules have three drive arms that can be extended. These are powered to move the robot forward or backward. Wheels on the other module's arms are not powered, but simply act as guides.

Explorer's "Innards" Under Control; Strong Uptake

The robot is assembled with a power and communications bus that runs through its center from one end to the other. It is wirelessly controlled via a custom signal that can travel through the pipe and out to its operator. It has "actively-servo-ed" steering, pitch, and other circuitry controlled by eight-bit processors that are connected to the center control bus — a spine of sorts. The control code that runs Explorer can be altered and wirelessly downloaded to the robot for new tasks.

The Explorer is remote controlled from a console that uses scripts that control translations and joint angles.



This drawing depicts the coming, 11-segment Explorer II pipeline snaking inspection bot. Each segment is clearly marked with its primary function/cargo including cameras, drive motors and legs, battery packs, support segments with non-powered arms, and sensor segments with additional sensing on top of the two cameras. Segment enlargements at top include (left to right) the camera module, drive module, joint, support module, and battery pack module. This image is copyright Hagen Schempf.

The robot is not autonomous and must be manually "teleoperated" through its wireless connection.

The natural gas utilities are sponsoring the Explorer with a license for commercialization already in place. In fact, Explorer has done many pipeline inspections already. Few commercial robotics projects make it all the way to production, but this one looks like a clear winner.

Powered Arms and Steering Mechanism

The powered arms that move the Explorer use a single motor that drives a spur-gear attached to a centrally "distributed drive shaft, which powers a ball screw. This drives a three-bar linkage that extends and collapses the arms. A nut is attached to the ball screw and an "anti-rotation feature" is used to keep the nut from moving.

The wheels at the end of each of the six (total) drive arms are driven in synch with each other via pass-through gear trains inside the arms themselves. The gear trains then move the dual wheels at the end of each arm (12 powered wheels in all). The traction of the wheels is sufficient for the robot to climb and descend pipes vertically.

The steering system is a mesh of two different types of steering. These include the roll joints attached to each drive module which are attached to pitch joints, and actuated degree-of-freedom pitch joints that connect the other modules.

The steering mechanism uses a brushless motor-gearbox in each end module; these are commutated stepper motors, with motor-step commands used as open-loop position estimates. These are mounted off-axis and drive a bevel gear through a shaft-mounted pinion. The central shaft mounted to the bevel gear is hollow and penetrates the end bell of the module. This lets wiring pass through the entire shaft and connect to a bevel pinion gear.

The bevel pinion gear connects to a coaxial sector bevel gear with a U-jointed bearing supported shaft that the axis rotates. The joint can bend at up to a 45-degree angle.

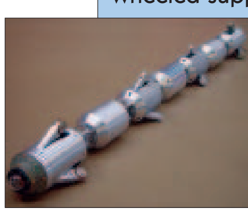
The steering mechanism also includes a switch that centers the bot during power loss. The bot's potentiometer gives it position feedback during operation, which is used to center it during power loss.

Explorer — It's Electric!

Explorer uses a "high-MIPS [million instructions per second] low-power CPU" to communicate with I^2C connected microprocessors to accomplish its control, data gathering, and I/O (input/

EXPLORER OVERVIEW

The Explorer is shown alone here with its extended support wheels (on the third and fifth segments), locomotors (extended on the first and last of its seven sections), batteries (not shown, inside the second and sixth segments), and its computer brain (in the fourth or middle segment). The wheeled support arms keep it centered in the middle of the pipe. Its seven modules, hinged one to another, aid in its crawling. With one of those bulbish, camera-equipped eyes at each end, it can see where it's going and where it's been — kind-of like having an eye in the back of its head.



The Explorer robot (the "Long-Range Untethered Real-Time Live Natural gas Main Robotic Inspection System" is its longer, more technically accurate name) is shown crawling atop the very kind of pipeline system it can crawl inside of to inspect. It may not look like much standing still, but it is capable of traveling at a rate of four inches every second for up to 10 hours (the life of the batteries) inside natural gas mains.

output) functions which traverse a custom wireless Ethernet backbone between the robot and the human-operated control device and data retrieval systems.

This CPU is a 32-bit low power model that also controls the locomotion and steering as directed by the human operator in real-time. Explorer's distributed eight-bit microprocessors communicate over the I^2C-bus. The wireless technology for communicating beyond the robot is a custom wireless LAN, which uses the pipe as a waveguide for long-range communications.

RESOURCES

Dr. Schempf's page at Carnegie-Mellon
www.rec.ri.cmu.edu/about/faculty/schempf/index.htm

Explorer II in progress
www.netl.doe.gov/technologies/oil-naturalgas/publications/td/NT42264_Phase1_Topical.PDF

Initial final report on Explorer
www.netl.doe.gov/technologies/oil-naturalgas/publications/td/41155_Final.PDF

Explorer II in Progress

Dr. Schempf and the CMU RI are working on the next generation pipeline explorer — the Explorer II. Based on Explorer I, the Explorer II stretches eight feet, encompassing its 11-module construction.

Explorer II modules include two camera fitted end modules, two drive modules (the second and 10th), two battery modules (the third and ninth), three support modules (the fourth, sixth, and eighth), and two sensor modules (the fifth and seventh).

Improvements over Explorer I include non-destructive evaluation (NDE) sensor systems that collect more data, the ability to withstand and function under pressures of up to 750 PSIG, which are the conditions inside high pressure, unpiggable steel distribution pipes.

Electronics, Software, and Hardware

The electronics have been upgraded from the Explorer I prototype. The computer brain has an updated OS and software kernel with improved performance. A locating Sodne (an electromagnetic detection sensor used by the pipeline pigging industry to detect the position of pigs in pipes) system is integrated into the bot (inside the end modules, which carry the sonde-coil and electronics) and coupled with external differential GPS to accomplish absolute positioning during pipeline inspections. The bot's power system has been upgraded to include lithium-based batteries for longer inspection mission times before recharging.

The electronics use distributed (eight bit) microprocessors (one or more per module, so 11+), which communicate to one or the other of the two 32 bit SBCs. A human operator controls every move and the data retrieved from a laptop-based remote controller.

The two sensor modules collect image data and send it to the laptop constantly and in real-time. The data is immediately sent on from there to a dedicated NDE storage computer for processing and analysis.

The computer brain of Explorer I, found in the center module, is now the computer brains (two) found in each end module inside Explorer II. These are 32 bit processors on single board computers. The end modules also contain the imaging hardware and software, as well as the off-board wireless and CAN-based on-board communication hardware and software.

CMU custom designed the architecture of every module, since this amazing robot has to work in such a unique envronment. SV

AGING NATURAL GAS PIPELINES REQUIRE FAST, CHEAP, EFFECTIVE INSPECTIONS; ENTER THE EXPLORER!

The US natural gas pipelines are getting older. These gas utilities are faced with an increasing need to inspect these pipe architectures more often. Previous inspection systems were short-range, push-pull tethered systems, requiring many excavations to insert and use them (they had ranges only up to 200 feet, making for a lot of holes in the ground when excavating over again at the end of every such distance). This increased the cost and length of inspections.

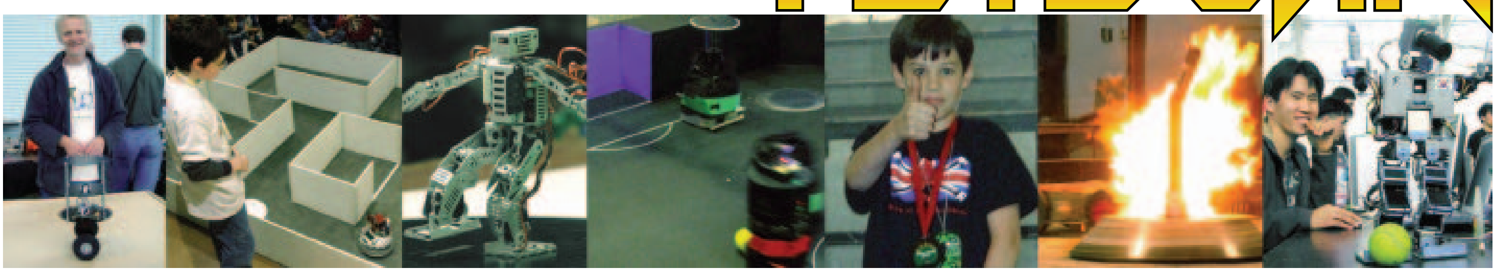
The new Explorer (Explorer II due out soon) can assess thousands of feet of pipe from a single excavation (entry) point for water in the pipes or other pipe abnormalities. This is a much cheaper method.

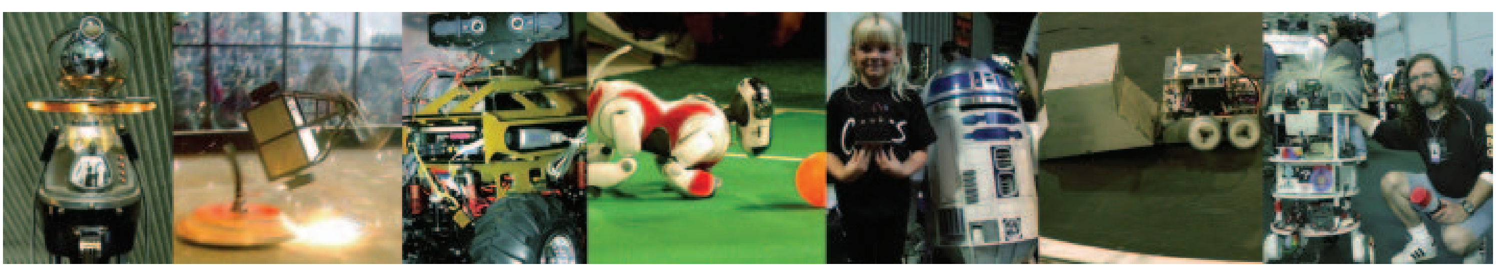
The battery-powered Explorer I completes eight-hour inspections in long-range six- to eight-inch piping and live gas mains today. The bot uses cameras to visually inspect the mains without tearing up miles of ground to get to them. (Explorer I has inspected thousands of feet of pipelines from a single entry point.)

The seven-module robot pipe-snake (soon to be 11 modules) uses wireless technology, cameras, and a train-like locomotion technique to move, capture defect images, and communicate them back to a human operator using a wireless laptop-based control system.

The pipeline Explorer — the first ever untethered, remote-controlled, live underground natural gas distribution pipe robot inspection system — was developed at Carnegie-Mellon University's Robotics Institute and continues to grow in size and capability while it is in practical use today.

Explorer I has completed many natural gas pipeline inspections and Explorer II will debut this summer, perhaps beginning its journey into a gas pipeline near you.





Our resident expert on all things robotic is merely an Email away.

roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Pete Miles

Q. I've been looking all over the Internet for a circuit that would follow someone — much like the electronic golf caddy does (the expensive ones). I can't seem to find any info on the type of circuitry and/or sensors that they use for them. I would like to make my own robo caddy to take on the golf course. Are they using radio frequency tracking? Is there something easy to make?

— Tony Cunningham
Sandpoint, ID

A. This is a first for me. I have been playing golf for about 20 years now, and I have never seen or even heard of a robotic golf caddy that follows you around on the golf course. This is a pretty interesting idea for those people that still want the exercise of walking the golf course, but don't want to carry the clubs around with them all day. The caddy that carried Rodney Dangerfield's golf bag in the 1980 movie *Caddyshack* would have loved to have one of these, or at least a remote controlled golf cart.

Remote controlled golf carts are slowly beginning to make inroads into high-tech equipment seen on today's golf courses, along with the battleship sized titanium drivers. Table 1 shows a list of several of these companies. Right now, I am not aware of any existing companies that sell autonomous robotic golf caddies. In the early 1990s, a company named GolfPro International developed a fully autonomous robotic caddy called the Intelecady which used a combination of radio beacons, GPS systems, and ultrasonic sensors to know exactly where it was on the golf course. Unfortunately, they went out of business in 2001.

The Shedda (''shadow'' in Gaelic) golf cart from Gettig Engineering and Manufacturing (**www.gettig.com/Shedda.html**) is probably the closest robotic caddy that automatically follows the golfer on the market today. I am not familiar with the exact details of how this robot knows how to follow the golfer. All I know is that a radio beacon is attached to the golfer's belt, and the robot maintains a certain distance away from the beacon. The robot will match the speed of the golfer, and when the golfer stops, the robot will stop. The radio frequency can be changed to one of seven different channels to avoid radio interference issues with other robots or other things transmitting on the same frequency.

TABLE 1. Remote Controlled Golf Caddy Companies

www.powakaddy.com
www.stewartgolfusa.com
www.electronicgolfcaddy.com
www.remotegolfcaddy.com
www.kolnex.us
www.gettig.com/Shedda.html

TABLE 2. Robotic Caddy Patents

5711388
5944132
5167389
6443543
4570732

To learn more about how these systems work, download a copy of the patents listed in Table 2. I like to use Free Patent Online (**www.freepatentsonline.com**) to get a PDF version of these patents. Though they don't provide the exact information on how to duplicate their work, it will give you enough information to understand what they did and how these systems operate so that it will guide you in your research directions.

It is my understanding that beacon following circuits are not difficult to make or purchase. A search of the Internet using key words "Robot Ultrasonic Homing Circuit," "Robot Infrared Homing Circuit," or "Robot Radio Homing Circuit" will yield all the information that you need. This is a very interesting topic, so if you are able to build one of these robots, please write an article for *SERVO* on what you did. I'm sure there will be many readers that will be excited to learn about your results since this technology can have many different applications from robots finding battery charging stations to robotic grocery carts that follow you through the store or home.

Q. That was a cool picture of a sumo robot you showed last month. I have a question for you. What is that shiny stuff on those wheels? Does it help with traction?

— Sparky Khuen

A. Sparky, I am surprised that you noticed the coating in the wheels in that photograph. The wheels are regular R/C car racing wheels (blue dot foam type). The coating is regular RTV (room temperature vulcanizing) silicon gasket seal. You can get this at any automotive parts store for a few dollars per tube. The night before a sumo contest, I apply a thin coating of the RTV to the surface of the wheels. By the next morning, they have dried enough not to be so sticky as to violate the so-called "no sticky wheels" rule that some contests use, but they are still very tacky to give excellent traction during the contest. The RTV begins to lose its tackiness after a day or two, so you need to do this a day or so before the contest. Also, the silicon does a great job at picking up every piece of lint and dirt crumb off of the sumo ring. So, you will need to clean the wheels after each match to keep them tacky throughout the entire tournament. I like to use rubbing alcohol, or — believe this or not — a baby wipe after each match to keep them clean. If you don't keep the surfaces of the wheels clean, they will eventually have less traction than the original foam. This is what gives me the competitive advantage in tournaments, and in sumo, traction is very important.

To get a fairly even coat of silicon across the wheel surface, I use one of those fake credit cards that comes in the mail all the time as a spatula to spread the silicon across the wheel's surface. Because I get these so often, I throw them out after each time I use them instead of trying to clean them. They make an ideal spatula since they are a thin, flexible piece of plastic.

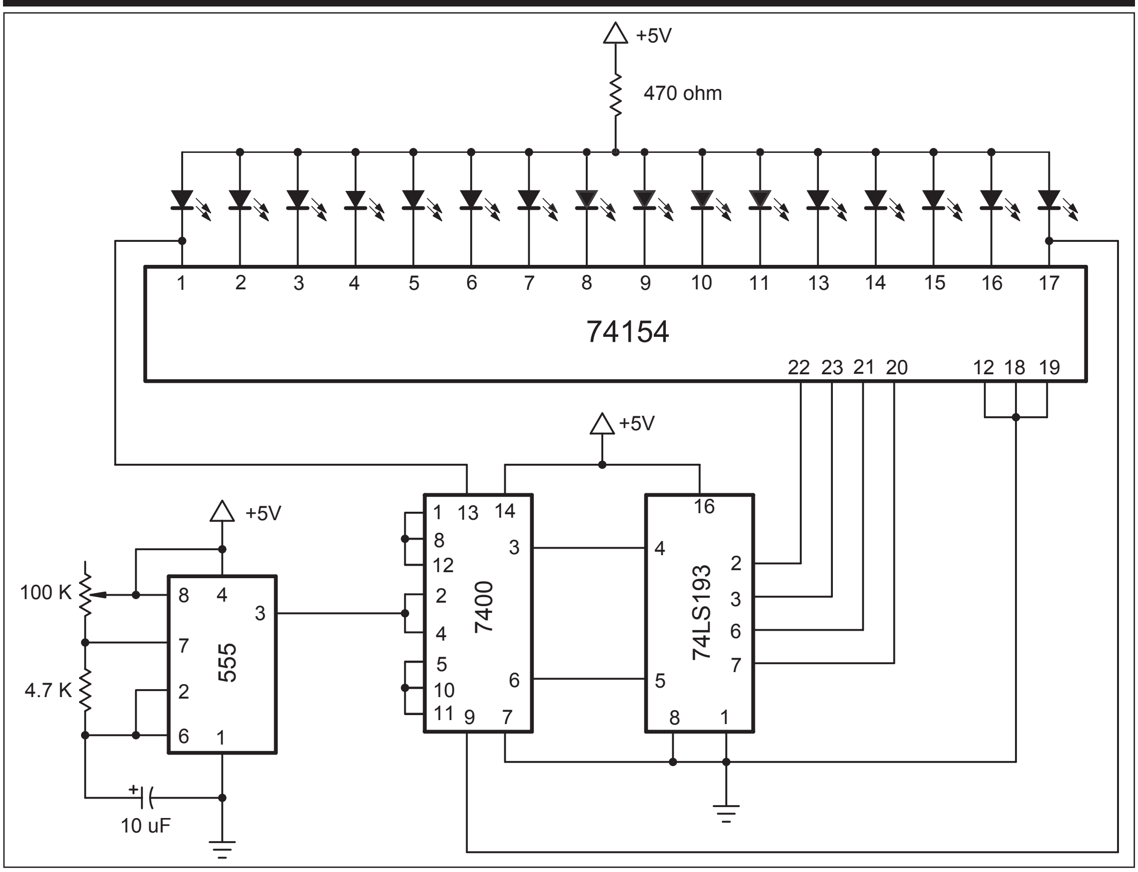
Figure 1. Applying RTV Silicon on the surface of 2.0 inch wide foam wheels.

Figure 2. Completed RTV silicon coated sumo wheels.

If you don't have any of these, a thin piece of plastic that is at least as wide as the wheel will work just fine. If you don't have any scrap pieces of plastic lying around, then a regular Popsicle stick will work. I used to use Popsicle sticks all the time until I decided to use fake credit cards. Figure 1 shows one of these fake credit cards being used to apply the silicon on the wheels, and Figure 2 shows some completed wheels. This is an inexpensive trick that greatly improves the traction on your robots. This technique can be applied to all sumo weight classes, and also combat robots.

Q. Do you know of a simple circuit that will blink a dozen or so LEDs back and forth without having to use a microcontroller to control each LED? I want to add a set of "eyes" that makes my robot look

Figure 3. Back and forth LED flasher.



like it is scanning the horizon.

— Pete Cook
San Diego, CA

A. Here is a very old circuit that does just what you asked for. When I say old, it has been at least 25 years since I first encountered it. Figure 3 shows a circuit that will flash 16 LEDs back and forth using the 74154 four-line to 16-line decoder chip to control the LED lighting sequence. The decoder chip receives a four bit binary address signal and converts that value into a single output signal on one of 16 different output lines. All of the outputs are normally high, and when one of the outputs is selected, the output goes low. By connecting an LED to each of the 16 outputs, and placing them in a linear array, the LED light can be made to move back and forth by incrementing and decrementing the four bit input address.

A microcontroller can be used to feed a sequentially changing address to the 74154, but this can also be accomplished by using the 74LS193 four-bit up/down counter. Here, the outputs of the up/down counter are fed to the inputs of the 74154 decoder. When the up or down inputs of the counter are toggled, the output address is incremented (or decremented) by one, thus causing the LED light to shift one place to the left or right. When the counter reaches 15, it will roll over to 0 when counting up, or when counting down, a 0 will roll up to a 15.

To cause the LED motion to change direction, a 7400 NAND gate is configured to switch between the up and down counting sequences of the 74913 whenever the LEDs at the ends of the 16 LED array are triggered (pins 1 or 17 on the 74154). The wring configuration of the four internal NAND gates has the convenience of only needing one clock signal to drive both the up and down counting inputs to the up/down counter.

A simple 555 timer is used to generate the clock signal to the NAND gates. Any type of a clock signal will work here, including a clock signal from a microcontroller, such as a PWM output signal. The LEDs will toggle to their next position whenever the clock's signal transitions from a high state to a low state.

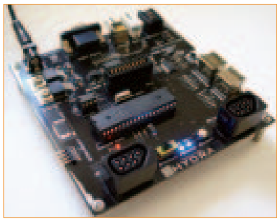
The clock's duty cycle doesn't matter with this circuit. The amount of time the LEDs are on is determined by the period of the clock's frequency (the inverse of frequency). The advantage of using a microcontroller for the clock signal is that the back and forth speed of the LEDs can be changed via software. Otherwise, the speed of the LED's motion is manually adjusted by adjusting the potentiometer to the timer.

If you want to use more powerful lights — such as a set of lamps for an outdoor display — you can substitute the LEDs with transistors so that a higher current power source can light the lamps. This circuit requires four different integrated circuits (chips) to drive the LEDs. If you are looking for a one-chip solution, then you will need to use an inexpensive microcontroller such as one of the PIC chips (**www.microchip.com**), S/X chips (**www.parallax.com**), or Atmel chips (**www.atmel.com**). But then again, there is nothing like doing things to old fashion way, using discrete components. **SV**

NEW PRODUCTS

DEVELOPMENT

The Propeller Powered HYDRA!



If you have ever thought about becoming a video game developer, now is your chance. Parallax, Inc. — a privately held company located in Rocklin, CA — has partnered with Nurve Networks LLC and Andre' LaMothe — a best-selling game development author — to release a new Propeller powered gaming product — the HYDRA Game Console. With the HYDRA, you can develop games, graphics, and media applications.

For beginner to intermediate coders, you need only basic programming experience in any Basic or C-like language. All of the hardware and software you need is included. Additionally, the HYDRA hardware is covered in detail with schematics, descriptions, dozens of games, demos, and tips allowing you to take full advantage of its resources, including its expansion port and 128K game card.

The HYDRA kit also comes with *Game Programming for the Propeller Powered HYDRA* — Andre' LaMothe's latest book. This comprehensive book covers everything you need to know about game programming for the Propeller in Spin and assembly language. All aspects of the Propeller chip are introduced, from its architecture to using the Propeller Tool for programming.

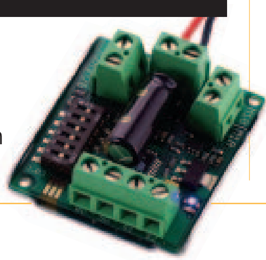
The Propeller chip was released by Parallax, Inc., in April of this year. The chip — designed at the transistor level — uses a new custom-silicon design for simultaneous multi-processing. The Propeller is a 32-bit architecture consisting of eight processors which run at 3.3V up to 80 MHz. The Propeller is programmed in both a high-level language (Spin) and low-level (assembly) language.

For further information, please contact:

Parallax, Inc.	Tel: **888 • 512 • 1024** Website: **www.parallax.com**

MOTOR CONTROLLERS

Sabertooth 2X5



Sabertooth 2X5 is Dimension Engineering's new lightweight and low-cost dual 5A motor controller. Weighing only 19g, it accepts battery voltages from 6 to 18V and will handle peak currents as high as 10A per motor.

The Sabertooth 2X5 allows you to control two motors with analog voltage, radio control, and serial modes. A built-in 5V BEC can provide power to an R/C receiver and a selectable lithium mode will allow you to safely use R/C LiPo battery packs.

As with Dimension Engineering's other motor drivers, the product's options are set with DIP switches and wiring connections are made with screw terminals, making it easy to reconfigure and move from project to project.

Sabertooth's custom designed synchronous regenerative H-bridge topology returns the motor's stored inductive energy to the battery in every switching cycle. This technique results in motors running cooler and extends battery life. It also provides more responsive control — allowing you to make instant stops and reverses.

A heat spreader comes preinstalled and the unit has electronic thermal and overcurrent protection for maximum durability.

The product retails for $59.99 and can be seen at the website listed below.

For further information, please contact:

Dimension Engineering	Website: **www.dimensionengineering.com/Sabertooth2X5.htm**

SENSORS

Flexible Stretch Sensor

The Stretch Sensor, now available from Images SI, Inc., is a unique sensor that changes resistance when stretched. The sensor has a nominal resistance of 700 ohms per linear inch. As the sensor is stretched, its resistance gradually increases. When the sensor is stretched to 200% of its original length, its resistance will approximately double to 1.4K ohms per inch.

The stretch sensor is a new way to measure stretch, displacement, and force. The sensor is a flexible, cylindrical cord .070 in diameter, with spade or ring electrical terminals at each end. Being flexible, the

sensor can measure displacement around turns and on curves.

Stretch sensors are available in the following stock lengths: 2", 4", 6", 8", 10", 12", and 14". Custom lengths may be ordered.

Flexible Stretch Sensors are sold with standard electrical terminals. Some applications for the Stretch Sensor are:

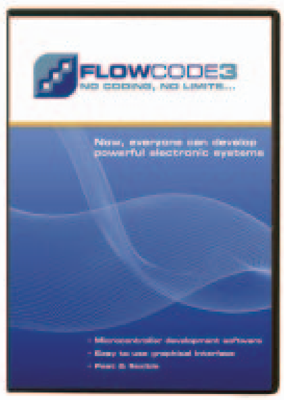
- Robotics
- Biometric displacement reading
- VR gloves and VR suits
- Physics applications and experiments
- Feedback sensor for air muscles

For further information, please contact:

Images Scientific Instruments, Inc.	109 Woods of Arden Rd. Staten Island, NY 10312 Website: **www.imagesco.com/sensors/stretch-sensor.html**

SOFTWARE

A New Flowcode



UK-based Matrix Multimedia has just announced the release of Flowcode Version 3: ultra-rapid development software for electronic systems.

For those that don't know Flowcode, it is more than just a software compiler as Flowcode's graphical user interface facilitates design, based on the popular PICmicro microcontroller, at a systems level. The design process has three stages: First, designers connect on-screen electronic building blocks to create the system. Then, designers use a flow chart to dictate the behavior of the system and simulate the results. Finally, Flowcode compiles the design into hex code for a PICmicro.

Matrix Multimedia has taken systems-level design concepts to a new level: Flowcode generates code for a large range of off-the-shelf hardware modules (including IrDA, Bluetooth, SPI, I^2C, Webserver, etc.) that match Flowcode software routines. The E-blocks hardware modules combined with Flowcode allow engineers to prototype systems with advanced functionality in a matter of a few hours.

A full evaluation of Flowcode can be downloaded from Matrix Multimedia's website.

For further information, please contact:

Matrix Multimedia	Website: **www.matrixmultimedia.com**

ROBOT KITS

Mini Solar Robot Kits Family

OWI introduces its new Mini Solar Robot and science kits. These kits are easy to assemble and demonstrate alternative energy principles. They are understandably simple, and friendly for eight-year-olds and up.

Happy Hopping Frog, Super Solar Racing Car, Frightened Grasshopper, Walking King Crab, and Attacking Inch Worm fit nicely into OWI's JR Science Series. Besides having jovial names, they have become OWI's premier entry level product.

Because of the entry level price points, retailers will appreciate their movements; both off the shelf and after assembly. If a make-it and take-it type product is what you are looking for, these hands-on products fit the bill.

Suggested selling price for OWI's new Mini Solar Robot and science kits is between $9.95-$12.95 USD.

For further information, please contact:

OWI, Inc.	17141 Kingsview Ave. Carson, CA 90746 **310 • 515 • 1900** Fax: **310 • 515 • 1606** Website: **www.owirobots.com**

COMBAT ZONE

Featured This Month

PARTICIPATION

Lessons Learned the Hard Way

● by Tim Wolter

I feel a bit sheepish admitting to some of the stupid things I have done while building and operating combat robots. I am, after all, a physician, and have on many occasions had to suture, patch, and lecture folks after they discover that not all things that seem like good ideas really are. But in my faint defense, I must say that my son and I have been around fighting robots since the early days, and lapses in judgement with low powered late 90's machines did not have the dire consequences of being casual around the new generation of ferocious juggernauts. Also, I should plead that we are entirely self-taught; I wasted my youth taking math and science classes, and had not been inside a shop since junior high. My son has a more rounded education, and actually understands the tools I wield with reckless abandon. In fact, when a mixture of concern and mild disdain is detectable on his face, I know I am pushing the old safety limits again.

- Lesson One: Most things that are fascinating to watch demand safety glasses. (Who doesn't like showers of grinder sparks?) Beware of aluminum, too. It is less dramatic, but hurts just as much.

- Lesson Two: Wear your hearing

protection. I favor the earmuff style; my son prefers the foam earplugs. Had I gotten smarter earlier, I would not have the high-pitched whining that I can hear in quiet places. This applies not just to building robots. I now wear them for arena set ups, and sometimes even at combat events.

• Lesson Three: Warning indicators should be blatant. We have tried various things to meet the requirement of a "power on" indicator. Cool neon lights that shine through the lexan sides are no good in a flood lit arena. And a teeny little buzzer that you can hear in the workshop is no good at all. Our usual expedient of positioning the speed controllers so that you can see the indicator lights through the top is borderline safe, but handy for troubleshooting purposes. Our best indicator was a car alarm siren we got for a buck at a thrift sale, but it was pretty annoying.

• Lesson Four: Replace worn saw blades, drill bits, and chop saw blades at the first hint of unreliability. You won't regret it.

• Lesson Five: When constructing an arena — even an antweight arena — the thickness of the lexan is of secondary importance. You need some sort of bumper system to keep the robots off the lexan. I have seen Super Heavyweight robots go airborne only to be stopped by cheap, replaceable rebar barriers.

• Lesson Six: Safety is relative. No amount of regulation will make thoughtless action safe; and lack of rules is not a big issue for the wary. Once, when encouraged by an event organizer to come up with something really "out there" for an exhibition machine, we created a cannon firing, remote-control Barbie Jeep. We had several layers of fail-safes built into the guns ... and used a copy of the event's safety regs to make paper cartridges to hold the "propellant!" SV

Event Equipment

● by Christopher Gilleski

The technology of running robotic combat tournaments is one of the most overlooked, yet vital components of a successful event. For those in the process of creating or revamping a tournament, there is always the temptation to go with a cheaper option. Robot combat events could not be run without the use of simple whiteboards and common programs such as Excel, and custom-designed software are all used in organizing events, some more successfully than others.

The first question presented to event organizers is whether or not to use a public registration system. The most prolific is the Builders Database located at **www.buildersdb.com** and it has hosted the registration of over 130 robot combat events. Despite this pedigree, new event organizers will often consider settling for keeping track of registration in an Excel sheet, or a simple text file. While this is a simple option, and cheaper than the Builder's Database's $2 fee per robot registered, it is likely not the best option.

The Builder's Database serves not only to handle registration, but to advertise events. Builders often wish to know what competition is going to be facing them at an event, or at the very least, want to know that the event they attend will have a sizable field for their entry to face. Registration by email or an Internet forum makes this information difficult to find, not just for competitors, but for the event staff. The database lists all relevant information about the builder and his entries making event preparation work far more streamlined than simply accepting entries yourself. Frequency conflicts can be easily identified, and fees are laid out for the competitor and event organizer.

Tournament bracket technology is another vital issue that must be dealt with when running an event. The temptation is always present to simply use pen and paper, or a white board for brackets. Written brackets are often difficult to organize when running a double elimination or round robin tournament. I have personally organized four robot combat tournaments, and used paper or white board brackets at each. The mess of brackets and white board you can see in this photograph from my first event shows what a mess doing brackets on a white board can become. These problems can still occur with the best of organizational

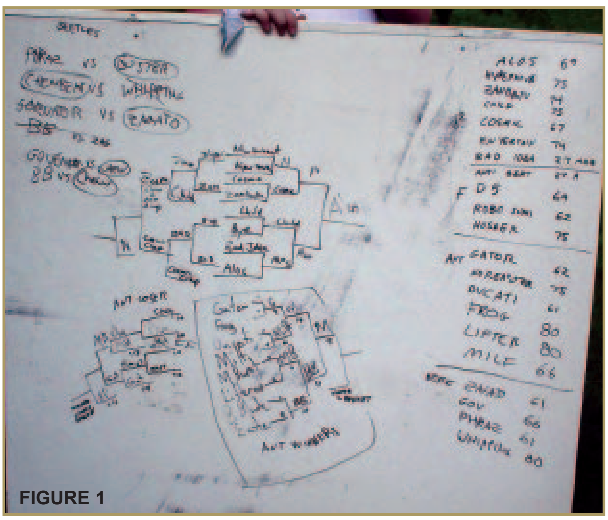


FIGURE 1

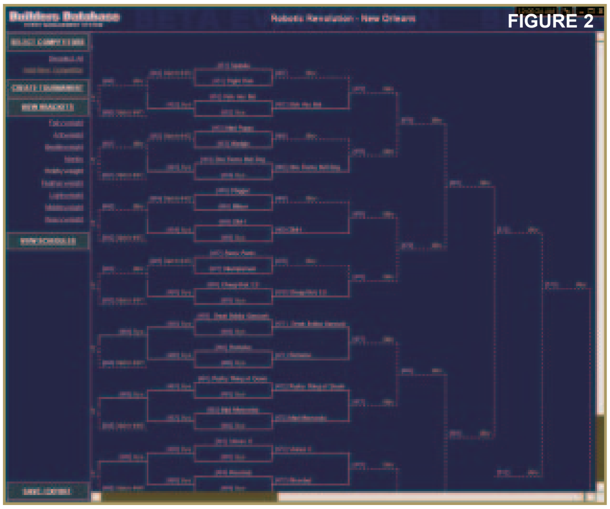


FIGURE 2

skills, which I will freely admit was sorely lacking at my events. Work is so often delegated for a moment or two at these events that it is far too easy for a robot's name to be misplaced in a bracket. Problems are not widespread, but do occur. The Second UConn Robotic Onslaught event I organized witnessed such an incident.

A robot in the loser's bracket was mistakenly taken out of the bracket before being eliminated from the event. This problem went unnoticed until much later in the event, leaving the competitor out of the event through no fault of his own. Keeping brackets neat and organized will quickly fall to the bottom of an event's list of priorities as other hassles pile up, and the brackets may end up much the same way as those shown in Figure 1.

Even with printed brackets on a page, there have been incidents where winners were misreported or placed in the wrong bracket at a handful of events such as Hobby Show Robot Conflict 2002 with the robot "Thing."

There are several alternatives available for bracket management. UI Productions — the group behind the Builders Database — has created an event management program that is pictured in Figure 2. There are also several bracket programs that can be found online for free, but are obviously not tailored to robot combat as the UI software. Microsoft Excel has also been used by several events, and has similar strengths and weaknesses as freely available bracket downloads. To staff that is experienced with running events and managing brackets, Excel or a free program may suffice. A group such as the North East Robotics Club that has people dedicated solely to running brackets can simply use Excel and other bracket software since they have run over a dozen events.

With inexperienced staff, a program such as has been made by UI Productions is an ideal solution. It allows for the simple elimination of frequency conflicts, differences in brackets between classes, and other issues that may not be as easily solved with a program not made specifically for robot combat.

This is not to say white boards and paper are completely useless. Both are excellent when used to support a computer bracket program. They present a low-cost alternative to projectors or LED displays for listing upcoming matches. Here, the pros and cons are far less pronounced, and lower cost alternatives perform nearly as well. Projectors allow for large, visible match listings if a large enough wall or screen is available. LED displays allow for easier input of matches and other information.

In most cases, events do not bring more than one display or projector, meaning all the information is usually only available around the arena itself. Another downside to scrolling LED displays is that all the information is not listed at the same time as it is on a white board or projector. This is a significant drawback as many venues have pits spread out in a large area, or even in multiple rooms. With this in mind, paper or white boards provide a cheap method of posting upcoming matches in the pit area for competitors to see without being forced to walk to the arena to see when they are needed in line.

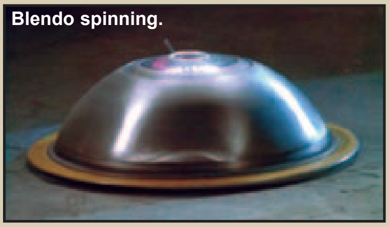
Instantaneous communication with every person at the event is vital in robotic combat. Alerts regarding frequency conflicts, safety reminders, and other pertinent information are handled far easier with the use of public address systems or megaphones. Lacking such equipment makes the tournament far more difficult to run as staff and competitors must be manually found rather than being informed by an announcement.

Without a public address system or megaphone, it is nearly impossible to inform competitors that a transmitter has been left on and is interfering with other robots. While the offending transmitter is found, a robot affected by such a conflict will run out of control in the arena, paralyzing the event until it can be stopped. Events are now making use of frequency scanners which detect such rogue transmitters, and combined with an address system, safety issues can be quickly fixed. Without such equipment, events will continue to witness the odd spectacle of people scampering through the pits yelling if a person is on their frequency.

Some of the problems presented appear insignificant, but even in the case of the smallest problems, they detract from the event and show a certain amount of disrespect to the competitors who paid to enter. Furthermore, they make the event more difficult to run, placing more burdens on the shoulders of the organizer and staff. The strangely communal nature of robotic combat events makes it possible that a competitor or fellow event organizer may be found who can loan needed equipment, making it even easier to run an event with all the needed equipment. **SV**

Combat Robotics Most Destructive Force: THE FULL BODY SPINNER

● by Robert Wilburn and Paul Reese


Blendo spinning.

Blendo stationary.

When combat robotics first took off in the mid '90s at Robot Wars in California, a bot named Blendo revolutionized the sport by taking advantage of the fact that kinetic energy can be stored in a spinning mass. Blendo's outer shell was that spinning mass. Fabricated using an inverted wok attached to a steel base ring with two nasty teeth, the shell was driven by a lawnmower engine and, once spinning, stored enough kinetic energy to easily shred its opponents. In its first competition, Blendo quickly demonstrated its superior design, spraying pieces of bot armor over the small perimeter walls toward the audience. After two fights, it was deemed too hazardous to compete by Marc Thorpe, the event organizer, and declared the de facto winner.

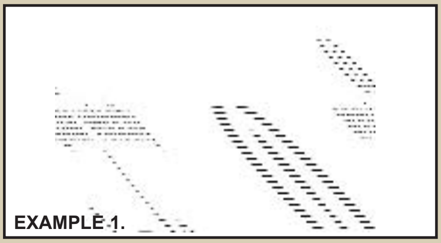
Since that day more than a decade ago, the sport has grown to accommodate the safe operation of the now named full body spinner (FBS) to the delight of viewing audiences, and with no changes in the laws of physics, others have adopted the use of stored kinetic energy in their FBS designs. Interestingly, Blendo was designed and built by Jaime Hyneman and wired by Adam Savage, both hosts of the popular TV show Mythbusters!

Full Body Spinner Theory 101 — Rotational Kinetic Energy

Rotational kinetic energy (KE) is what gives full body spinners their destructive power. The basic concept is you have a shell of a certain shape and mass that is rotated about a center axis to a certain peak angular velocity or RPM (ω). The shape of the shell and the distribution of its weight in relation to its center axis of rotation contribute to the shell's moment of inertia (I) or resistance to angular velocity changes.

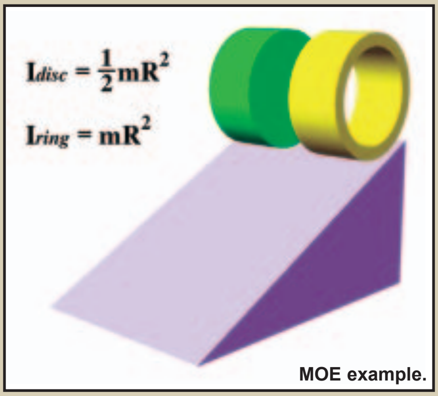
The classic downhill race between the ring and the disk shown in the MOE example below demonstrates the moment of inertia principle. If a ring and a disk of equal mass and radius are placed at the top of a hill and released at the same moment, Newton's second law for rotation tells us the disk will win the race because the moment of inertia — or resistance to changes in rotational force — is greater in a ring, so it takes longer to accelerate. In fact, we find that the disk has only half the moment of inertia of the ring.

The moment of inertia — or MOI — is important in combat robotics because most full body spinner shells are more like rings than disks. Builders try to put as much mass as possible on the perimeter of their shells which increases the MOI which, in turn, increases the kinetic energy stored and the robot's destructive power. Rotational kinetic energy (KE) can be calculated as shown in Example 1.


EXAMPLE 1.

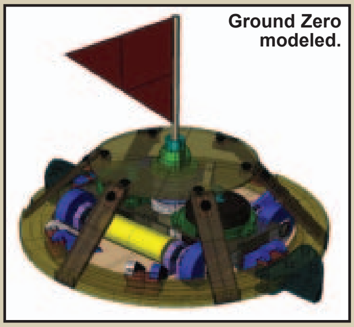
So, we can see how the moment of inertia — which is a product of the shell's mass distribution and shape — contributes to KE. But notice that angular velocity or RPM has a much greater impact because it's squared in the equation. Doubling the RPM of your shell will quadruple the total KE!

Given these two factors, builders must make decisions about how much their shell will weigh



MOE example.

Killjoy's 55 lb shell has a high MOI.



Ground Zero modeled.

Ground Zero tooth completed with S7 steel cutters.

versus how fast it will spin. For example, a Middleweight bot with a 50 lb. shell spinning at 1,450 RPM might store equal KE as a Middleweight with a 21 lb. shell spinning at 2,230 RPM. A savings of 29 lbs. with the same KE! The heavier shell, while stronger, will have a slower spin-up time and will be more challenging to incorporate into a design due to its greater percentage of the bot's total weight. The lighter shell will spin-up faster and be easier to incorporate into a design due to its lighter weight, but will be inherently weaker and more prone to deformation or failure. All FBS builders must strike a balance between a shell's mass, strength, MOI, RPM, spin-up time, gearing, torque, and horsepower applied.

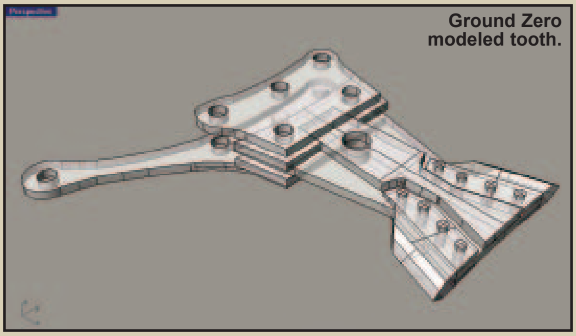
Full Body Spinner Design and Build Tips

For me, half of the fun of combat robotics is the design and build phase. Here are some tips to make your spinner's design and build phase flow smoothly.

• *Do your research*. Like any R/C-based hobby, combat robotics costs money. Select a weight class you can afford to compete in. Sure, bigger bots are cool, but they are also much more expensive to build and maintain, not to mention logistically difficult to transport. The price of a single weapon motor for a 340 lb. super heavyweight might be enough to cover the entire cost of a competitive 12 lb. hobbyweight class bot and a box full of spare parts, and trust me, you're going to need spare parts.

Visit the sports official sanctioning body, The Robot Fighting League (RFL) **http://botleague.net/** and its combat robotics forum **http://forums.delphiforums.com/THERFL** to get started. Here you will find a wealth of information including rules and regulations and a friendly, knowledgeable community willing to answer your questions and provide positive feedback. Weight class choices start at 5 oz. and go up to 340 lbs.

• *Consider using a 3D modeling program to design your robot*. Make your hobby an educational experience as well as a fun one. Test your design ideas and strategies in cyber space before you start building. Modeling is fun and might prove useful for other projects outside of robotics. Among the combat robot community, SolidWorks and Rhino3D are popular choices. I recommend Rhino3D because it's very easy to learn and is available for download in a fully functional trial version that will allow 25 saves. With proper save management, this should be enough to complete your design. After 25 saves, the program remains fully functional, only it will no longer save. If 3D modeling is not something you feel you can handle, then break out the graph paper and cardboard mock-ups to get the job done! Many great bots were hand-built without the use of a computer.

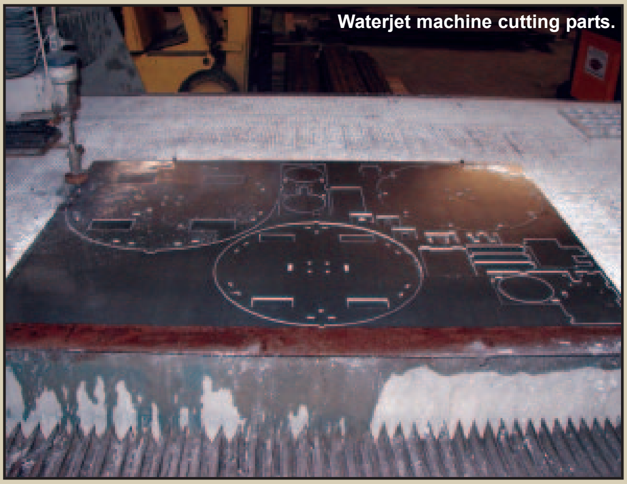


Ground Zero modeled tooth.

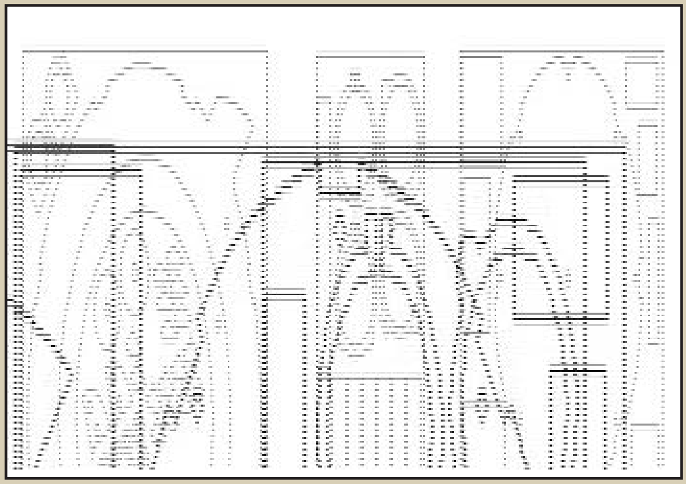
If you do decide to try 3D modeling, you can save some time by importing completed 3D models. Most of the common off-the-shelf components such as motors, wheels, speed controllers, batteries, and receivers have already been modeled and are available for download on builder sites such as **happy robots.com**. Also, most drive components can be downloaded from the Stock Drive Products website **sdp-si.com**.

Most 3D modeling programs can perform useful calculations such as total weight and moment of inertia. Knowing the exact weight of your modeled design as you progress, combined with the published weights of the components you plan to use, will allow you to modify the design as required along the way to ensure your completed bot is underweight. Finishing your bot to discover you're 10% overweight with no easy way to make up the difference is something you want to avoid.

The MOI of your completed shell design can also be calculated by modeling programs. Plug this into the KE rotational formula discussed above along with angular velocity to get your design's rotational kinetic energy. This, of course, is not a requirement when designing a spinner, but it is fun to



Waterjet machine cutting parts.



Exported 2D layout ready for waterjetting.

know where you stand.

Perhaps the greatest advantage to using 3D modeling is the ability to export your modeled parts into a 2D file format and have them cut out by a waterjet machine. A waterjet machine uses a computer controlled, high pressure stream of water induced with a fine abrasive grit to blast a very clean line through almost any material up to five inches thick. It can cut your parts out in minutes. The amount of time and material saved and accuracy achieved by a waterjet versus the moderate cost of the service makes it worth considering. Otherwise, those elaborate chassis components you designed must be cut out by hand.

If you decide the waterjet service is not in the cards, you can still take advantage of the modeling program's ability to print a 2D image of the part in actual size. The printout can then be glued to your material where it will provide you with the exact lines to cut along, maintaining fairly decent dimensional accuracy. The more dimensionally accurate your parts are, the easier the assembly will be. The sport's choice for waterjetting service is **METFAB.biz**. They provide excellent service and offer online quotes.

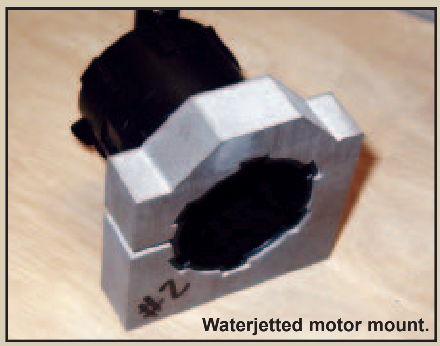
• *Incorporate a directional indicator and, if possible, a self-righting device*. Spinners require some sort of visual directional indicator, otherwise the driver has no sense of front and rear because the only thing he sees is the blur of the outer shell. Knowing your heading at all times is crucial to effective driving. You don't want to accidentally drive into the perimeter steel I-beam because you lost your sense of direction. Many builders use lights or LEDs that are visible through carefully located holes in the shell. Others use shafts or appendages that emerge from the top center of the shell and are bent to indicate the front or rear or have a flag attached.

Spinners are inherently hard to flip over due to the gyroscopic forces in play when they're at full RPM but trust me, you will eventually be that proverbial upside-down turtle. A self-righting device aims to keep you from being flipped over. Most self-righting devices made for shell spinners are sturdy rods or hollow shafts that double as the directional indicator. Not all competitive spinners have self-righting devices, but at one time or another, they have all wished they had.

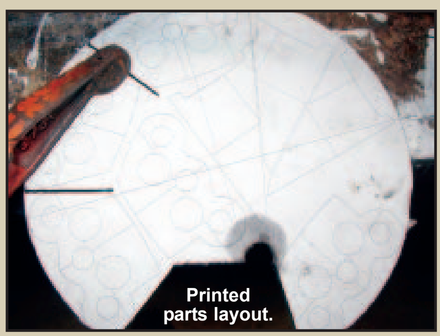
• *Common chassis and shell materials*. The primary materials used in today's combat robots are aluminum and titanium because of their high strength and light weight. Titanium has the highest strength-to-weight ratio and is weldable, but it can be

Ground Zero with self-righting directional flag.

Waterjetted parts.



Waterjetted motor mount.



Printed parts layout.

somewhat pricey. Among aluminum alloys, 6061 and 7075 are the preferred choices. If welding is planned, the 6061 alloy must be used. If welding in not planned, use the 7075 alloy with fasteners because its 84% stronger than 6061.

Due to its density, the use of steel is normally avoided except for on striking surfaces such as teeth where heat treated S7 tool steel is the proven choice. The majority of FBS shells are made with titanium these days. However, with careful weight budgeting, steel alloys such as Chromoly 4130 and 4340 have been used successfully and proven very robust. **Titaniumjoe.com** is the sports' preferred source for titanium. Titanium welding and fabricating services are available from **Teamwhyachi.com**, as well as any other fabrication service you can dream up.

• *Ordering parts and components*. Try to invest in proven components. The sport has been around long enough to provide solid evidence of which brands and models can withstand the brutal shock and G-loads encountered. This ties back into the research part of the design phase. There are many sources for combat robot parts. Check out the **Robotmarket place.com**, **Teamwhyachi.com**, and **Teamdelta.com** for starters.

• *Popular Full Body Spinners*. Since the inception of combat robotics, dozens upon dozens of full body spinners have competed. Here are a few websites of some of the sports' best: **Teamlogicom.com**, **Roboticdeathcompany.com**, **Teamwhyachi.com**, and **Teamotown.com**.

Final Thoughts

Building a full body spinner can be a serious challenge, so research and planning are key to success. Carefully consider your abilities and budget before selecting a weight class. Keep in mind that, the lighter the weight class, the easier it will be to complete your spinner. Smaller generally means: easier to fabricate, less dependent on outside services, requires less material, and most importantly, requires less time and money. I hope you gained an understanding of the basic concepts behind combat robotics most destructive force — the Full Body Spinner. Good luck! SV

TECHNICAL KNOWLEDGE

Inrunner Brushless Motors — Getting the Heat Out

● by Russ Barrow

Recently, many small combat robots and electric airplanes have moved to lower RPM, higher torque outrunner motors. Having personally used many types of these motors, they can provide unique mounting and simplified drive capabilities. However, inrunner motors continue to offer higher power and smaller packaging per weight. To understand the advantages of the inrunner brushless motors, we need to understand what factors favor the design.

A motor's worst enemy is heat. When a brushless motor is powered from a battery or voltage source, the voltage will energize the coils/windings, creating a magnetic field that will spin the motor shaft assembly (rotor) to a specific RPM, as defined by the size of the wire (gauge) and the number of wire rotations (turns) around an iron-based core (stator). The coils (typically in sets of three for either a delta or wye configuration) will have a defined resistance. The current the motor pulls is a direct result of the load applied, which can be represented as power output (power in – motor losses = power output).

FIGURE 1

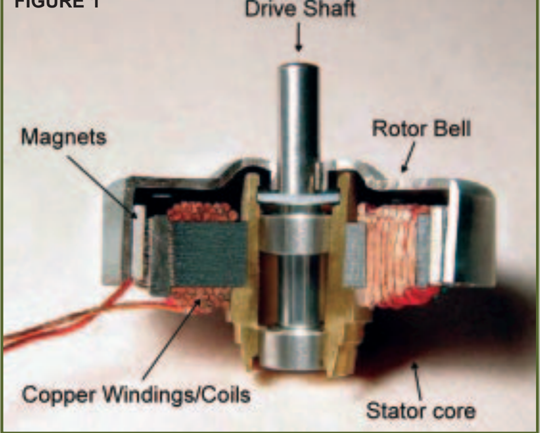
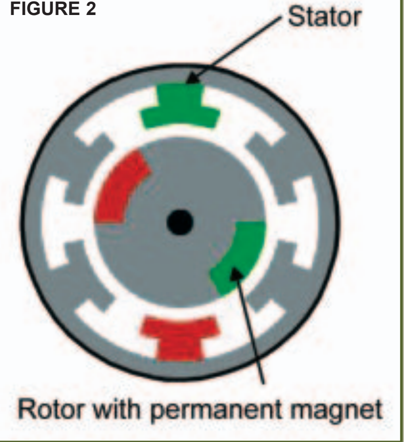


FIGURE 2



A larger load will require more power into the motor to turn the shaft, and therefore, require more current through the coils (power = voltage x current). The resistance of the coils, along with other motor dynamics (often referred to as motor efficiency), results in heat, and this heat will increase with load/current. How well a motor can dissipate heat will define the maximum power output (in Watts) that a motor can reliably deliver.

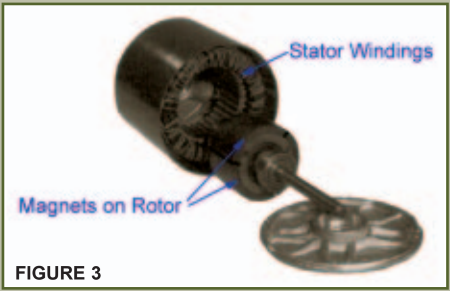


FIGURE 3

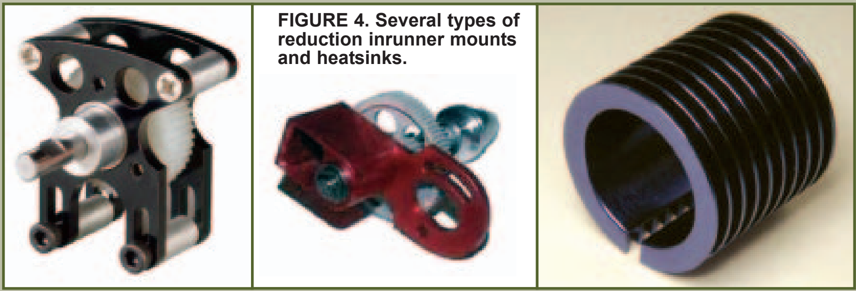


FIGURE 4. Several types of reduction inrunner mounts and heatsinks.

The stator coils consist of insulated copper wires wound one on top of the other around an iron stack. With enough heat, the plastic shield coating (electrical insulation) the coil wires will start to burn off. Once the wires are un-insulated, current will find the path of least resistance, and move from wire to wire (short) versus traveling through the coil. As this happens, the resistance of the coils decreases, causing even more current to pass through the core. As a result, the coils transfer less magnetic field to the spinning shaft assembly, reducing the motor efficiency. This chain reaction of events will happen in a matter of seconds, and can be identified by smoke coming from the motor, reduced RPM of the motor, and/or charring (black coloring) of the coils.

Most outrunner brushless motors do not dissipate heat as efficiently as an inrunner, and this is due to how they are designed. The outrunner brushless motor has a stationary coil center core — or stator — that generates a magnetic field. A cylindrical bell with magnets placed along the inside perimeter of the bell, is placed over the stationary core. The bell spins around the core. The air gap between the stator and the magnets of the bell is minimized for optimal magnetic efficiency, but at the same time impeding good airflow through the motor.

In addition, since the outside bell of the motor is spinning, only the circular mounting base can be used to attach the motor to a heatsink or model. Some manufacturers have provided angled port holes in the bell in an attempt to force air across the stator when the bell is spinning. Unfortunately, with smaller motors found in electric airplanes and small combat robots, these ports produce more undirected air turbulence than airflow through the motor. Figure 1 shows a cut-away of a common outrunner brushless motor.

Inrunner brushless motors have the coils (or stator) placed around the perimeter of the motor can (outside cylinder). The motor drive shaft is connected to a set of magnets or an induction rotor that spins inside the stator. Figure 2 illustrates the internal structure of an inrunner brushless motor.

Since the stator is attached to the motor can, the coil windings will transfer a considerable amount of heat to the cylindrical motor walls. Therefore, the entire surface of the motor can be used to dissipate heat. When an inrunner brushless motor is used with a heatsink or additional airflow is created around the motor, considerable power output is possible. Figure 3 shows a disassembled inrunner brushless motor.

The only drawback to the inrunner brushless design is the higher RPM and lower torque per revolution the motor produces as compared to the outrunner. This is due to the larger diameter of the stator and rotor on the outrunner motor. In general, inrunner motors will require 2X the reduction as an outrunner to match the torque at the driven shaft. But with many combat robot designs, an increase in weapon RPM reduction is possible with nothing more than a smaller motor pulley or gear. In fact, many manufacturers sell mounts, drive shafts, and gears based on electric airplane propeller requirements (see Figure 4).

In addition, most inrunner motors have several different stator winding options that affect the torque and RPM of the motor, allowing the user to match the torque and RPM to the application. For example, the GWS GWBLM005 brushless inrunner series of motors have several different windings that provide the following RPMs per volt (KV) with no load applied (free spinning motor): 4,600 (blue), 3,900 (green), 3,000 (black), and 2,300 (yellow).

I have recently begun using the GWBLM005 series of GWS inrunner brushless motors that not only validate the inrunner advantages, but also offer a price that is in the sub $25 category (see Figure 5). I am using the GWBLM005A (green) motor on my antweight combat robots Dark Pounder and Dark Siren with great results, and I have not found a better power-to-weight, low-cost motor (see Figure 6). SV

FIGURE 5. The GWS GWBLM005 inrunner brushless motor series.

FIGURE 6. GWBLM005A mounted in an aluminum channel to dissipate heat in Dark Siren. The motor generates an impressive 54,000 rpm when powered from a 14.8 volt, four-cell Lithium Polymer battery pack.

PRODUCT REVIEW — Mega Motor USA

● by Chad New

In today's age, brushless motor technology is constantly evolving. New building techniques are being developed and different materials are being put through their trials. When it comes to choosing a brushless motor, your options are extremely numerous, but as a combat builder, you want the best quality of motor that will give you the maximum amount of durability and power for a fair price. So basically, you want the best!

In 2000, Mega Motor USA was established and began to sell their motors to enthusiasts all over the country. Today, they provide high quality motors at very affordable prices with great customer service. Their motors are hand wound, which gives them a substantially stronger magnetic field than machine wound motors. They also use high quality Neodymium magnets which are stronger than cobalt or ferrite magnets.

Mega Motor currently sponsors my combat team, Team Wazio, so we use their motors on two robots. One is an antweight, "Get-R-Done," and the other a hobby weight, "Apogee." Get-R-Done uses an RC-400/7/12 which gives a no load RPM of 1,115 RPM per volt. With a weight of only 39 grams, it leaves plenty of weight for an effective weapon, as well as armor. Get-R-Done uses this motor on a direct driven press fitted drum where the motor is actually put inside the weapon and spun 1:1.


Apogee


Get-R-Done

As a member of Team Wazio, I'm happy to report that the motors have proven — through combat testing — to be much more durable than motors of comparable price and size. I also use Mega Motor's largest outrunner — the RC-41/30/12 — on Apogee. This motor delivers 510 RPM per volt with no load, weighs only 14 oz, and is able to handle a maximum of 40 amps! Apogee spins a three pound horizontal blade at 3,000 RPM, which is able to inflict major damage on anything the blade comes in contact with. Team Wazio is going to expand its use of Mega Motors in the coming 2007 season and will keep *SERVO* posted on their performance.

In my opinion, Mega Motor has motors that will fit almost every application from brushed drive to brushless weapon. With their high quality service and products, **www.MegaMotorUSA.com** is definitely worth a look. **SV**

EVENTS

UPCOMING — March and April

Central Illinois Bot Brawl 2007 — Takes place on 3/10/2007 in Peoria, IL. Visit **http://circ.mtco.com** combat and non-combat event. RC combat antweights; Auto Sumo 3kg, 500g, LEGO; Line Following; Line Maze. $7 per bot pre-registration, $10 day of event. Spectators free.

Seattle Bot Battle 5 — Takes place on 4/10/2007 in Seattle, WA. Presented By Western Allied Robotics at the Seattle Center, Center House. Event Time: 12pm-5:00pm, Safety Inspection: 10:30am-11:30am. If a lot of robots register, event may start safety and fights earlier. Three and 12 lb classes, Double Elimination or Round Robin (RFL Rules). NO ICE or open flames allowed. Entry Fee: $40 for first 12 lb robot, $25 for first 3 lb robot. Additional robots are half price. Entry fee discount for helping with arena setup and take-down. Special entry fee considerations for builders who are under 18. Arena is 12'x12', no hazards, one pushout likely. Pushout will have at least a 3/8" lip around it to make accidental driving into it difficult. Go to **www.westernalliedrobotics.com/** for more details.

RattleBots Invitational — Takes place on 4/14/2007 in Dorchester, WI. Presented By WHRE. Trophies for all classes; cash prizes for all classes with three or more bots. Go to **www.rattlebots.com/** for more details. **SV**

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

The first event on the calendar this month is RoboWars. Despite the name, this is not a contest for radio controlled vehicle destruction. These folks have been holding autonomous robot Sumo events since 1991. In 2001, they added a BEAM Solaroller event. If you're not familiar with Solarollers, they're small autonomous robots powered by photovoltaic cells. This got me wondering about other solar-powered robot contests. If anyone is planning a larger scale contest along these lines, I'd love to hear about it.

The last major solar robot event I remember hearing about was the Trans-Tasman Solar Challenge back in 1996. The robots in that contest were autonomous boats and the course ranged from Porirua Harbor near Wellington, New Zealand to Townsville Beach in North Queensland, Australia. That's a distance of about 2,000 nautical miles! The robots — which were up to four meters in length — were allowed to use GPS for navigation and were required to radio their location back to the judges at least once per day. Maybe it's time to try another solar-powered robot contest on that scale. Let me know what you think!

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

March

3 **RoboWars**
Montreal, Canada
Sumo and BEAM Solaroller events.
www.robowars.ca

9-10 **AMD Jerry Sanders Creative Design Contest**

Continued on page 67

Seeing With OpenCV

Follow That Face!

by Robin Hewitt

PART 3

Last month's article in this series explained how to implement and configure face detection. This month, I'll show you how to use OpenCV to track a face once you've detected it.

Face Tracking in OpenCV

Tracking a face is more difficult than tracking a strongly-colored object. Skin reflects the ambient light in subtle, changing ways as a person's head turns or tilts.

In principle, you could track a face by locating it over and over in every frame, using the Haar detector described in last month's article. To do that, however, you'd need to decide if the face you detected in each frame is the same face. If the detector finds more than one face in a frame, you'd need to decide which detection is the one you're tracking. Finally, if a person's head tilts towards one shoulder, or turns towards profile view, the frontal face detector will no longer detect it, so you'd need to handle that situation, as well.

Fortunately, OpenCV includes specialized code for tracking a face efficiently, using continuity between frames to help find the best match for the face it's following.

The algorithm that OpenCV uses for face tracking is called Camshift. Camshift uses color information, but rather than relying on a single color, it tracks a combination of colors. Since it tracks by color, it can follow a face through orientation changes that the Haar detector can't handle. The sidebar, "How OpenCV's Face Tracker Works," explains this algorithm in more detail.

Camshift was originally developed for hands-free gaming. It's designed to be very fast and "lightweight" so the computer can do other tasks while tracking. Since it was developed as a gaming interface, Camshift also has an (limited) ability to detect changes in head position, such as tilting the head to one side. Could you use that ability to communicate with your robot? Maybe two fast head tilts mean "Come here, robot!"

Figure 1 shows OpenCV's face tracker in action — following a face as it tilts to one side and during a turn to profile.

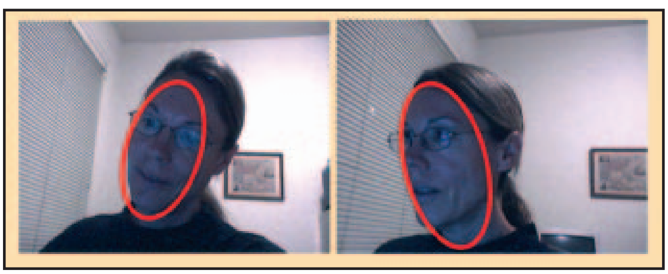


FIGURE 1. OpenCV's face tracker in action. It's able to follow a face as it tilts to one side and during a turn to profile.

The Camshift Demo

The OpenCV samples directory contains a program called camshiftdemo. You can get some good hands-on experience and an intuitive feel for the Camshift algorithm with this demo program. Here are the steps for doing that:

1) Plug in a webcam.

2) Launch the program called camshiftdemo in the samples directory.

3) Use your mouse to select a rectangle centered tightly on your face.

4) Click in the video-display window and type the letter *b*. (The display should change to look something like the view in Figure 2.)

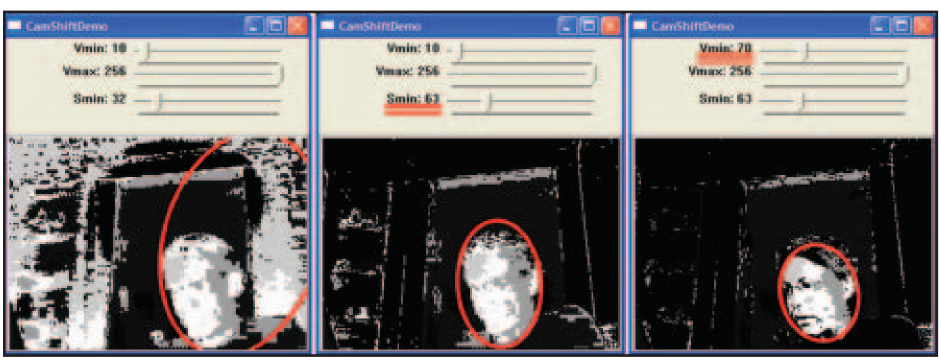


FIGURE 2. To tune the Camshift parameters `smin` and `vmin`, run the camshiftdemo program in the samples directory. These parameters are easier to set if you toggle to the backprojection view by clicking in the view window, then typing *b*.

How OpenCV's Face Tracker Works

OpenCV's face tracker uses an algorithm called Camshift. Camshift consists of four steps:

1) Create a color histogram to represent the face.

2) Calculate a "face probability" for each pixel in the incoming video frames.

3) Shift the location of the face rectangle in each video frame.

4) Calculate the size and angle.

Here's how each step works:

1) *Create a histogram.* Camshift represents the face it's tracking as a histogram (also called a barchart) of color values. Figure A shows two example histograms produced by the Camshift demo program that ships with OpenCV. The height of each colored bar indicates how many pixels in an image region have that "hue." Hue is one of three values describing a pixel's color in the HSV (Hue, Saturation, Value) color model. (For more on color and color models, see "The World of Color," *SERVO Magazine*, November '05.)

In the image region represented by the top histogram, a bluish hue is most common, and a slightly more lavender hue is the next most common. The bottom histogram shows a region in which the most common hue is the rightmost bin. This hue is almost, but not quite, red.

2) *Calculate face probability — simpler than it sounds!* The histogram is created only once, at the start of tracking. Afterwards, it's used to assign a "face-probability" value to each image pixel in the video frames that follow.

"Face probability" sounds terribly complicated and heavily mathematical, but it's neither! Here's how it works. Figure B shows the bars from a histogram stacked one atop the other. After stacking them, it's clear that the rightmost bar accounts for about 45% of the pixels in the region. That means the probability that a pixel selected randomly from this region would fall into the rightmost bin is 45%. That's the "face probability" for a pixel with this hue. The same reasoning indicates that the face probability for the next histogram bin to the right is about 20%, since it accounts for about 20% of the stack's total height. That's all there is to it.

As new video frames arrive, the hue value for each pixel is determined. From that, the face histogram is used to assign a face probability to the pixel. This process is called "histogram backprojection" in OpenCV. There's a built-in method that implements it, called `cvCalcBackProject()`.

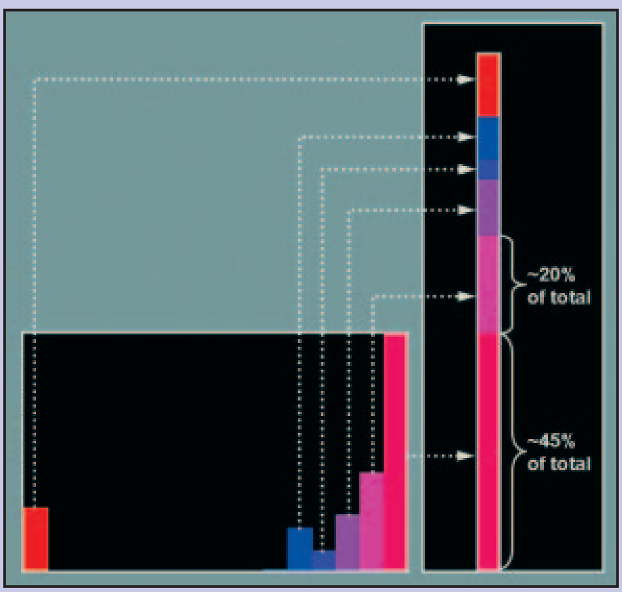


FIGURE B. To see what "face probability" means, imagine stacking the bars in a histogram one atop the other. The probability associated with each color is the percent that color bar contributes to the total height of this stack.

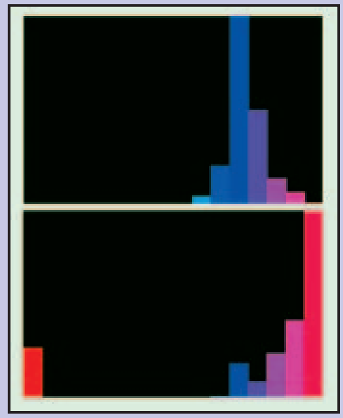


FIGURE A. Two examples of the color histogram that Camshift uses to represent a face.

Figure C shows the face-probability image in one video frame as Camshift tracks my face. Black pixels have the lowest probability value, and white, the highest. Gray pixels lie somewhere in the middle.

3) *Shift to a new location.* With each new video frame, Camshift "shifts" its estimate of the face location, keeping it centered over the area with the highest concentration of bright pixels in the face-probability image. It finds this new location by starting at the previous location and computing the center of gravity of the face-probability values within a rectangle. It then shifts the rectangle so it's right over the center of gravity. It does this a few times to center the rectangle well. The OpenCV function `cvCamShift()` implements the steps for shifting to the new location.

This process of shifting the rectangle to correspond with the center of gravity is based on an algorithm called "Mean Shift," by Dorin Comaniciu. In fact, Camshift stands for "Continuously Adaptive Mean Shift."

4) *Calculate size and angle.* The OpenCV method is called "Continuously Adaptive" and not just "Mean Shift" because it also adjusts the size and angle of the face rectangle each time it shifts it. It does this by selecting the scale and orientation that are the best fit to the face-probability pixels inside the new rectangle location.

FIGURE C. The normal and face-probability views as Camshift tracks my face. In the face-probability view, black pixels have the lowest value, and white, the highest. Gray pixels lie somewhere in the middle.

```
main()

//// Constants
const char * DISPLAY_WINDOW = "DisplayWindow";
#define OPENCV_ROOT  "C:/Program Files/OpenCV/1.0"

//// Global variables
IplImage  * pVideoFrameCopy = 0;

void main( int argc, char** argv )
{
  CvRect * pFaceRect = 0;
  if( !initAll() ) exitProgram(-1);

  // Capture and display video frames until a face
  // is detected
  while( 1 )
  {
     // Look for a face in the next video frame
     captureVideoFrame();
     pFaceRect = detectFace(pVideoFrameCopy);

     // Show the display image
     cvShowImage( DISPLAY_WINDOW, pVideoFrameCopy );
     if( (char)27==cvWaitKey(1) ) exitProgram(0);

     // exit loop when a face is detected
     if(pFaceRect) break;
  }

  // initialize tracking
  startTracking(pVideoFrameCopy, pFaceRect);

  // Track the detected face using CamShift
  while( 1 )
  {
     CvBox2D faceBox;

     // get the next video frame
     captureVideoFrame();

     // track the face in the new video frame
     faceBox = track(pVideoFrameCopy);

     // outline face ellipse
     cvEllipseBox(pVideoFrameCopy, faceBox,
         CV_RGB(255,0,0), 3, CV_AA, 0 );
     cvShowImage( DISPLAY_WINDOW, pVideoFrameCopy );
     if( (char)27==cvWaitKey(1) ) break;
  }

  exitProgram(0);
}
```

FIGURE 3. The main program listing for detecting a face in a live video stream, then tracking it using the Camshift wrapper API.

5) Adjust the sliders for `smin` and `vmin` until the ellipse is well positioned and the background is mostly black.

6) Repeat Step 4 to toggle back to normal view, then use Camshift to track your face.

Tuning Camshift

As mentioned above, Camshift uses a combination of colors to track faces. In the representation that Camshift uses, color is undefined for pixels that have a neutral shade (white, gray, or black). Color can be computed for pixels that are *almost* neutral, but their color values are unstable, and these pixels contribute noise that interferes with tracking.

Camshift uses two parameters — `smin` and `vmin` — to screen out this noise. These parameters define thresholds for ignoring pixels that are too close to neutral. `vmin` sets the threshold for "almost black," and `smin` for "almost gray." These two threshold levels will need to be adjusted for your setup to get good results with Camshift.

Camshift also uses a third parameter called `vmax`, to set a threshold for pixels that are too bright. But `smin` has the side effect of also eliminating pixels that are close to white, so you shouldn't need to tweak `vmax` to get good results.

The easiest way to select good values for your setup is with camshiftdemo. As suggested in the preceding section, it's easier to set these if you toggle the viewing mode by clicking the view window and typing *b*. (This alternative view is the called the "face-probability," or "backprojection" view. It's explained in the sidebar.)

Figure 2 shows the effect of adjusting `smin` and `vmin`. Initially, in the first frame, these were at their default values. At these levels, Camshift displayed a very large ellipse that included not only my face, but half the room as well! The reason for the oversized face detection is clearly visible in the face-probability view. Background pixels with a nearly neutral shade contributed too much noise when `vmin` and `smin` were at their default values.

The middle and right views in Figure 2 show the effect of increasing first `smin`, then `vmin`. In the right-hand view, noisy pixels have been largely eliminated, but the face region still produces a strong signal. Tracking is now quite good, and the ellipse is well positioned.

The Simple Camshift Wrapper

OpenCV includes source code for camshiftdemo, but it's not easy to adapt, since it combines user-input handlers and view toggling with the steps for face tracking.

If you're programming in C++, rather than in C, you could use the CvCamShiftTracker class, defined in cvaux.hpp. Again, however, this class is fairly complex, with many interfaces, and is only available to C++ programmers.

To make the Camshift tracker more accessible, I've written a wrapper for it in C with four main interfaces:

1) `createTracker()` pre-allocates internal data structures.

2) `releaseTracker()` releases these resources.

3) `startTracking()` initiates tracking from an image plus a rectangular region.

4) `track()` tracks the object in this region from frame to frame using Camshift.

There are two additional interfaces for setting the parameters vmin and smin:

1) `setVmin()`
2) `setSmin()`

The Camshift wrapper is online at **www.cognotics.com/opencv/downloads/camshift_wrapper/index.html**.

FIGURE 4. The helper functions `initAll()` and `exitProgram()` handle program initialization and cleanup.

initAll()

```
int initAll()
{
  if( !initCapture() ) return 0;
  if( !initFaceDet(OPENCV_ROOT
     "/data/haarcascades/haarcascade_frontalface_default.xml"))
     return 0;

  // Startup message tells user how to begin and how to exit
  printf( "\n******************************************\n"
          "To exit, click inside the video display,\n"
          "then press the ESC key\n\n"
          "Press <ENTER> to begin"
          "\n******************************************\n" );
  fgetc(stdin);

  // Create the display window
  cvNamedWindow( DISPLAY_WINDOW, 1 );

  // Initialize tracker
  captureVideoFrame();
  if( !createTracker(pVideoFrameCopy) ) return 0;

  // Set Camshift parameters
  setVmin(60);
  setSmin(50);

  return 1;
}
```

exitProgram()

```
void exitProgram(int code)
{
  // Release resources allocated in this file
  cvDestroyWindow( DISPLAY_WINDOW );
  cvReleaseImage( &pVideoFrameCopy );

  // Release resources allocated in other project files
  closeCapture();
  closeFaceDet();
  releaseTracker();

  exit(code);
}
```

Combining Face Detection and Tracking

In camshiftdemo, you needed to manually initialize tracking with the mouse. For a robotics application, it would be much nicer to initialize tracking automatically, using a face detection that the Haar detector returned. (See last month's article for details on implementing face detection.)

This section shows how to do that using the Camshift wrapper described above. The program described here detects a face in a live video stream, then tracks it with Camshift. The source for code for the complete program, called "Track Faces," is also available online at **www.cognotics.com/opencv/downloads/camshift_wrapper/index.html**.

The Main Program

Figure 3 shows the main program listing for detecting a face in a live video stream, then tracking it using the Camshift wrapper API. (This portion is in TrackFaces.c in the download.) There are three main program segments:

1) Detect a face.
2) Start the tracker.
3) Track the face.

1) *Detect a face*. Lines 15-27 implement a loop to examine video frames until a face is detected. The call to `captureVideoFrame()` invokes a helper method to bring in the next video frame and create a copy of it. (Recall from Part 1 of this series that it's never safe to modify the original video image!) The working copy is stored as `pVideoFrameCopy`, declared at line 6.

2) *Start the tracker*. When a face is detected, the code exits this loop (line 26) and starts the tracker (line 30), passing it the face rectangle from the Haar detector.

3) *Track the face*. Lines 33-48 contain the face-tracking loop. Each call to the wrapper's `track()` method (line 41) invokes Camshift to find the face location in the current video frame. The Camshift result is returned as an OpenCV datatype called `CvBox2D`. This datatype represents a rectangle with a rotation angle. The call to `cvEllipseBox()` at lines 44-45 draws the ellipse defined by this box.

Helper Functions

In addition to the main program,

captureVideoFrame()

```
 1 void captureVideoFrame()
 2 {
 3   // Capture the next frame
 4   IplImage  * pVideoFrame = nextVideoFrame();
 5   if( !pVideoFrame ) exitProgram(-1);
 6
 7   // Copy it to the display image, inverting it if needed
 8   if( !pVideoFrameCopy )
 9     pVideoFrameCopy = cvCreateImage(cvGetSize(pVideoFrame),8,3);
10   cvCopy( pVideoFrame, pVideoFrameCopy, 0 );
11   if( 0==pVideoFrameCopy->origin ) cvFlip(pVideoFrameCopy,0,0);
12 }
```

FIGURE 5. The helper function `captureVideoFrame()`. At line 11, the call to `cvFlip()` flips the image upside down if the origin field is 0.

References and Resources

- *OpenCV on Sourceforge*
  http://sourceforge.net/projects/opencvlibrary
- *Official OpenCV usergroup*
  http://tech.groups.yahoo.com/group/OpenCV
- G.R. Bradski, “Computer video face tracking for use in a perceptual user interface,” *Intel Technology Journal*, Q2 1998.
- D. Comaniciu and P. Meer, “Robust Analysis of Feature Spaces: Color Image Segmentation,” *CVPR*, 1997.
- *The Simple Camshift Wrapper*
  www.cognotics.com/opencv/downloads/camshift_wrapper/index.html
- Source code in this article can be downloaded from:
  www.cognotics.com/opencv/servo

detectFace()

```
 1 CvRect * detectFace(IplImage * pImg)
 2 {
 3   CvRect* r = 0;
 4
 5   // detect faces in image
 6   int minFaceSize = pImg->width / 5;
 7   pFaceRectSeq = cvHaarDetectObjects
 8       (pImg, pCascade, pStorage,
 9       1.1,
10       6,
11       CV_HAAR_DO_CANNY_PRUNING,
12       cvSize(minFaceSize, minFaceSize));
13
14   // if one or more faces are detected, return the first one
15   if( pFaceRectSeq && pFaceRectSeq->total )
16       r = (CvRect*)cvGetSeqElem(pFaceRectSeq, 0);
17
18   return r;
19 }
```

FIGURE 6. The `detectFace()` function. The min_neighbors parameter is set to 6 to reduce the chance of a false detection.

TrackFaces.c also contains helper functions for initialization and cleanup — `initAll()` and `exitProgram()`. These are shown in Figure 4.

At line 21 in `initAll()`, the call to the Camshift wrapper's `createTracker()` function pre-allocates the wrapper's internal data structures. It's not necessary to pre-allocate the tracking data, but doing so speeds the transition from face detection to tracking. The next two statements (lines 24-25) set the parameters `smin` and `vmin`. The best values to use for these depends on your setup, so it's a good idea to select them ahead of time using the camshiftdemo program, as described above.

Figure 5 shows the listing for `captureVideoFrame()`. At line 11, a call to `cvFlip()` flips the image upside down if the `origin` field is 0. The reason for doing this is that some webcam drivers — especially on Windows — deliver image pixels starting at the bottom, rather than at the top, of the image. The `origin` field indicates which row order the IplImage uses. Some OpenCV functions will only work correctly when these images are inverted.

Finally, Figure 6 contains the `detectFace()` function. Although this code should be familiar from last month's article, one point worth noting is that the `min_neighbors` parameter should be set high enough that false face detections are unlikely. (Otherwise, your robot might start tracking the refrigerator magnets!) At line 10, I've set it to 6, which is more restrictive than the default value of 3.

Coming Up

So far, the faces we've been finding and following have been anonymous. The robot can tell there's a face present, and can follow it, but has no way of knowing whose face it is. The process of linking faces to names is called face recognition. OpenCV contains a complete implementation of a face-recognition method called eigenface.

The remaining two articles in this series will explain how to use OpenCV's eigenface implementation for face recognition. In the first of these, I'll explain how the algorithm works and give you code to create a database of people your robot “knows.” The article following that takes you through the steps for recognition from live video, and gives you tips to help you get the most out of eigenface.

Be seeing you! **SV**

SMART SERVO

Understanding the AX-12

by Michael Simpson

While researching my next RobotFest project, I came across the AX-12. It takes something special to get me fired up, but this little gem does the trick. There are two ways to get your hands on an AX-12. You can purchase them individually or as part of the Bioloid kit. Both can be purchased from CrustCrawler at **www.crustcrawler.com**.

When purchased separately, the AX-12 Servo Kit comes with two frame sets and a good deal of hardware, as well as a nice 8" cable, as shown in Figure 2.

Let's take a closer look at the AX-12 and how to use it.

FIGURE 1

AX-12 Features

The feedback provided by the AX-12 enables you to constantly monitor the device. You can look at its current position, as well as the load being placed on the device. You can monitor current voltage and temperature, as well. Another cool feature is that you can turn off the motor on the AX-12 and monitor the current position of the device. This makes it possible to pose more complex robots which will aid in their programming. I will use this feature in future articles when I build more complex robots.

Before moving on, I need to mention that the AX-12 is not a toy and I do not recommend its use in robots where small children are likely to come in contact with the arms and legs of your bot. The reason is that the AX-12 is extremely powerful and with a holding torque of 16 Kg, they can pinch or even break small fingers.

One of the more important features is the communication speed. I have dealt with servo controllers in the past that used 9600 bps or even 115200 bps. The problem is that when you start issuing many commands to many servos, you start to get delays. This also takes processing time away from your microcontroller. At 1,000,000 bps, you can send several commands to many servos and get near instant results.

AX-12 Physical Characteristics

The AX-12 is roughly the size of a standard servo without the mounting tabs as shown in Figure 1. In lieu of the typical servo mounting tabs, there are 20 small mounting tabs. These tabs are designed so that small #2M nuts can be inserted into each one. This allows you to attach the AX-12 using #2M machine screws.

The AX-12 also has a special bearing located opposite the main drive.

AX-12 FEATURES:

1,000,000 bps communications Speed.

Full feedback on Position, Speed, Load, Voltage, and Temperature.

Can be set to full rotation mode (gear motor mode).

Full 300 degree movement in 1024 increments.

Full control over speed in 1024 increments.

Full control over max torque in 1024 increments.

Built-in LED that can be used as a status indicator.

Automatic shutdown based on voltage, load, or temperature.

Single cable network connections.

You can control hundreds of AX-12 actuators with only two data ports.

Synchronized servo movements.

Servo movement range can be set by user.

FIGURE 2

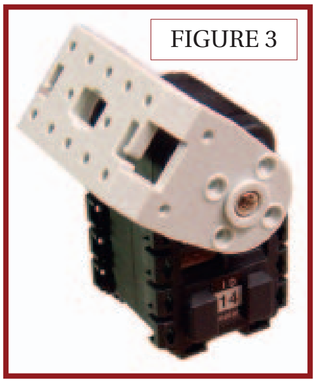


FIGURE 3

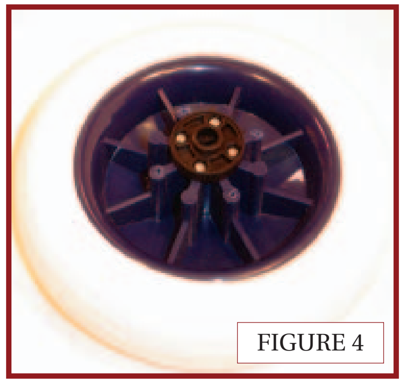


FIGURE 4

This allows you to attach various frame sets like the one shown in Figure 3. This makes assembling robot joints very easy.

Mounted on the drive shaft is a small hub. This hub has four mounting holes with #2M nuts installed. This makes it easy to attach frame sets or other hardware to the hub. You can even attach wheels to the hub. In Figure 4, you can see how I attached the hub to a large 6" foam wheel. Why would you attach a wheel to a servo? The AX-12 is not a servo, but a very powerful and efficient gear motor with a tiny microcontroller installed. This microcontroller lets you configure the AX-12 like a servo or like a gear motor in a fully rotational mode.

I found the easiest way to mount the AX-12 was to take the included OF-12S frame set and enlarge a few of the holes with a 3/32" drill bit as shown in Figure 5. This will allow you to easily self-tap the piece using standard #6 machine screws.

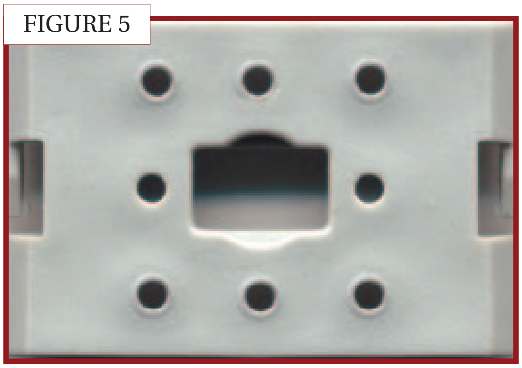


FIGURE 5

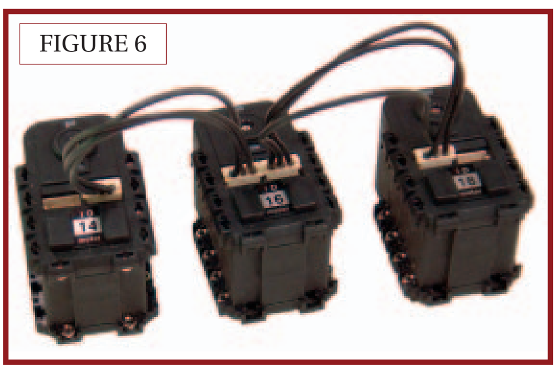


FIGURE 6

AX-12 Electrical Characteristics

The AX-12 requires a 7V-10V power source to operate. At 7V, the speed is .269 seconds for a 60 degree rotation. At 10V, this improves to .195 seconds. You have full control of the speed and it has a maximum operating current of 900 mA.

Unlike standard servos, all AX-12s are connected on a three-wire bus. This allows you to daisy-chain the cables as shown in Figure 6.

A small three-wire cable is used to connect the servos to the controller. The pin spacing on the connector is .1" so you can use a standard servo header or make your own as shown in Figure 7. You will only need the header on the cable that is connected to the microcontroller.

AX-12 Communications Protocol

The AX-12 protocol is quite simple. In order to communicate, you need to be able to communicate at 1,000,000 bps and since the protocol is half duplex, you need to be able to tie your TX and RX leads together and have the ability to take your TX off-line, as needed. The speed alone will eliminate most microcontrollers. With the PC software that is included with the Bioloid kits, you can change the baud rate but I don't recommend it. The slower the communications speed, the more unresponsive the AX-12 becomes.

Let's look closer at the actual protocol. Don't worry if you don't understand all the bits and bytes as I have provided both Dios and Zeus libraries that take care of all the lower-level communications (see Table 1).

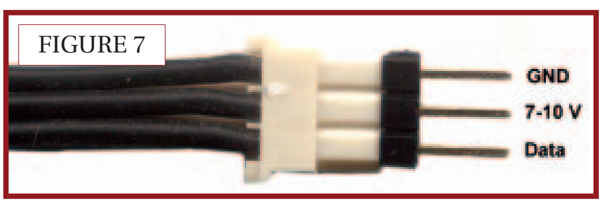


FIGURE 7

Start

The first two bytes are always 255. They indicate the start of an incoming packet.

ID

Only 0-254 may be used. A value of 254 indicates a broadcast to all devices on the network.

Length

If a single instruction is used with no data, then the length value will be 2. If one data parameter is used, then the length value will be 3.

Instruction

Valid instructions are as follows:

1	Ping
2	Read Data
3	Write Data
4	Reg Write
5	Action
6	Reset
7	Sync Write

The Reg Write instruction is used in conjunction with the Action command to help synchronize certain commands. You send a Reg Write command to all the AX-12s, then issue an Action instruction. This tells the AX-12 to issue the stored command.

Data1-N

This is the optional data parameter(s). Some instructions like Reset and Ping do not have data parameters. Others have one or more data parameters.

Checksum

All bytes except for the start bytes are added together, then the bits of that sum are inverted to generate the checksum.

Most of the instructions you will send to the AX-12 involve reading or writing one of the registers. For instance, to set the Goal Position register, you would write to register 30. All the registers, as well as packet examples, are listed in the AX-12 manual and can be downloaded from the CrustCrawler website at **www.crustcrawler.com/products/bioloid/docs/AX-12.pdf**.

When a command packet is sent to the AX-12, it will respond with a packet of its own. This status packet is much simpler and will contain the data you may have requested, as well as an error code (see Table 2).

Start

Same as with the command packet: two bytes of 255 which indicate the start of the incoming packet.

ID

Will contain the ID of the device indicated by the packet just received.

Length

If a single instruction is used with no data, then the length value will be 2.

Error

A single byte indicating the status of the last command packet. Each bit has a particular meaning (see Table 3).

Byte	Name	Description
1	Start	Always 255
2	Start	Always 255
3	ID	The ID of the AX-12 you wish to talk to
4	Length	The number of parameters +2
5	Instruction	The action for the AX-12 to perform
6	Data 1-N	Optional data in addition to instruction
7	CheckSum	= Sum of the ID + Length + Instruction + Parm1 + ParmN

TABLE 1

Byte	Name	Description
1	Start	Always 255
2	Start	Always 255
3	ID	The ID of the AX-12 you wish to talk to
4	Length	The number of parameters +2
5	Error	Reports the error status of the last command
6	Data 1-N	Optional data in addition to instruction
7	CheckSum	= Invert(Sum of the ID + Length + Instruction + Parm1 +... ParmN)

TABLE 2

Data1-N

This is the optional data parameter(s). It will be sent if you have requested data from the AX-12.

Checksum

Also same as the command packet.

In summary, you send a command packet and get a status packet back. If you requested data, it will be contained in the status packet as data parameters. At 1,000,000 bps, this all takes place very quickly.

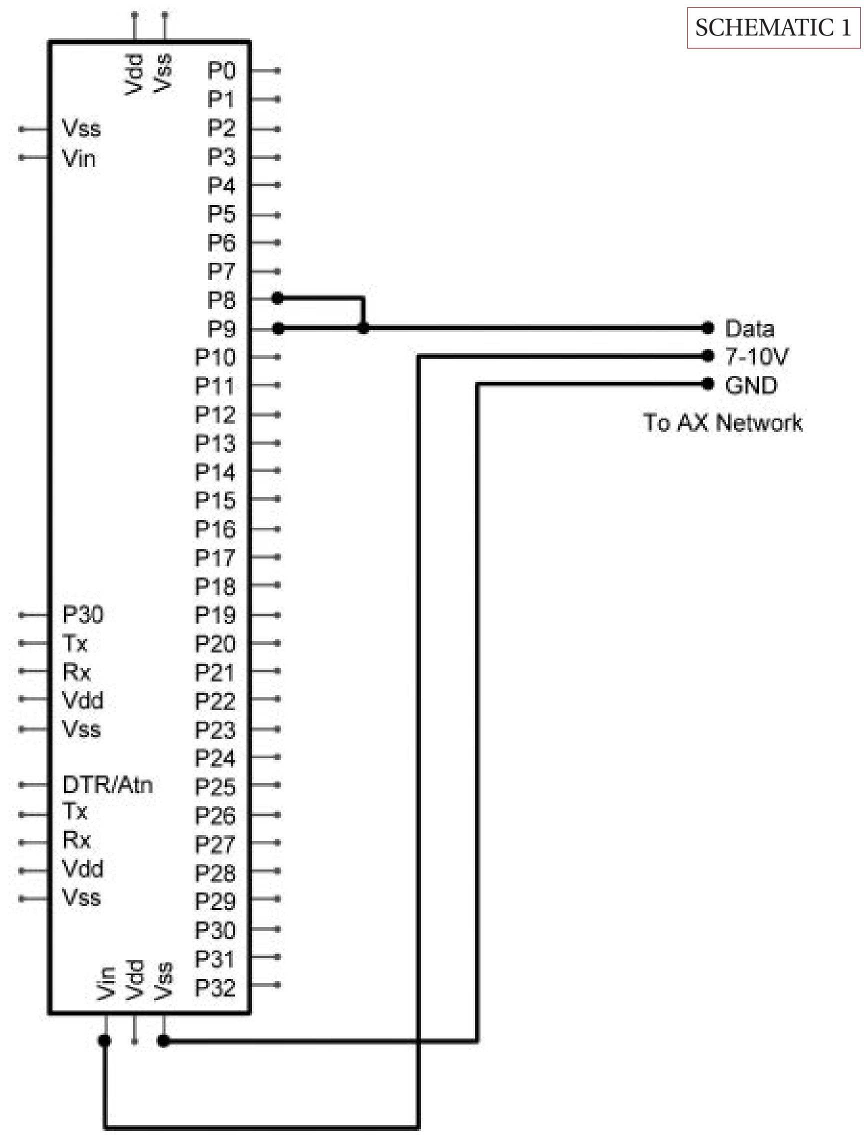
Microcontroller Interface

Another strength of the AX-12 is that you don't need a servo controller. It is a simple task to connect a microcontroller to the AX-12. However, a few things need to be kept in mind when selecting a microcontroller to use. First, to take advantage of the high speed interface of the AX-12, you need a microcontroller that can communicate at 1,000,000 bps. You also need the ability to place the transmit lead in a high impedance state

Bit	Name	Description
0	Input Voltage Error	If set to 1, indicates the operating voltage is out of range. The high and low range is set by registers 12 and 13.
1	Angle Limit Error	If set to 1, indicates the goal position sent is out of range. The limit range is set by registers 6-9.
2	Temperature Error	If set to 1, the internal temperature of the AX-12 has exceeded the limit set in register 11.
3	Range Error	Set to 1, if the parameter data is out of range for the specific instruction in the last packet.
4	Checksum Error	Set to 1, if the checksum received in the last packet is incorrect.
5	Overload Error	Set to 1, if the load on the servo has exceeded that which is set by registers 14 and 15.
6	Instruction Error	Set to 1, if an invalid instruction has been received or an Action command has been received without sending a Reg Write Command.
7	0	Not used.

TABLE 3

SCHEMATIC 1



when receiving data from the AX-12. The DiosPro microcontroller is a perfect match. With just a few lines of code, you can communicate with the AX-12.

Schematic 1 is a simplified schematic of the DiosPro Workboard. It shows how simple it is to connect an AX-12 to a DiosPro chip or board. You simply need to connect ports 8 and 9 of the DiosPro together. These are then connected to the data leads on the AX-12.

Important: The Dios Workboard can handle a wide range of input voltages, but if you are using it to power an AX-12 device, you should use a power source in the range of 8-12V.

Each AX-12 has a built-in LED. Program 1 shows how easy it is to blink that LED. You simply need to change the device constant to reflect the actual device ID of the AX-12 you are communicating with.

I have written a complete library that is now included with the free Dios compiler.

PROGRAM 1

```
DiosPro
func main()

    const device 18
    AXinit()

Loop:
   AXwritebyte(device,25,1)
   pause 200
   AXwritebyte(device,25,0)
   pause 200
   goto Loop

endfunc

include \lib\AX.lib
```

AX-12 Bot Example

Many of you who have seen my robotic articles in the past know that I like to use a simple IR interface to test my robot creations. The commands for doing this are built into the DiosPro, so a single call will allow you to use a universal remote to issue commands to even the most complex bot.

As I mentioned earlier, the AX-12 can be operated in full rotational mode. This means they can be used as very powerful gear motors with full speed control. Unlike a modified servo, the AX-12 gives you a full range of control over the speed.

To place an AX-12 in full rotational mode, simply issue the following commands:

```
AXwriteword(AXnum,
   AXCCW_Angle_Limit,0)
AXwriteword(AXnum,
   AXCW_Angle_Limit,0)
```

Now, to place the motor in motion, issue the command:

```
AXwriteword(lAXnum,
   AXMoving_Speed,speed)
```

Speed can be any value of 0 to 1023. If you set bit 10, the motor will rotate in the opposite direction. A simple way to do this is by adding 1024 to the speed.

Construction

I used a Dios Workboard Deluxe to make experimenting easier. It was a simple matter to wire the AX-12 network cable and Vishay IR to the breadboard shown in Figure 9.

The base for the AXbot is made from a piece of 1/8" x 5" x 7" wood or Plexiglas. (You can also use compressed PVC.) I shaped the front of my AXBot as shown in Figure 8. Feel free to make as many modifications as you like.

The four outside standoffs are

marked by placing the workboard on top of the base and marking the four holes. Drill 1/8" holes into these four spots.

Mark the holes for the caster or roller ball for the front support, again as shown in Figure 9. This support should be as far forward as you can place it.

Use the 12S mount that comes with the AX-12 to mark four holes so that you can mount each AX-12 in the position shown. Placement is not critical but the wheels should form a triangle with the front caster or roller ball. The AX-12 should be roughly located as shown in Figures 9 and 10.

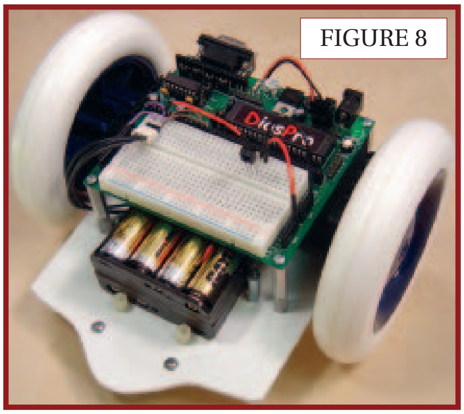


FIGURE 8

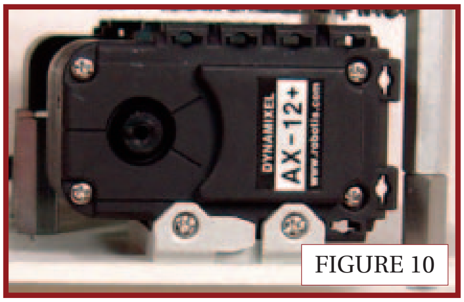


FIGURE 10

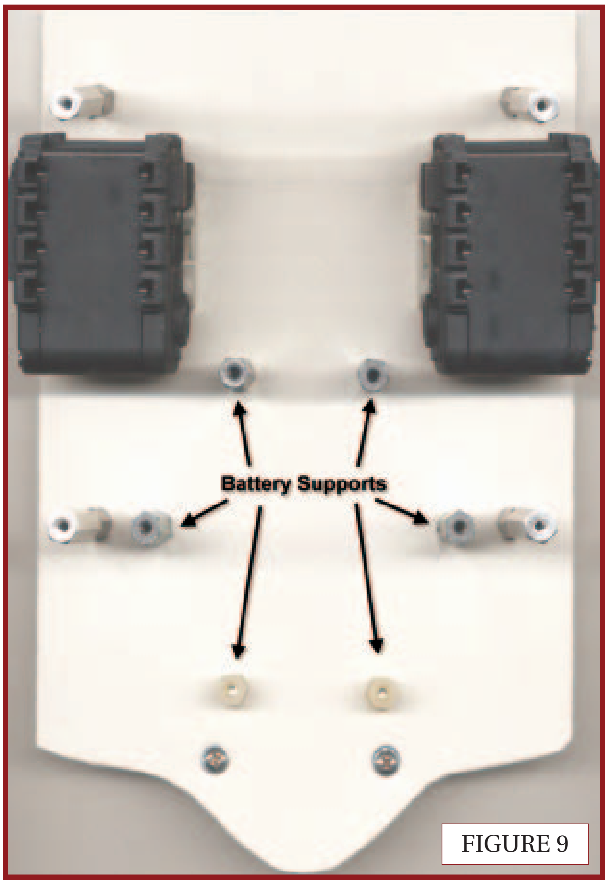


FIGURE 9

Once the holes are marked, you will need to drill 1/16" holes so that you can mount the 12S. Once attached to the base, you can mount the AX-12 to the 12S mount.

You will need to make a 9.6V battery pack for your AXbot. I used two four-cell AA battery holders connected in series as shown in Figure 11. The battery holders come with foam tape on the back. I removed this and used black tape to hold the two pieces together. If you already have a pack, it may just be a simple matter of creating an adapter for the connector. You may also use a 7.2V pack, but the bot won't be as fast and it won't have as much run time.

You need to add some standoffs to hold the battery pack in place. To mark the location of these standoffs, just place your pack on the base. Make sure it is placed a bit forward so that the weight of the packs rests near the front of the wheel/roller triangle. This will keep your bot from flipping when moving forward. When in place, mark the locations for the battery supports as shown in Figure 9. I used 3/4" standoffs for the rear and sides and 1/4" for the front. This will allow you to slide the pack in the front of the AXbot.

The Dios Workboard Deluxe has a header for supplying power. This is convenient if you don't want to use the built-in coax connector. By connecting your battery to the COAX and AC2 as shown in Figure 12, you connect to the bridge rectifier and switch. This allows you to use any polarity on the connection and the ability to switch the power on and off via the switch.

Important: Make sure the connection is as described. If you connect your battery to the wrong pins, you will damage the DiosPro chip. Depending on the connector that you use, you may have to move one of the wires on the connector.

To connect the AX-12 network to the DiosPro, you need to use a three-pin header as mentioned earlier. Plug the


FIGURE 11

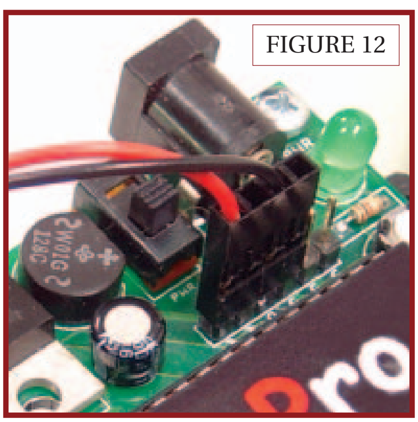


FIGURE 12

FIGURE 13

PARTS LIST

AVAILABLE FROM CRUSTCRAWLER (www.crustcrawler.com)
- Two, AX-12 Smart Servos

AVAILABLE FROM KRONOS ROBOTICS (www.kronosrobotics.com)
- Two, four-cell, AA battery holders #16323
- DiosPro Chip #16148 (also available from CrustCrawler)
- Dios Workboard Deluxe #16452 (also available from CrustCrawler)
- Vishay IR module #16226
- 5/8" Roller ball #16311

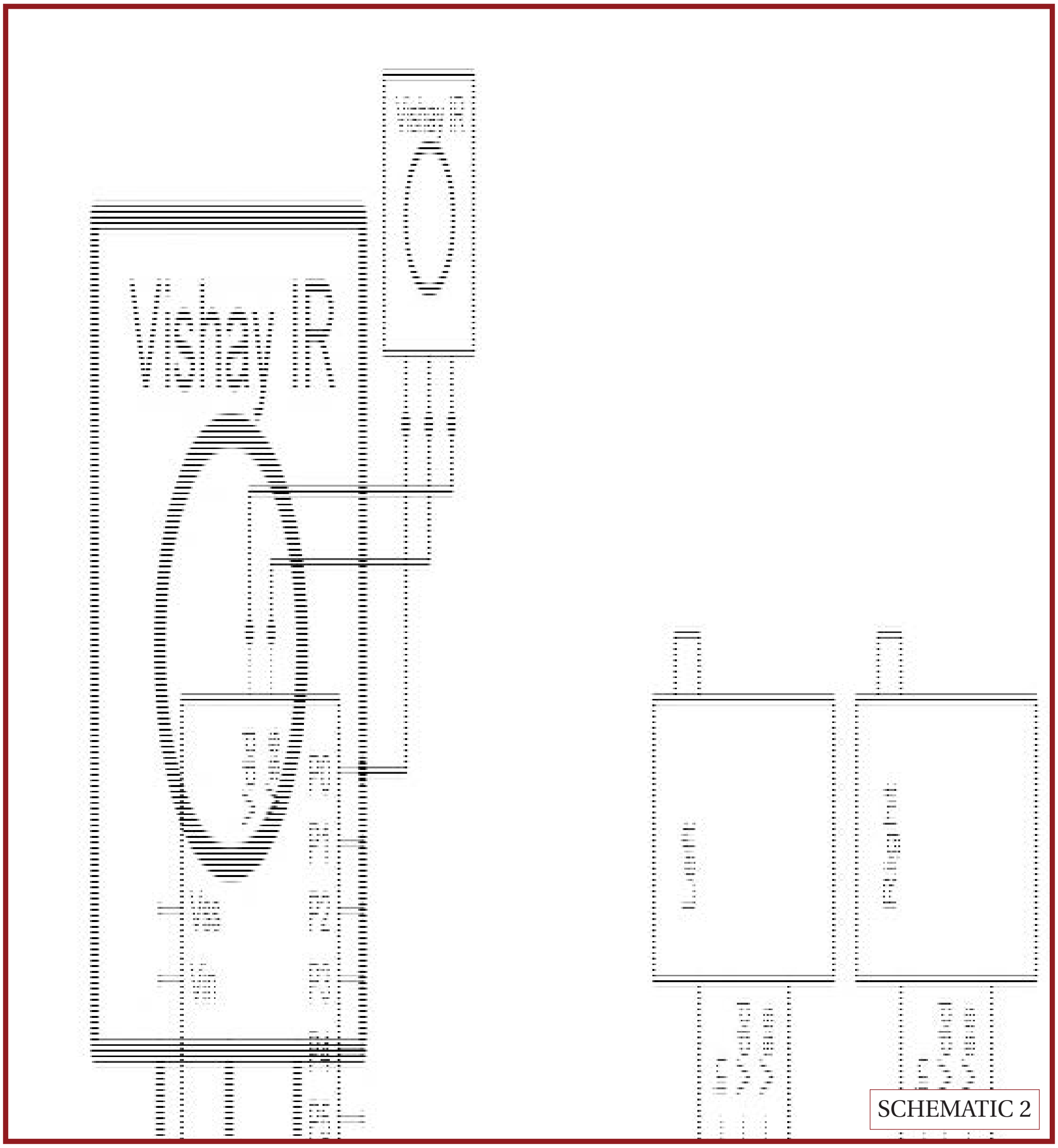
MISCELLANEOUS
- One, 5" x 7" x 1/8" thick Plexiglas, compressed PVC, or hobby plywood. You can pick one of these up at most home or craft centers.
- Two, Wheels. These can be foam wheels that you can pickup from any hobby store. You can also use surplus wheels like those found on baby carriages.
- Four, 1-1/2" #4 standoffs
- Four, 3/4" #4 standoffs
- Two, 1/4" #4 standoffs
- 16, 3/8" #4 machine screws
- Two, #4 hex nuts.
- I purchased all my standoffs and #4 hardware from Jameco Electronics at **www.jameco.com**.

other end of the header into the breadboard, then wire data and power leads as shown in Figure 13. You will also use the breadboard to wire the Vishay IR module.

Schematic 2 shows the connection details of the AX-12 network and the IR module.

Testing the AXbot

I have included two programs to get you started in programming the AXbot. The first program called AX12IRbot.txt uses the channel and volume pads on an IR remote to move the AXbot. The remote can be any Sony compatible remote. If you are using a universal remote, just set it to Sony TV or VCR mode. Make sure you set the lservo and rservo constants at the beginning of the program to match the two AX-12s you are using.



SCHEMATIC 2

The AX12IRBot2.txt program is a bit more advanced. This program adds ramp logic to smooth the speed transitions of the AXbot. Feel free to experiment with the ramp routines to fine-tune your own AXbot.

What's Next

We have barely taxed the power of the AX-12. Actuators like the AX-12 are the future of robotics. Next month, I will show you step-by-step how to build a giant BioCrab using 18 AX-12s.

If you are into robotics, animatronics, or anything that requires a high power servo, I recommend picking up a couple AX-12s. The price of just under $45 is pretty much what you would pay for an advanced servo with less holding power and no feedback.

All the example programs, as well as the source, are available for download at **www.kronosrobotics.com/Projects/ax12a.shtml**. **SV**

The Vex Challenge

Coming to a School Near You

by Bryce and Evan Woolley

At the end of last year, the Rhode Island Science & Technology Advisory Council announced its goal of bringing the FIRST Vex Challenge program to all of the 67 public high schools, charter schools, and career/technical centers in the state of Rhode Island. Governor Donald Carcieri has voiced enthusiastic support for the program, and the Science & Technology Advisory Council describes the Vex Challenge as a natural fit with their mission of promoting science education and preparing students for the jobs of tomorrow.

This statewide institutionalization of a FIRST educational robotics program is the first of its kind in the nation, and as such, is a great success for Dean Kamen's vision of revitalizing the image of engineering in education and a success for anyone invested in the future of technology and progress in this nation.

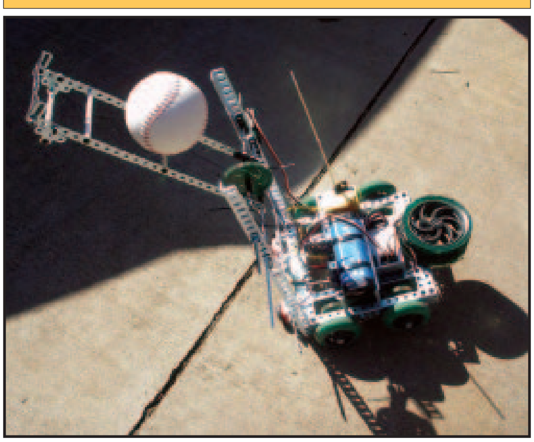
All told, the Rhode Island program is estimated to cost the state $140,000. For a program of this magnitude and importance, such a price tag seems quite reasonable. A hundred thousand dollars is a drop in the bucket to most State budgets, especially for our home state of California (which ranks among the top 10 economies in the world by itself), but which does not have such a program for FIRST of its own yet — even though the second biggest industry in California is aerospace (the third being entertainment).

One of the key descriptors in the FIRST Mission is "accessible," and Rhode Island's curricular incorporation of the Vex Challenge is the closest realization of that ideal yet. When we brought a FIRST team to our high school (which was one of the first FIRST teams in our county), our network of support was lacking. It was difficult to get the word out to fellow students, since they had never heard of the program. Finding faculty support was a challenge, since the program was foreign to our teachers.

If it was not for the enthusiastic support and advocacy of our parents — who originally told us about the program and championed the effort at our school (where our mother is an employee) — we may never have become involved in the first place. That was no fault of our school district, nor of our teachers, or of the FIRST program itself.

Although the FIRST program had been in existence for 13 years, it just had not expanded to our area yet. The challenge of the inception of our team made the ensuing experience no less enjoyable or educational, but too often such a challenge proves too daunting for others. That is why Rhode Island's initiative is so important and even necessary. FIRST is a program that ought to be in every school and such ardent state support is an instant catalyst for the expansion that is critical to the program's accessibility.

A VEX ROBOT.



Solid institutional support from the government and the state helps to encourage the formation of new teams and prevent the attrition of those struggling for lack of support in their more immediate spheres (the school or the district). Initiatives like the one in Rhode Island create a statewide network of mutual reciprocity, cooperation, and support among teams and schools, which exemplifies FIRST values like gracious professionalism.

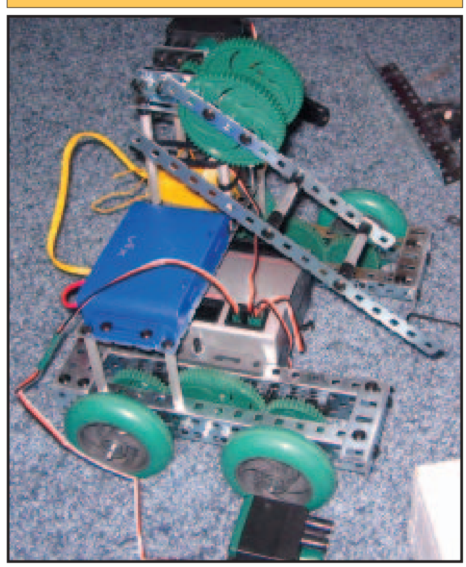
A Conversation With Dean Kamen

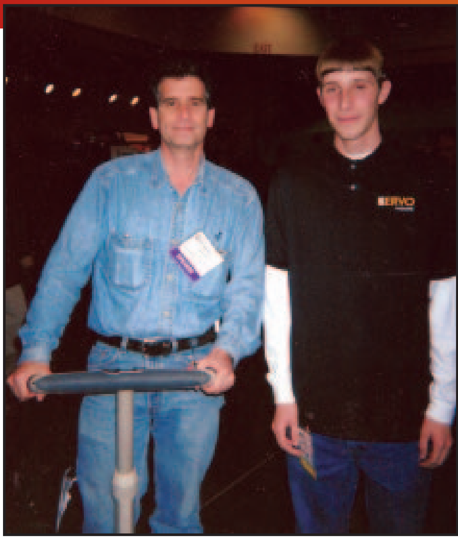
The Rhode Island initiative is a momentous landmark in the history of FIRST and we had the opportunity to interview FIRST founder Dean Kamen to hear his thoughts on the milestone and his concerns for the future.

How did Rhode Island come to be the first state to adopt such an initiative?

"Congressman Langevin [Rhode

VEX ARMWORK.





DEAN KAMEN AND EVAN WOOLLEY.

Island state senator] (who happens to use an iBOT) came to an event and then asked me, like everyone else who goes to an event for the first time, 'why isn't this in every school in the country?' Rhode Island may only have 55 public high schools, but I have been looking for someone to throw down the gauntlet to commence a race between the governors.

"We want to make this a high profile but fun win-win race between the governors and states. Last year after the finals, I worked with the Governor and other people down there to organize a summit for Rhode Island to get science and technology in the schools because a lot of the other industries down there, like fishing, have been beginning to disappear. So Rhode Island can be the first to say 'we have FIRST in every school.' We have received help from Brown University, lots of FIRST teams, and alums from FIRST teams. There are congresspersons saying 'Okay, we are going to make sure that we put enough money in the state budget so we can fund every school to have a FIRST team.'

"But I will never be satisfied. I want to raise the bar, and keep raising the bar. I want to talk to more governors [at the Kickoff]. People need to realize that FIRST is fun, the press needs to realize this. We need to get attention to FIRST in fun and serious ways. We need the support of more governors and senators to dramatically increase the number of high schools. I want FIRST to go from being in 1,500 high schools to being in 150,000 high schools."

What challenges face the expansion of FIRST and the future of science and technology in this country?

"The problem is not money, and not resources. We are the richest country in the world. The problem is cultural. It is a free country, so you get what you celebrate. Our culture celebrates sports and entertainment, more so than any other culture in the world. Most other cultures revere knowledge, education — things that make them successful.

"Our culture believes in its own bravado; we've been drinking our own kool-aid too long. They believe a football game is fun while — for reasons I will never understand — science is cast as tedious and unexciting. We have a society that doesn't get it. Perhaps it is the non-technical background in government. Perhaps we have lived too long with stereotypes of scientists as weird sociopathic nerds in white lab coats off in the corner.

"We need to get them to understand that FIRST is not your typical science fair. When you talk to people about a science fair, they think of a plastic paramecium on a lab table in the basement of a middle school, then they go home and flip on the Super Bowl. Attend event — become an advocate. Become mainstream through media attention.

"Frustratingly, everything good about science and technology happens quietly, and the only thing that makes the front page news is Britney Spears.

"A problem is that FIRST has no depth in the media — we have breadth; it may be a mile wide in its media exposure, but it is only an inch deep. There isn't enough depth. FIRST needs to be just not a mile wide, but a mile deep in its media exposure. We need more than just the lip service, but also real coverage of the successes and stories of those involved. Whole inner city schools can have better graduation rates, yet it seems it is of no place of consequence in public media. The hurdle is overcoming the superficial nonsense and overcoming the bias of stereotypes.

"We need to reenergize America. A hundred years ago, the heroes of America were the Edisons and the Wrights of society; now our society takes all of that for granted. We don't convince minority kids that they can be the next century's Orville Wright.

"Nanotechnology can change the way the material world looks and feels; we need kids to focus on that, to see it as their future. Looking at kids in other cultures, who continually thirst for education, makes you realize that if we don't change quickly, the light will go out here and it will not go on very easily again.

"Technology allows us to create our world; if there was ever a time we could not give up on being number one in technology, it is now. Before, the United States could rely on its own natural resources to be number one in the world. Those days are over, and it is a good thing, as long as we can continue to develop our technology to compensate for that. But the problem is that we are stagnating, and with natural resources running out, it creates a perfect storm and we have to get out of it.

"We get what we celebrate. We need to make a kid believe 'wow, I really can do this, science and technology is accessible and it is fun.' We need to change what they do with their lives. Our media culture has them aspire to bouncing a ball or being on stage. 39 million kids a day get hundreds of images a day of sports and Hollywood, but rarely images of people that create or invent. We are not showing them what life is like for people who think [for a living]. So it is not surprising why our culture looks like it does. We need to create situations and opportunities for our kids — people have to understand that we are competing for the minds and hearts of our kids.

"You have to talk about science and technology the same way they talk about sports in the sports magazines — they don't show the knee surgeries, the 5 am mornings, and that only one in a hundred makes it. Our culture shows all the fun and exciting highlights in virtually everything except in science and technology. All they show about science is kids asking 'why am I doing all this calculus homework?' With coverage in *SERVO* you can get a lot of technical gee whiz, but work hard to appeal to a broader public audience; appeal to someone else aside from the geeks that we already have on our team. Make parents think that 'I want my kids to do this,' go to a fun cultural event rather than go to a field and watch their kids kick each other.

"On TV these days, we have sports every hour of the day, reality TV nonsense; we have *ER* and *LA Law* — but why isn't there LA Lab? Why aren't there shows that show kids that it is exciting to understand the world around you and that it is exciting to create it? Engineering needs to be something that kids are excited about, something that parents want their kids to do. There are millions of jobs out there in science and technology; kids just need to want it to have it."

A New Political Ally

At the time we interviewed Dean Kamen, he was preparing for the Kickoff of the 2007 FIRST season. A special guest at that event was Governor Linda Lingle of Hawaii, who had come across the country to see what all the buzz was about. She wanted to see the program for herself, and as Dean Kamen commented, "anyone who leaves Hawaii to go to New Hampshire in the winter must be committed." She graciously offered to share her thoughts about the FIRST program and Dean Kamen's vision with us.

"We need to get science and technology to be something that students talk about. We need to gather enough information to convince our colleagues how FIRST should be on our national governor's agenda.

"This is my first time coming to a kickoff event and I am amazed. This is a remarkable program and should be in every high school in the nation.

"I have had conversations with Dean [Kamen] and Jim [Langevin] and I hear their desire to get FIRST to a broader audience; the governors can do that.

"Hawaii needs this technology to have the standard of living continue to improve, which is part of my agenda. FIRST fits into that agenda.

"We need to bring this to the governors. Each state has different priorities and each approaches these issues differently. But every governor recognizes the importance of promoting innovation to improve the standard of living."

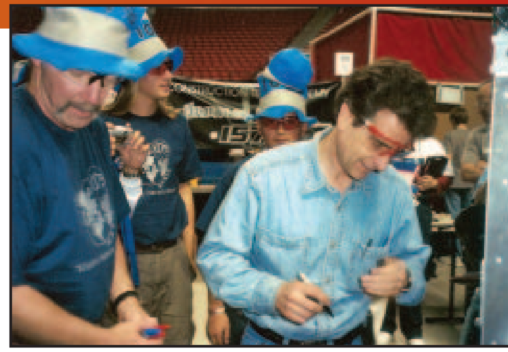
Concluding Discussion

As Dean Kamen pointed out, the challenges that face FIRST and the future of science and technology in our society are not financial, economic, or even political — but cultural. The gross domestic product of the United States rivals that of the entire European Union, and the US is by far the richest single nation in the world. Our insatiable capitalist economy demands developments in science and technology to sustain itself. Political agendas across the nation recognize the importance of innovation to uphold and improve the standard of living, and as with Congressman Langevin of Rhode Island and Governor Lingle of Hawaii, attendance to a single event can engender the strongest political advocates.

The greatest challenges that face FIRST are cultural — what Dean Kamen calls "the great American lie" — a vacuous media driven culture that hypes superficial material pursuits over intellectual cultivation and enrichment, coupled with the bias of negative stereotypes about the field of science and technology.

But what will motivate kids to go into science and technology in the first place? As the experience of thousands of students across the nation who have participated in FIRST testifies, science and technology is already a fun and interesting thing to do. The mentality of a hundred years ago that idolized scientific inventors and technological innovators was never really lost, although it has been overshadowed by today's superficial preoccupation with the media, sports, and entertainment. A cultural shift has indeed taken place, but it is not irreversible or inevitably ruinous.

We need to socialize the nation's youth to be just as readily exposed to positive images of fun and rewarding lives in science and technology as they are currently exposed to the images of glamour and fame in a life of movies or sports. The Rhode Island initiative is a great step toward that exposure, because every kid in the school system will have access to these positive images via FIRST. And as such, they will have access to the images of today's heroes of science and technology, like Dean Kamen, who can inspire them to live a life of thought, innovation, and fulfillment through education.



DEAN WITH TEAM 1079.

Dean Kamen's Challenge

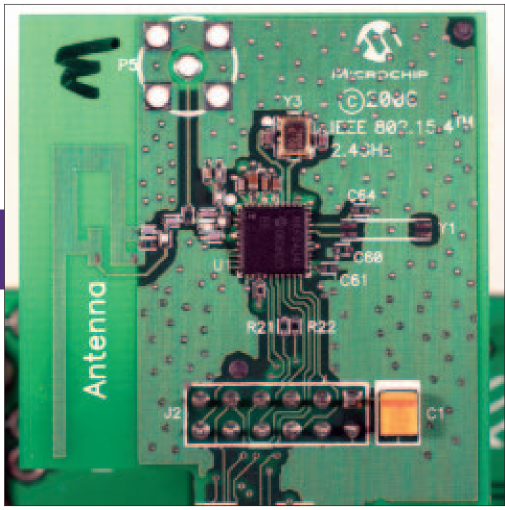
Dean Kamen's challenge to FIRST teams across the nation is to help make the program more mainstream. Encourage other schools in your area to form teams of their own. Invite them to your team if they are unable to. Tell your local newspapers about the amazing accomplishments that FIRST enables and inspires you to achieve. Contact your representative or congressperson and tell them that you want to see an initiative like the one in Rhode Island put on your state's agenda.

Becoming involved in this program, or in the promotion of science and technology in general, does not just have to do with pursuing a professional career to make your parents happy — the future prosperity of our country depends on it. This is not meant to be a daunting moral obligation; it can be fun and fulfilling in its own right to help others around you. Help keep the light of inspiration burning brightly.

While the cultural icons of today may survive in tattered baseball cards or in old syndicated reruns, the heroes of science and technology will always be important to the society they helped create and improve and will live on through their ideas and inventions. SV

Low Power Robot Communications

PART 2



by Peter Best

TO DO LIST:

- **Write our MRF24J40 daughterboard initialization code.**
- **Establish a data packet format.**
- **Write our MRF24J40 transmit and receive routines.**
- **Code up and run a small test application.**

If you missed Part 1, the To Do list above represents the tasks that we need to complete to get our MRF24J40-based IEEE 802.15.4-2003 nodes talking to each other. Hopefully, you have completed the construction of the MRF24J40 host hardware as we're going to dive right into the details of the PIC18LF4620 firmware.

I used the HI-TECH PICC-18 C compiler, the MPLAB IDE, and an MPLAB ICD2 to create the MRF24J40 firmware we're about to discuss. However, since HI-TECH PICC-18 C source is super portable, you will be able to use my C source code as a template for other PIC C compilers or PIC languages such as Basic. I've included a complete MRF24J40 host firmware source code listing that you can download from the *SERVO* website (**www.servomagazine.com**). So, grab the MRF24J40 host source code and follow along.

Initializing the MRF24J40 Host Hardware

Many of the aliases used in the code are defined in a file called MRF24J40_defs_A.h. Since we will be writing for a pair of MRF24J40 units — A and B — there is also an MRF24J40_defs_B.h definitions file, which contains address and alias information for the second "B" MRF24J40 host. Both of the A and B host include files also contain pertinent addressing information and a map of the MRF24J40 registers. Here's a set of aliases we'll use to represent the MRF24J40 control pins and their PIC18LF4620 general-purpose I/O counterparts:

```
#define MRF24J40_CS              LATC0
#define MRF24J40_CS_TRIS         TRISC0
#define MRF24J40_RESETn          LATC2
#define MRF24J40_RESETn_TRIS     TRISC2
#define MRF24J40_WAKE            LATC1
#define MRF24J40_WAKE_TRIS       TRISC1
```

When functions are not the most efficient means of accomplishing a task, we will turn to a number of macros, which are defined within the main body of the MRF24J40 application. For instance, the following SPI chip select macros will be used heavily by the SPI-based MRF24J40 register read and write functions:

```
#define MRF24J40_CS_1()\
{\
   MRF24J40_CS = 1;\
   GIEH = savedBits.bGIEH;\
}

#define MRF24J40_CS_0()\
{\
   savedBits.bGIEH = GIEH;\
   GIEH = 0;\
   MRF24J40_CS = 0;\
}
```

This macro — found in the HardwareInit function — begins the MRF24J40 host hardware initialization process:

```
#define SPIInit()   SSPIF = 1
```

The PIC18LF4620's SPI interface must see SSPIF set to a logical high before any SPI operations can commence.

We must disable and deselect the MRF24J40 to prepare it for configuration. To do this, we only need to set the MRF24J40_CS and MRF24J40_RESETn pins to a logical high. I pointed out earlier that the MRF24J40_CS and MRF24J40_RESETn lines are actually PIC18LF4620 PORTC general-purpose I/O pins. So, this is all we need to do to disable the MRF24J40:

```
    MRF24J40_CS              = 1;
    MRF24J40_RESETn          = 1;
    MRF24J40_CS_TRIS         = 0;
    MRF24J40_RESETn_TRIS     = 0;
```

With those four lines of code, we have instructed the PIC18LF4620 to place general-purpose I/O pins RC0 and RC2 in output states and set their values to a logical 1.

The MRF24J40 will use its INT pin to interrupt the PIC18LF4620 whenever a frame is received and stored into its receive FIFO. The MRF24J40-generated interrupt will take the PIC18LF4620's RB0 pin from a high to low level to indicate the interrupt condition. Thus, we must tell the PIC18LF4620 to expect a falling edge as interrupt activity on its RB0 pin. This is easily done with a single line of C code:

```
INTEDG0 = 0;
```

INTEDG0 is actually a bit within the PIC18LF4620's INTCON2 (Interrupt Control Register 2) that indicates the interrupt as a falling edge on the PIC18LF4620's RB0 pin.

At this point, we've only partially initialized the PIC18LF4620's SPI interface. We still need to define the PIC18LF4620's SPI interface pins and their I/O directions. From the PIC18LF4620's point-of-view, the SCK and SDO SPI lines are outputs with the PIC18LF4620's SDI line being a data input fed by the MRF24J40. The SCK line should idle in a logically high mode and thus is set to a logical 1 along with the SDO line in the code that follows:

```
LATC3          = 1;   // SCK
LATC5          = 1;   // SDO
TRISC3         = 0;   // SCK
TRISC4         = 1;   // SDI
TRISC5         = 0;   // SDO
```

The LATCX instructions actually set the respective output latch on the PIC18LF4620 port pin. For instance, general-purpose I/O pins RC3 and RC5 will read logically high following the execution of the LATCX instructions. In the PIC world, writing to the TRISXX registers determines if a general-purpose I/O pin will be an input (1) or an output (0).

The MRF24J40 requires data to be clocked into its SPI input pin on the rising edge of SCK, which means the MRF24J40 will send data on the falling edge of SCK. We set the PIC18LF4620's SPI interface to reflect the MRF24J40 ruling by writing a 0xC0 to the PIC18LF4620's SSPSTAT register. PIC18LF4620's SPI interface is then enabled by filling the PIC18LF4620's SSPCON1 register with 0x20:

```
SSPSTAT = 0xC0;
SSPCON1 = 0x20;
```

Tending to the MRF24J40's LEDs and pushbutton switches is a must because if we don't set the LEDs up to blink or get the PIC18LF4620 ready to sense the pushbuttons, this whole thing is for naught (as Jethro Bodine would say). The pushbutton-LED code is simple and looks like this:

```
ADCON1 = 0x0F;   //no analog inputs
TRISA = 0xE0;    //PORTA RA0-RA3 = output
RBIF = 0;        //clear interrupt on change flag
RBPU = 0;        //enable weak pullups on PORTB inputs
TRISB4 = 1;      //enable S1 switch input
RISB5 = 1;       //enable S2 switch input
```

Since we won't be using PIC18LF4620's analog-to-digital converter, we must configure all of the PIC18LF4620 analog input pins as digital I/O pins. That is done by simply writing 0x0F to the PIC18LF4620's ADCON1 register. With the analog inputs inactive, we can now set the PORTA pins attached to the LEDs (RA0 and RA1) for output operation.

The pushbutton switches will be sensed using the PIC18LF4620's interrupt on change feature, which works only on PORTB inputs RB4-RB7. Interrupt on change works like this: the state of the inputs RB4 through RB7 are initially memorized by the PIC18LF4620 with the pushbutton switches open. If any of the input states on RB4 through RB7 change (i.e., a pushbutton is depressed), an interrupt is generated, which is signaled by the RBIF flag. The interrupt on change handler code reads the PORTB inputs and determines which switch was depressed. We can then use the pushbutton information to make a decision in our application code. The interrupt on change interrupt handler code is very simple and looks like this:

```
if(RBIF)
{
    portb_data.Val = PORTB; //read PORTB switch states
    if(portb_data.bits.b5 == 0)
    switch_status.RB5_toggled = 1;  //S2 depressed
    if(portb_data.bits.b4 == 0)
    switch_status.RB4_toggled = 1;  //S1 depressed
    RBIE = 0;             //disable interrupt on change
    LATB = PORTB;         //memorize PORTB state
    RBIF = 0;             //clear int on change flag

}
```

The final piece of the hardware initialization process involves firing up the RS-232 port. The most labor-intensive piece of bringing up a PIC's RS-232 port is the calculation of the baud rate divider. I used the services of the compiler to do the math and the rest falls into place as shown here:

```
#define BAUD1   57600    //desired baud rate
#define FOSC    16000000 //oscillator frequency
#define DIVIDER1 ((unsigned int)(FOSC/(16 * BAUD1) -1))
SPBRG = DIVIDER1;        //load baud rate divisor
TRISC7 = 1;              //receive pin = input
TRISC6 = 0;              //transmit pin = output
TXSTA = 0x24;            //high speed baud rate-enable TX
RCSTA = 0x90;            //enable serial port RX and
                         // serial port pins
```

As you can see, the configuration of the PIC18LF4620's RS-232 port is just a matter of filling in the correct EUSART (Extended Universal Synchronous Asynchronous Receiver Transmitter) control registers. With the PIC18LF4620's general-purpose I/O ready to feed the MRF24J40 control registers, we can now move on to the initialization of the MRF24J40's PHY.

Prepping the MRF24J40

The initialization of the MRF24J40 flows much like that of the PIC18LF4620. Most all of the things we do to set up the MRF24J40 will only be done once. To make the task of configuring the MRF24J40 easier, many of the required control register settings are power-up default values. For instance, we really want to operate using MRF24J40-generated CRC (Cyclic Redundancy Check or checksum) values in the transmit FIFO. The value of the TXCRCEN (Transmit CRC Enable) defaults to enable the MRF24J40's transmit CRC engine. The same holds true for the RXCRCEN bit, which power up with receive CRC checking enabled.

Unless you have specific interest in certain settings of the MRF24J40 PHY, you only need to know how to turn on the MRF24J40's PLL (Phase Locked Loop), set the MRF24J40 transmit output power level, and select the operating channel. Performing the aforementioned operations is a simple matter of writing the desired data into the desired MRF24J40 control register. This is all laid out for you in the PHYInit function, which is part of the C source code download.

Since the MRF24J40 will be performing address resolution duties in its MAC engine, we must assign addresses to the MRF24J40 host modules. Every IEEE 802.15.4-compliant device must have a unique 64-bit address. In the case of the MRF24J40, this 64-bit address is kept in the MRF24J40's EADRX register set. The 64-bit address for the MRF24J40 "A" node is:

```
#define MAC_LONG_ADDR_BYTE7 0x00
#define MAC_LONG_ADDR_BYTE6 0x04
#define MAC_LONG_ADDR_BYTE5 0xA3
#define MAC_LONG_ADDR_BYTE4 0x00
#define MAC_LONG_ADDR_BYTE3 0x00
#define MAC_LONG_ADDR_BYTE2 0x00
#define MAC_LONG_ADDR_BYTE1 0x00
#define MAC_LONG_ADDR_BYTE0 0x41     //ASCII "A"
```

While the 64-bit address for the MRF24J40 "B" node is:

```
#define MAC_LONG_ADDR_BYTE7 0x00
#define MAC_LONG_ADDR_BYTE6 0x04
#define MAC_LONG_ADDR_BYTE5 0xA3
#define MAC_LONG_ADDR_BYTE4 0x00
#define MAC_LONG_ADDR_BYTE3 0x00
#define MAC_LONG_ADDR_BYTE2 0x00
#define MAC_LONG_ADDR_BYTE1 0x00
#define MAC_LONG_ADDR_BYTE0 0x42     //ASCII "B"
```

Loading up the MRF24J40 EADRX register set is very easy to do. The extended address bytes are located in the MRF24J40's Short Address Control Register area. We've already written the routines necessary to read and write the MRF24J40's Short Address RAM area. So, all we have to do is apply the Short Address RAM area write function as follows:

```
    WriteShortRAMAddr(EADR0, MAC_LONG_ADDR_BYTE0);
    WriteShortRAMAddr(EADR1, MAC_LONG_ADDR_BYTE1);
    WriteShortRAMAddr(EADR2, MAC_LONG_ADDR_BYTE2);
    WriteShortRAMAddr(EADR3, MAC_LONG_ADDR_BYTE3);
    WriteShortRAMAddr(EADR4, MAC_LONG_ADDR_BYTE4);
    WriteShortRAMAddr(EADR5, MAC_LONG_ADDR_BYTE5);
    WriteShortRAMAddr(EADR6, MAC_LONG_ADDR_BYTE6);
    WriteShortRAMAddr(EADR7, MAC_LONG_ADDR_BYTE7);
```

That takes care of the IEEE 802.15.4-2003 64-bit extended address. However, we really don't want to use the 64-bit address as it takes up too much space in our messages. Instead, we'll define a set of 16-bit short addresses and use them in our packets. We'll also establish a PAN identifier of 0xCAFE. Here's what that looks like for MRF24J40 Host "A":

```
#define DESTADDR_LSB        0xBB    //destination address
#define DESTADDR_MSB        0xBB
#define PANID_LSB           0xFE    //PAN ID
#define PANID_MSB           0xCA
#define SHORTADDR_LSB       0xAA    //source-local address
#define SHORTADDR_MSB       0xAA
```

And, MRF24J40 Host "B":

```
#define DESTADDR_LSB        0xAA    //destination address
#define DESTADDR_MSB        0xAA
#define PANID_LSB           0xFE    //PAN ID
#define PANID_MSB           0xCA
#define SHORTADDR_LSB       0xBB    //source-local address
#define SHORTADDR_MSB       0XBB
```

Again, we apply our Short Address RAM area write routines to put the 16-bit address information and PAN ID into the correct MRF24J40 registers. It's a walk in the park:

```
void SET_ShortAddress( void )
{
      WriteShortRAMAddr(SADRL,SHORTADDR_LSB);
      WriteShortRAMAddr(SADRH,SHORTADDR_MSB);
}
void SET_PANId( void )
{
      WriteShortRAMAddr(PANIDL,PANID_LSB);
      WriteShortRAMAddr(PANIDH,PANID_MSB);
}
```

All that's left to do before we can seriously consider transmitting and receiving is activating the interrupts. We must enable the MRF24J40 INT line on PIC18LF4620's RB0, enable the interrupt on change interrupt, and turn on the PIC18LF4620's global interrupt engine. Piece of cake:

```
    INT0IE = 1;          //enable RB0/INT
    RBIE = 1;            //enable interrupt on change
                         // interrupt on PORTB RB4-RB7
    IPEN = 1;            //enable priorities on
                         // interrupts
    GIEH = 1;            //enable all interrupts with
                         // the priority bit set
```

At this point, the MRF24J40 Host modules are ready to rock and roll.

Data Packet Format

Rather than reinvent the wheel, we'll use the IEEE 802.15.4-2003 approved packet layout. Since we can only send a total of 127 bytes in an IEEE 802.15.4-compliant frame, the data layout is rather simple. A transmit packet begins with a security header length, which will be 0x00 for our purposes as we are not implementing any security. The byte following the security header length represents the actual length of the packet, not including the length byte or the pair of CRC bytes that get tacked onto the packet automatically by the MRF24J40. Two bytes of Frame Control information follow the length byte. The Frame Control bytes tell the receiver how to handle the packet. A sequence number follows the Frame Control bytes. The sequence number is an identifier used in the acknowledgement process. We won't request an acknowledgement in our application. However, an acknowledgement can be requested by setting a bit inside of the Frame Control bytes. Data follows the sequence number and a pair of CRC bytes completes the packet. To keep things simple, we'll only transmit one byte of data, which represents the status of the pushbutton switches. Thus, we can "can" our transmit sequence as follows:

```
WriteLongRAMAddr(0x000, 0x00);   //security header length
WriteLongRAMAddr(0x001, 0x0C);   //length of the packet
WriteLongRAMAddr(0x002, 0x01);   //Frame Control
WriteLongRAMAddr(0x003, 0x88);   //Frame Control
WriteLongRAMAddr(0x004, sequence_byte++); //seq byte
WriteLongRAMAddr(0x005, PANID_LSB); //destination PAN ID
WriteLongRAMAddr(0x006, PANID_MSB);
WriteLongRAMAddr(0x007, DESTADDR_LSB); //dest address
WriteLongRAMAddr(0x008, DESTADDR_MSB);
WriteLongRAMAddr(0x009, PANID_LSB);     //source PAN ID
WriteLongRAMAddr(0x00A, PANID_MSB);
WriteLongRAMAddr(0x00B, SHORTADDR_LSB);  //src address
WriteLongRAMAddr(0x00C, SHORTADDR_MSB);
WriteLongRAMAddr(0x00D, switch_data);    //data
```

The packet gets written into the TXFIFO, which begins at location 0x000 in the Long Address RAM area. On the receiving end, the transmitted packet is stuffed into the MRF24J40's RXFIFO beginning at address 0x300 of the Long Address RAM area. The bytes in the RXFIFO are stacked identically to the way they were ordered in the transmitter's TXFIFO with a couple of exceptions. There is no security header length byte. Thus, the first byte in the RXFIFO is the packet length byte. The CRC bytes are followed by a link quality index byte (LQI) and a received signal strength indicator (RSSI) byte. We won't use them in our application, but you can look at them if you pull them out of the RXFIFO.

Send It

You're going to love this! Once we write the packet to the TXFIFO, we send it like this:

```
//set TXRTS bit in TXNMTRIG register
WriteShortRAMAddr (0x1B, 0x01);
```

Receive It

The MRF24J40 received the packet we coded up, passed it through address checking, checked its CRC, stuffed it into the receiver's RXFIFO, and generated an interrupt. The receiver's interrupt handler code checks to see if the interrupt is a receive packet interrupt, clears the interrupt mechanism, reads the length byte, and pulls the packet from the RXFIFO into the PIC18LF4620's waiting receive buffer, which we carved out in our application code (char RxBuffer[RX_BUFFER_SIZE];).

I stopped the receiving MRF24J40 Host node after it captured a packet I kicked off by depressing one of the pushbuttons on the other MRF24J40 Host module. What you see in Figure 1 is a capture of the PIC18LF4620 receive buffer contents, which happens to contain the transmitted packet. Note that our pushbutton data will always be found at offset 12 of the packet. In the case of Figure 1, the 0x20 data byte tells us that I punched S2, which is connected to PIC18LF4620's RB5. If you're into reading hexadecimal dumps, you'll also discern that the packet originated from the "B" node as the destination address is 0xAAAA. If reading dumps is not what you yearn to do when you wake up, I simultaneously captured the transmission of the packet with Daintree Networks SNA, a ZigBee-IEEE 802.15.4 packet sniffer/analyzer software package. The Daintree Networks SNA breakdown of the packet is shown in Figure 2. Just in case you were wondering about that Frame Control byte pair, the Frame Control bits are demystified in Figure 2.

Use It

Sending the pushbutton status across the Ether is a cool thing to do. However, it is a useless process unless we use the data in some fashion. Our little application simply senses a pushbutton depression and sends a data byte containing the status of the switches. The receiving node receives the data byte and illuminates the associated LED. Both the "A" and "B" MRF24J40 Host nodes run identical application code. That means when you press S1 on node "A," you will illuminate the LED connected to RA0 of the node's PIC18LF4620 and extinguish the LED connected to RA1. If you depress S2 on node "B," you will turn on the RA1 LED and blow out the RA0 LED on node "A." Remember that you also have an RS-232 serial port at your disposal on each of the MRF24J40 nodes that can be called into action, as well. For instance, we could code up an application that takes incoming data from the node's RS-232 port, process it, and transmit it.

Our LED control application is shown in Listing 1. The MRF24J40_Enable function contains the code sequences that initialize the PIC18LF4620 hardware, the MRF24J40 daughter board, and set up the node's addresses. In actuali-

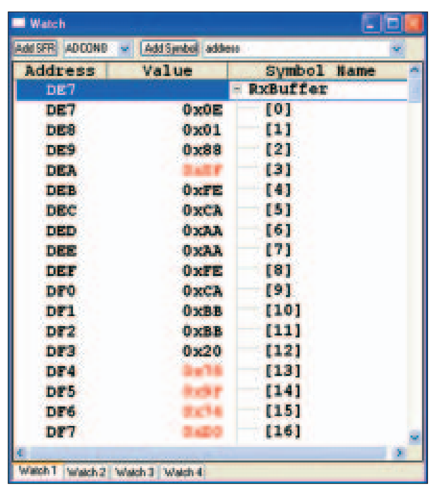


FIGURE 1. Since this is a "canned" message, our data will always be located at offset 0x0C.

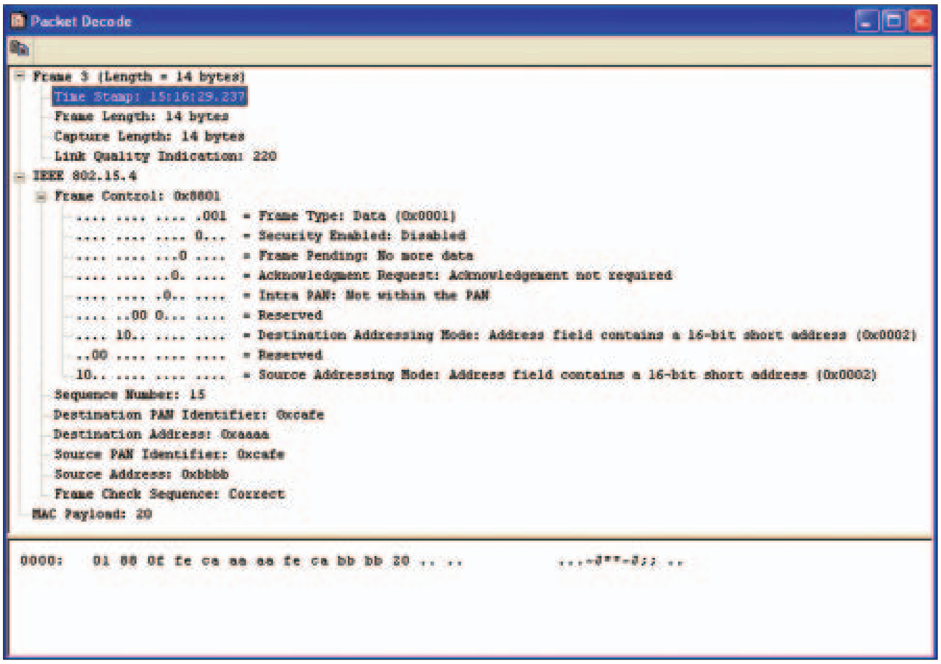


FIGURE 2. Being able to match up the Daintree Networks SNA window contents with the MPLAB ICD2 receive buffer dump contributes lots towards understanding what is really going on under the covers.

```
void main(void)
{
    unsigned int timer;

    MRF24J40_Enable();
    printf("\r\nMRF24J40 ONLINE\r\n");
    sequence_byte = 0x00;
    switch_status.RB5_toggled = 0;
    switch_status.RB4_toggled = 0;

    do{
        if(switch_status.RB5_toggled)
        {
            switch_status.RB5_toggled = 0;
            switch_data = 0b00100000;
            Transmit_Frame();
            for(timer=0; timer<0xFFFF; ++timer)
                NOP();
        }
        if(switch_status.RB4_toggled)
        {
            switch_status.RB4_toggled = 0;
            switch_data = 0b00010000;
            Transmit_Frame();
            for(timer=0; timer<0xFFFF; ++timer)
                NOP();
        }
        if(RxBuffer[0x0C] == 0b00100000)
        {
            LATA0 = 0;
            LATA1 = 1;
        }
        if(RxBuffer[0x0C] == 0b00010000)
        {
            LATA0 = 1;
            LATA1 = 0;
        }

        for(timer=0; timer<0xFFFF; ++timer)
            NOP();
        RBIE = 1;

    }while(1);
}
```

LISTING 1. Nothing works here until we've done our homework on the supporting functions.

ty, we could ignore the sequence byte as we are not using acknowledgements. However, it is good to drive the sequence byte as you may want to include the use of acknowledgements in your application.

The states of the pushbutton switches are ultimately represented by bits within the switch_status structure. The interrupt on change interrupt handler code sets the bits according to which pushbutton is depressed, assembles the data byte, and transmits the packet containing the pushbutton status data byte. The data byte is always located at offset 0x0C in the packet and the receiving node simply reads the data byte and manipulates the LEDs according to the value of the data byte. This is so simple even a caveman could code it.

Build On It

We have taken the most basic of IEEE 802.15.4-compliant processes and sent a meaningful data byte wirelessly between a pair of IEEE 802.15.4-compliant nodes. You are not limited to a single data byte as you can pack up to a total of 127 bytes (102 data bytes minimum) into the packet you want to transmit. The less address information you send, the more data you can cram into your packet. If you only want to communicate within the same PAN, you can turn on the Intra PAN bit in the Frame Control word and not send the source PAN ID address bytes.

The application we've just walked through is the basis for IEEE 802.15.4-2003 and ZigBee communications networks. All we have to do to move into the IEEE 802.15.4-2003 world is to embrace the IEEE 802.15.4-2003 protocols and apply them to our basic transmit and receive routines. To get to ZigBee, we simply layer the ZigBee protocol on top of the IEEE 802.15.4-2003 protocol layers. The ZigBee layer is what you know as a ZigBee stack.

As I've shown, you really don't need a complicated ZigBee stack to use the services of IEEE 802.15.4-compliant transceiver modules. The bottom line is that if you need to put a low-cost, low-power wireless connection to work, you now have the knowledge to make it happen at the lowest of levels. And, don't worry about building up your own MRF24J40 daughterboard. You can buy them all day from Microchip Direct. Easy-to-build Microchip hardware coupled with easy-to-code firmware makes for easy wireless interconnects. **SV**

Source

Microchip — **www.microchip.com**
MRF24J40 IC, MRF24J40 daughterboard, PIC18LF4620, MPLAB ICD 2

Contact the Author

Peter Best can be reached via email at peter@nerdvilla.com.

ROBOT SIMULATION: *Physics*

by Bryan Bergeron

Video games such as the popular Mech Warrior series achieve realism through 3D graphics, sound, and — more importantly — believable physics. The giant robot tanks groan under the stress of gravity, projectiles glance off of rock faces and opposing tanks at the correct angles, and explosives create damage and debris that's appropriate for the placement of the charge and the material — all because of realistic game physics. The software and hardware tools used by the game industry to create physically accurate virtual worlds are relevant to robotics because the same tools can be used to simulate the motion and handling of real robots.

Last month, Part 1 of this series provided an introduction to simulation technology, with an emphasis on AI behaviors. This second installment continues the exploration with a focus on the tools and techniques available for modeling and simulating the physics of robots. Read on to learn how — armed with a PC and a few inexpensive software tools — you can work with realistic simulations of the same shape-shifting rovers under development by NASA or robots of your own design.

Physics Simulation

Robotics is a test bed for theoretical AI, machine vision, and virtually every field of engineering because many of the algorithms and designs that perform flawlessly on paper fail to work in practice. Real-world factors such as gravity, friction, momentum, mechanical vibration, and electrical noise play havoc on real robot control systems, sensors, effectors, and platforms. Recognition of the value of accurate robot physics simulation isn't new. Engineers have relied on kinetic models and simulations for decades to perfect the design of one-of-a-kind robots. However, until recently, these systems lacked the fidelity of even the cheapest modern PC and console games, and their use was limited to a select few scientists with access to mainframe and minicomputer systems and the mathematical prowess to harness the processing power.

Fortunately for roboticists, the multi-billion dollar video game industry has invested heavily in developing technologies that enable non-mathematicians to develop and deliver robot simulations that provide realistic physical responses to their environment. Depending on whether the robot is an airborne, underwater, or ground vehicle, simulating the physics of robot movement and interaction may involve calculating air and water resistance, gravity, collisions, center of gravity, rotation, rigid body dynamics, static and dynamic friction, inertia, wave motion, buoyancy, viscous drag, or angular momentum. Accurately simulating the physics of even the simplest carpet rover requires consideration of the robot's mass, velocity, acceleration, torque, and restitution (amount of bounce), and may draw upon Newtonian Physics, Euclidian geometry, calculus, and matrix algebra.

To illustrate the challenge of accurately and quickly simulating robot physics, consider the collision of a differential drive robot and a wall, as shown in Figure 1. Simulating the action of the robot requires consideration of friction between the robot and wall, the resultant torque on the robot body, and the angular velocity of the robot as it rebounds from the wall. These three factors describe how kinetic energy is dissipated and transformed when the robot collides with the wall.

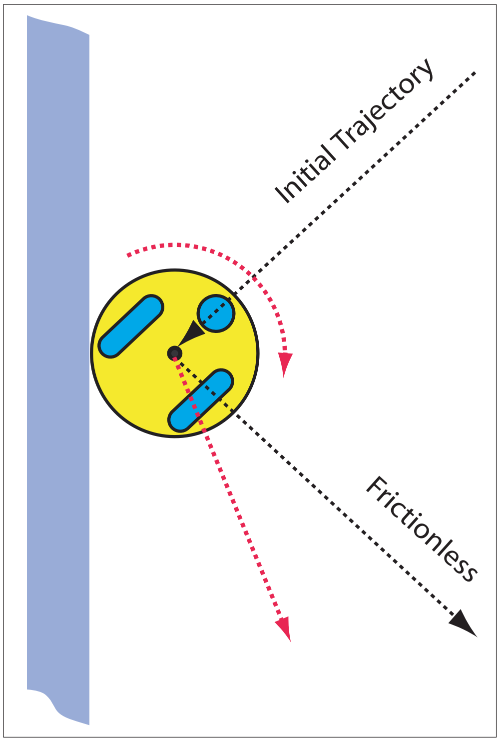


FIGURE 1. Robot-wall collision with and without friction.

To simplify the discussion, let's

focus on the relatively simple robot-wall interaction, and assume that the robot is gliding over a frictionless surface. A more complete — and more computationally intensive — model would incorporate factors such as the friction from the wheels, the gyroscope effect caused by the spinning wheels, the effect of differential power applied to the wheels, the robot's center of gravity, and characteristics of the floor. As readers experienced with SumoBot competitions know, simply cleaning the rubber wheels of a robot with rubbing alcohol prior to an event can markedly improve the handling characteristics of the robot.

When the robot initially contacts the wall, it experiences a friction force that acts tangentially to the robot's surface. This tangential force alters the linear velocity of the robot (the red arrow in Figure 1) and creates a torque on the robot (red arc in Figure 1), altering its angular velocity, relative to what would be expected in a frictionless world.

Quantifying the changes in linear and angular velocity is challenging because the robot and wall will deform on impact, creating additional resistance. We can create an approximate model by assuming the robot and wall are rigid and that the static coefficient of friction is insignificant compared with the kinetic coefficient of friction. With these simplifications, the change in linear velocity is a function of two variables: the kinetic coefficient of friction and the impulse force normal to the surface of the wall.

Because the friction force acts on the surface of the robot, it creates a torque about the robot's center of gravity that causes the robot to spin clockwise. The key to determining the new angular and linear velocities is to first solve for the impulse force, which is the force generated during the brief time of the collision. The general equation for impulse force (*J*) involving collision of two bodies takes the form [1]:

$$J = \frac{-v_r(e+1)}{\frac{1}{m_1} + \frac{1}{m_2} + \mathbf{n} \bullet \left[\left[\frac{\mathbf{r}_1 \times \mathbf{n}}{I_1} \right] \times \mathbf{r}_1 \right] + \mathbf{n} \bullet \left[\left[\frac{\mathbf{r}_2 \times \mathbf{n}}{I_2} \right] \times \mathbf{r}_2 \right]}$$

In this case, n is the normal vector at the point of contact, r_1 is the distance from the robot center of mass to the point of contact, and r_2 is the center of mass of the wall; m_1 is the mass of the robot, and m_2 is the mass of the wall. I_1 is the robot's moment of inertia, I_2 is the wall moment of inertia, and v_r is the relative velocity of impact.

Note that this is a general equation that is applicable to the collision of two 3D bodies, such as two robots. In this example, the second body is a wall. Relative to the robot, the mass (m_2) and moment of inertia (I_2) of the wall are huge, and their reciprocals are effectively zero. Even if the reciprocal values are set equal to zero, solving the impulse force equation in real time without additional mathematical simplifications and approximations would bring a multi-core, multi-processor PC to its knees.

There are volumes describing programming tricks that can be used to minimize the computational load of solving equations in real-time, starting with substituting integer for floating point math. Obviously, working at this level of detail isn't for the mathematically squeamish. However, unless your goal is to develop new low-level mathematical routines, working at this level probably isn't the best use of your time. It is important to understand the underlying physics principles so that you can decide which factors to consider and which to ignore in creating your robot simulations.

Tools

Roboticists typically work with tools that provide a level of abstraction above the raw mathematical operations. MobotSim, Webots 5, and MatLab/Simulink, which were introduced in Part 1, provide varying levels of abstraction and physical simulation fidelity. MobotSim provides the highest level of abstraction and the lowest fidelity. Although it's the most painless of the three programs to work with in generating simple behaviors, there is no physics engine or library. As a result, the mathematics of all physics algorithms must be developed in Basic.

Webots 5 is more capable, especially when multiple robots must be simulated. Although working with the program requires fluency in C/C++, you can achieve excellent physics fidelity. It's an obvious choice if you happen to own one of the robots the program is pre-configured to simulate. For other robots, there may be better choices.

MatLab and Simulink are particularly powerful tools to model and simulate the physics of robotics motion. The learning curve can be a bit steep if you're not accustomed to working with matrices, and even with the add-on toolboxes, you'll have to understand the underlying mathematics. However, once mastered, MatLab and Simulink can be used to simulate robot physics to whatever level of detail you require in your work.

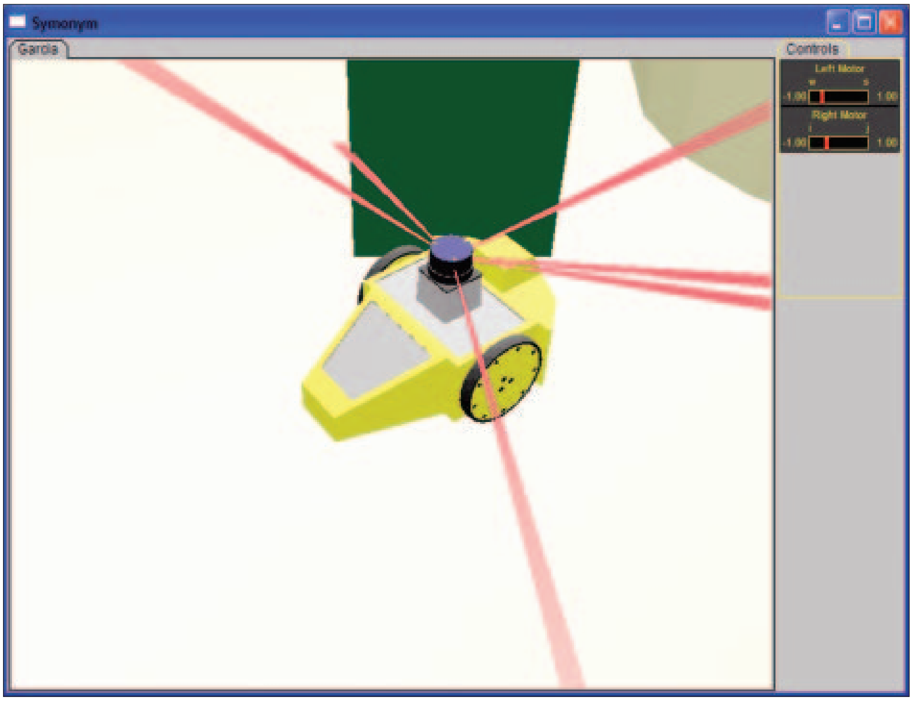
To illustrate the range of tools available for simulating robot physics, three additional tools are presented here: Symonym, an example of a proprietary robot simulator with physics support; Dark Physics, an example of a physics engine used to extend the functionality of a video game shell; and Microsoft Robotics Studio Simulation, a videogame-like environment that's part of the Microsoft Robotics Studio. As described below, these tools hide much of the mathematical detail and allow you to focus on the robotics.

Symonym

Symonym — a sophisticated robot simulation environment with 3D physics — is designed as an aid for researchers developing realistic simulations of robots and robot swarms. Acroname — one of my favorite robot parts suppliers and the developer of Symonym — also uses the simulation environment to create realistic demonstrations of its sensors and robot platforms. Although the Symonym SDK is priced beyond the hobby market, the player and several simulations are available as free downloads (**www.symonym.com**).

A particularly good example of simulated robot physics is the Garcia robot fitted with the Hokuyo laser rangefinder, as shown in Figure 2. Even if you never plan to invest $2,700 on the laser rangefinder, the simulation is worth running to experience the difficulty in manually steering a differential drive robot.

FIGURE 2. Acroname's Symonym environment showing the Hokuyo laser rangefinder in operation aboard a Garcia robot.



Using your keyboard, you can steer the robot past several obstacles in a room.

Thanks to the underlying physics calculations, the simulated robot has inertia, and if you rub up against a wall, the friction will torque the robot into the wall. If you steer the robot too vigorously, you can flip it onto the laser rangefinder — something you wouldn't want to duplicate on a real robot. The graphics component of the simulation illustrates the planar coverage provided by the rangefinder. The only thing you might miss in Symonym is the whirring of DC motors and the thud normally produced by a real robot running into obstacles.

Game Engines

As described in the previous article, game engines designed for video game development provide a rich environment for developing robot simulations. In particular, many game engines are designed to accept optional physics software engines and hardware acceleration. As shown in Figure 3, a game engine typically takes multimedia assets — the 3D models, planar images, and sound — and uses them to create a realistic 3D environment. Under Windows, most of the media manipulations are handled by calls to DirectX. Instead of manipulating image data at the byte level, programmers make calls to the graphics toolbox provided by DirectX.

Most modern video cards incorporate a graphical accelerator or graphics processing unit (GPU) to offload some of the mathematical operations associated with rendering 3D scenes from the CPU to the GPU. On a Windows machine, the architecture of the GPU is optimized for the image manipulations supported by DirectX, such as mapping texture maps onto 3D meshes. Because the GPU is optimized for specific graphics operations, the graphics quality is typically much higher than would otherwise be possible with a given CPU and memory.

In much the same way, physics engines provide a toolbox of physics routines that can be used by the game engine to provide more realistic 3D output. These routines include operations on vector spaces, including the addition, multiplication, transposition, and inversion of matrices, which are the basis for 3D translation, scaling, and rotation [1].

A physics processing unit (PPU) — a relatively recent addition to the PC gaming world — offloads physics manipulations from the CPU, freeing main processor cycles for other activities. A PPU enables you to specify more objects and more robots in an environment, and to maintain simulated physics accuracy at faster than real-time interactions.

The only consumer-level PPU on the market is the PhysX chip by AGEIA, which also sells the AGEIA Physics Engine — conveniently, the only engine that can realize the benefit of the PPU. I use the PPU manufactured by BFG, shown in Figure 4 which AGEIA claims can support 533,000 complex collisions/second or 22 billion instructions per second. Translating these numbers to speedup depends on the configuration of your PC, how the AGEIA Physics Engine is called, and the complexity of the robot environment. On my Alienware Area 51 3.87 GHz Pentium 4 with 2 GB RAM and a GeForce 6800 with 245 MB RAM, the speedup factor is about 2X running demos accompanying Dark Physics.

AGEIA is not forthcoming on basic specifications of the PhysX chip, such as power consumption. However, the BFG version of the PhysX card is power hungry. In addition to power from the PCI bus, it requires a feed from a stan-

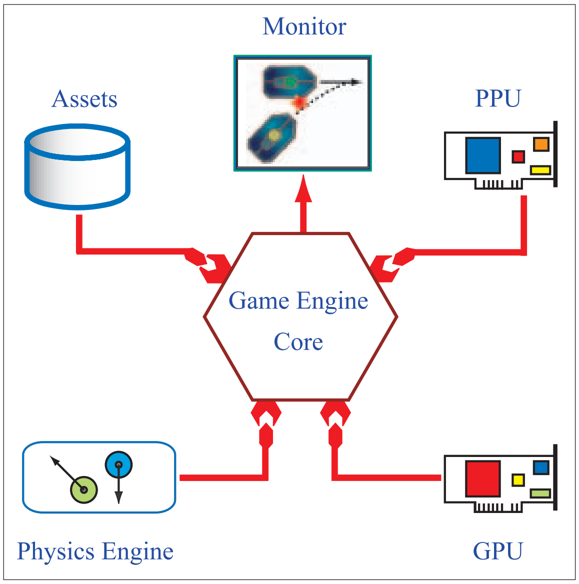


FIGURE 3. Game engine showing an add-on physics engine and physics processing unit (PPU).

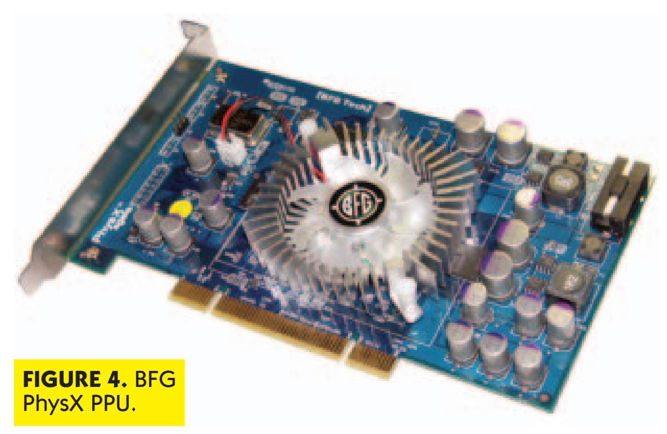


FIGURE 4. BFG PhysX PPU.

dard four-pin Molex power cable. I measured current at 2.7 Amps through the power cable. The card is also noisy, in that it adds another fan to a PC system. Luckily, you don't have to own a PhysX chip to run simulations that use the AGEIA Physics Engine. The runtime engine is available as a free 20 MB download from the AGEIA site (**www.ageia.com**).

You can explore the power of the AGEIA Physics Engine by purchasing Dark Physics, a $30 add-on DLL for the Dark Physics Professional game engine. The power of the physics engine is that it provides low-level physics control using high-level statements. Together with the other elements of the game engine, you can create visually and physically accurate 3D graphics, sound, advanced lighting techniques, and texture mapping (see Figure 5).

To illustrate ease-of-use, consider that specifying the global gravity vector within Dark Physics requires a single line of code — phy set gravity x#, y#, z# — where x#, y#, and z# are acceleration in m/sec^2 along the standard x, y, and z axes. For example, normal earth gravity — "g," approximately 9.8m/sec^2 — is defined as:

```
phy set gravity 0,-9.8,0
```

Simulating the physics of a Martian rover would entail using a y value of -3.77 or -1.62 to simulate the gravity on the moon.

As another example of the utility of the physics engine, the smooth plastic shell of the 4WD robot shown in Figure 5 can be defined as:

```
phy make material id, "RobotShell"
phy set material dynamic friction id,0.2
phy set material static friction id,0.3
phy set material restitution id,0.2
```

To simulate the physical properties of plastic, the values assigned dynamic and static friction and restitution are relatively low. The lower the restitution, the more energy is dissipated and the less the robot will rebound from an impact. A robot shell with a higher level of restitution would lose less energy and will be reflected more on impact.

For comparison, consider the handful of Dark Physics statements required to define the physics of the brick wall surrounding the robot course in Figure 5:

```
phy make material id, "BrickWall"
phy set material dynamic friction id, 1.0
phy set material static friction id, 25.0
phy set material restitution id, 0.0
```

In this example, the static friction assigned to the brick wall is significantly greater than what we assigned to smooth plastic. Furthermore — as with real brick — there is virtually no restitution. The value assigned to dynamic friction is also greater than what we assigned to smooth plastic.

Modeling a joint is more involved than simple friction, but still mathematically painless. Consider a cylindrical joint attaching a wheel to the axle of a robot or connecting a rotating gripper to the body of a robot arm. Such a joint can be specified in Dark Physics with the following command:

```
phy make cylindrical joint joint id, rigid body a,
rigid body b, anchor x#, anchor y#, anchor z#, axis
x#, axis y#, axis z#
```

where *joint id* is the identification number of the joint, *rigid body a* is the first rigid body to be attached, and *rigid body b* is the second rigid body. Assuming the bodies involved (e.g., an axle and wheel) are non-deformable simplifies the underlying mathematics. In reality, the axle will flex somewhat under load, and the wheel/tire even more so.

The anchor points — *anchor x#, y#, and z#* — define the relative translational movement between the two bodies along a single axis, such as when the bodies are allowed to slide along the axis of the joint. Relative rotation along each axis is specified by *axis x#, y#, and z#*. In the case of a simple wheel-axle joint, there is rotation along a single axis, and there is minimal translational movement.

As a point-of-reference, the simulation shown in Figure 5 was developed with less than 100 lines of Basic code, including the routines for front wheel steering and for defining the environment. With a physics engine, the major challenge of designing a robot simulation with a game shell is that of developing the graphic assets, especially the 3D articulated model of the robot.

The easiest way to obtain a 3D model of a robot is to

FIGURE 5. 4WD *SERVO* robot attacking a ramp, developed in Dark Basic Professional with the Dark Physics engine.

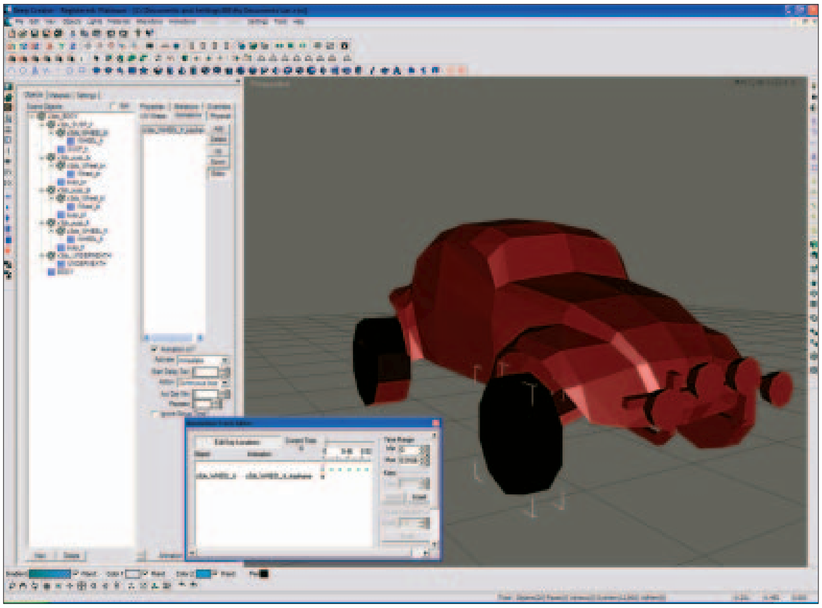


FIGURE 6. Customizing front wheel animation in a 3D mesh modeling program.

repurpose one of the many free or low-cost vehicle models available through sources such as The Game Creators, Renderosity, TurboSquid, and 3DRT. Figure 6 shows a 3D modeling environment in which the articulation of the front wheel is being defined.

There are dozens of 3D modeling programs suitable for designing new robots or modifying existing designs, ranging from shareware to commercial products costing several thousand dollars. Professional animators will recognize programs such as Max and Maya, and most lower-cost applications can at least read files developed with these standard tools.

My favorite entry-level 3D modeling program is Caligari Game Space Light. The program is fully functional, but limited in the number of file types supported and the maximum level of detail. However, it's more than adequate for designing a hexapod, wheeled robot, or robot arm.

Microsoft Robotics Studio Simulation

No discussion of robot physics simulation would be complete without mentioning the latest and perhaps eventually most significant competitor in the arena — Microsoft Robotics Studio Simulation (MRSS). MRSS is part of the Microsoft Robotics Studio (MRS), an integrated robotics development system.

The MRS supports a wide variety of programming options, including the .Net languages C# and VB.Net and the scripting languages IronPython, JavaScript, as well as VPL (Microsoft's Visual Programming Language). VPL is particularly impressive, in that it provides a graphic programming environment akin to the LabView-light software that accompanies the LEGO NXT. However, VPL seems powerful enough to develop serious robot controllers.

The simulation examples provided by Microsoft in the October 2006 release of the studio are written in C#. The Microsoft visual studio express edition of C# — like the other express editions and the robot studio environment — is available for free download.

If you have experience with JavaScript, much of C# will seem familiar. Furthermore, like the other visual studio languages, the Microsoft C# compiler creates code that runs on a virtual machine, the Common Language Runtime (CLR). With a few extra steps, you can compile C# to native machine code for maximum performance.

Because the Ageia Physics Engine is callable in the Microsoft Robotics Studio Simulation development environment, defining the physics properties of materials resembles the routines used in Dark Physics. For example, defining the height, texture, restitution, dynamic friction, and static friction associated with the ground is expressed in C# as:

```
void AddGround()
{
    HeightFieldEntity ground = new HeightFieldEntity(
        "ground", // name
        "groundtexture.dds", // texture file
        new MaterialProperties("ground",
        0.3f, // restitution
        0.4f, // dynamic friction
        0.6f) // static friction
    );
    _simEnginePort.Insert(ground);
}
```

As with Dark Basic Professional and many other PC-based game engines, object textures are stored in DirectDraw Surface (DDS) format, which is Microsoft's file format for textures and cubic environment maps. I use the freeware Photoshop DDS plug-in, one of several DDS utilities available from the nVidia developer website (**www.developer.nvidia.com**), to create DDS files from TIFF or JPEG images.

A scene rendered in MRSS runtime is shown in Figure 7. The control panel, including the steering mechanism for the Pioneer P3-DX robot, is in the lower right corner of the display. Of particular note is the laser rangefinder data window in the lower part of the control panel. It shows the Martian globe to the left of the robot. By using the keyboard or Xbox360 control pad, you can practice driving a simulated Pioneer robot around the environment. Even if you never get the chance to operate a real $35,000 Pioneer P3-DX, you can learn a lot about differential drive robots and laser rangefinders.

In addition to specifying the physics of the robots and objects in MRSS, you can easily manipulate textures with Photoshop or other image editors. As you'll see, changing texture maps or 'skins' is a quick and painless way to customize the appearance of your robots and environments. For example, I used Photoshop to edit the original texture files, which were of a gray carpet and of the earth, to a metal deck and the globe of Mars. This example also highlights one of the issues of developing your own graphics assets — that of establishing a standard for scale. You'll want to

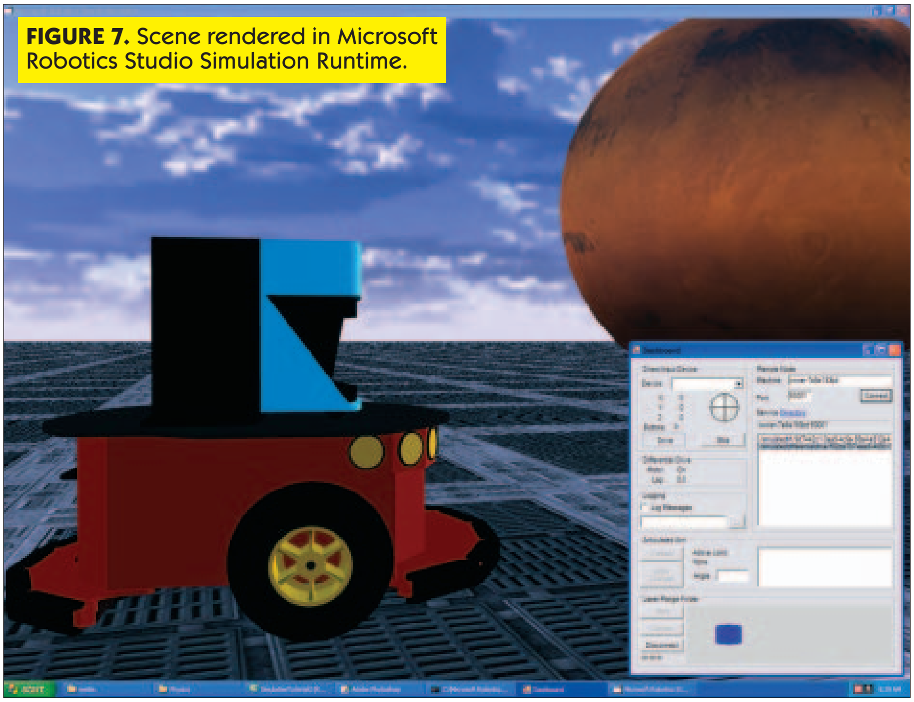


FIGURE 7. Scene rendered in Microsoft Robotics Studio Simulation Runtime.

create robots and other objects of the same relative scale so that physical interactions are as realistic as possible.

While the MSSR is Spartan by game engine standards, it does support the Ageia PhysX chip and graphics accelerator cards that follow the DirectX standard. In addition, the simulation tutorials that accompany the studio are limited to simulations of wheeled robots — the Pioneer P3-DX shown on Figure 6 and a robot developed with the LEGO NXT system — with only a few onboard sensors.

The simulation environment isn't as powerful as a full-fledged game engine, but when you consider it's integrated with an environment capable of controlling robots through serial or wireless connectivity, the MSSR has amazing potential. Although I've found the Bluetooth connectivity between MSR and the LEGO NXT is problematic, I've had success with an A7 Engineering Bluetooth transceiver and the Serializer board from the Robotics Connection.

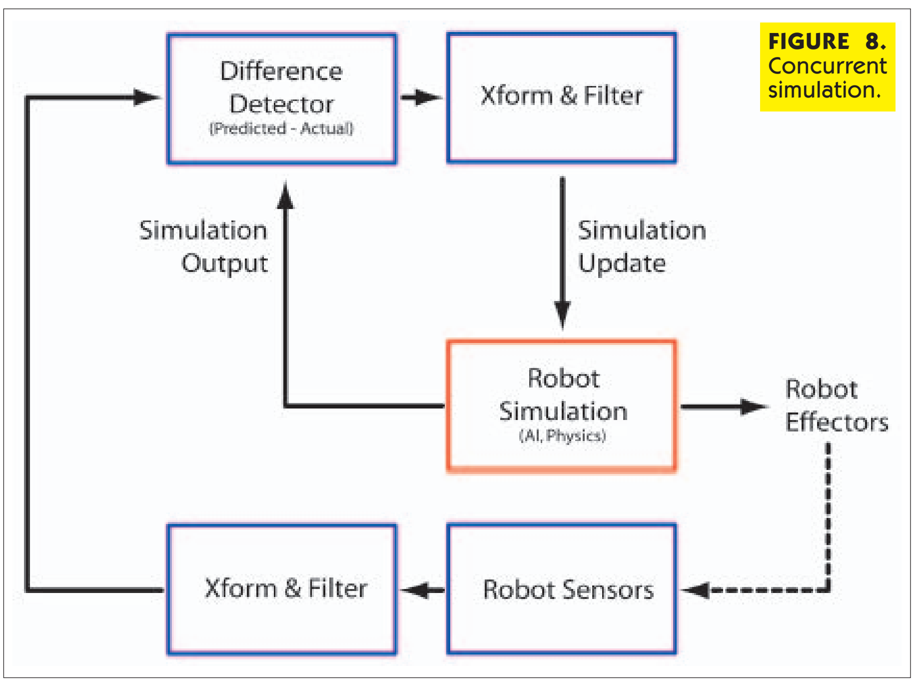


FIGURE 8. Concurrent simulation.

From Here

With the tools described above, you can design any sort of robot that you can imagine with only a PC. Moreover, you don't have to limit your exploration to problems such as optimum sensor placement on dual-drive robots. The tools can be used to simulate swarms of robots, animatronics, or the operation of robot arms.

One of my current projects involves concurrent simulation — using a robot physics simulation as a predictive tool or supervised learning module for a physical robot in real time, as shown in Figure 8. The goal of this simulation technique is to enable the robot to plan ahead, using real-time sensor data to update the simulated world. Think of it as virtual reality for robots.

The challenge of developing concurrent simulation highlights many of the limitations of simulating robot physics. When real-time performance is critical, there is a trade-off in accuracy, precision, and response time. Physics Processor Units and multi-core PCs can partially address this issue by allowing the execution of more complex models in the same amount of time.

Furthermore, some sensors are difficult to model because of the complexity of a model that considers all relevant factors. Consider the ubiquitous sonar rangefinders. A realistic model of a robot navigating with a sonar rangefinder should incorporate an acoustic model of the environment that models the absorption of sound energy by certain materials and almost complete reflection by others. Then there's the issue of how to model a swarm of sonar-equipped robots, with each robot flooding the environment with sonar pulses. The problem of resolving signal from noise quickly becomes unmanageable.

Whether you want to tackle the sonar problem or simply perfect your robot design, there are several resources available. If you're comfortable with examples in C/C++, try *Physics for Game Developers* [2], which is part of the O'Reilly series. For a review of concepts such as impulse and torque, *The Feynman Lectures on*

Physics [3] provides an approachable review. If you decide to create your own 3D robot graphics, there are several online resources worth visiting, including **www.gamasutra.com** and **www.gavedev.net**. To explore C# and the Microsoft Robot Development environment, a good place to start is the 10-hour *Absolute Beginner's Video Series*, available for free download from Microsoft.

Next time, we'll look at how to use simulation at the system level for applications such as PID design and tuning. **SV**

References

[1] Dalmau, D., *Core Techniques and Algorithms in Game Programming.* 2004, Indianapolis: New Riders.

[2] Bourg, D., *Physics for Game Developers.* 2002, Sebastopol, CA: O'Reilly & Associates.

[3] Feynman, R., R. Leighton, and M. Sands, *The Feynman Lectures on Physics, The Definitive Edition Volume 1.* 2005, Reading, MA: Addison-Wesley Publishing Company.

Resources

Absolute Beginner's Video Series
Msdn.microsoft.com/vstudio/express/visualcsharp/learning

Affordable 3D Models
3DRT.com and **turbosquid.com**

Ageia PhysX Engine — available through *BFG Tech*
www.BFGTech.com

Dark Basic Professional, Dark Physics, Dark AI, Styx, and Low-cost Models
The Game Creators
www.TheGameCreators.com

GameSpace Light
Caligari Corporation
www.caligari.com

MatLab/Simulink
The MathWorks
www.mathworks.com

MobotSim
MobotSoft
www.mobotsoft.com

Serializer Board for the Microsoft Robotic Studio
The Robotics Connection
www.roboticsconnection.com

Symonym Player
www.symonym.com

Webots 5
CyberBotics, LTD
www.cyberbotics.com

Beginner's Robotics on $50 a Month

- by Eddy Wright and John K. Jellman

PART 4
Servos, Sonar, and a Second Microcontroller: Seeing With Sound

Amateur robotics is FUN.

Amateur robotics is EDUCATIONAL.

Amateur robotics is EXPENSIVE!

By now, your CIRCBot has become more capable with additional sensors and the ability to display messages on an LCD screen. This month, we will enhance CIRCBot even more by adding a sonar sensor and a servo to rotate it. Plus, we will show you how multiple microcontrollers can communicate with each other and work together.

Servos

Servos are small devices used to mechanically control the position of something. In RC cars, they are often used to control steering. In robotics, they are used in many different ways; from opening and closing simple grippers to acting as muscles in more sophisticated biped robots. For CIRCBot, we will use a servo to rotate a sonar sensor. This allows us to point the sonar sensor in multiple directions, allowing us to sweep any entire area. BASCOM-AVR provides built-in support for servos, making programming the movements simple.

Sonar

The sonar sensor we are using is the MaxSonar-EZ1 from MaxBotix as shown in Figure 1. This small sensor is perfect for robots needing to measure

MojoBus (by Brian Dean of BDMICRO LLC; www.bdmicro.com)

For quite a while, I have experimented with various bus protocols for use on robots. The goals of MojoBus are simplicity of sending and receiving messages and consistency of format between varying devices. A single packet format is supported, which is:

[>]>t[,f]:cmd;[cmd;cmd;...]

Items inside [] are optional. A linefeed ends the message, where:

t = bus id of recipient (required)
f = bus id of sender (optional)
cmd = command to execute

Command format:
parameter | parameter=value

Commands can be read-only or read-write, depending on whether it makes sense or not. I've defined a basic set of commands that all devices should support. These are:

RESET = ro, reset device
VERSION = ro, display firmware revision
ANNC = ro, display identity string, delayed by bus ID order
WHO = ro, display identity string
ID = rw, set controller bus ID
BAUD = rw, set RS232/RS485 serial baud rate, units = bps

The person who implements this can define whatever commands they want, of course, depending on what makes sense for the device they build. The idea is simply so that one can put an H-bridge on the bus, as well as other devices like sensors, etc., and be able to communicate with them all in a consistent way that is not too complicated yet not too constraining. For example, MojoBridge, the high level controller interface to the RX50, supported the following additional commands:

D = rw, set duty cycle -100 to 100, units = %
ENCP = rw, set encoder sample period, units = ms
TPOS = rw, set motor target position, units = encoder
TVEL = rw, set motor target velocity, units = encoder
APOS = ro, display actual motor position, units = encoder
AVEL = ro, display actual motor velocity, units = encoder
KP = rw, set PID P constant (float)
KI = rw, set PID I constant (float)
KD = rw, set PID D constant (float)
PID = rw, set PID P, I, and D constants (float)
REV = rw, set motor reverse 0 or 1

Thus, to set the duty cycle of this device to 50%, and its bus ID is 7, one would send it the following message: >>7:d=50;

In C, this could be done something like this:

```
sprintf(buf, ">>%d:d=%d;\n", target, duty);
puts(buf);
```

Multiple commands to the same device can be strung together. i.e., >>7:d=50;rev=1; a linefeed ends the message to that device. The protocol is nice in that you can connect to a simple terminal emulator on a PC and type in commands directly and see results. This is great for testing and trying things out. Also it is easy to parse. A sender ID is supported also. If the sender sends its own sender ID, and if the command requires the recipient to produce any results, those results should themselves be in MojoBus protocol format. i.e., like this:

Sender: (request position from bus ID 7, sender is bus ID 3) >>7,3:APOS;

Recipient (bus ID 7) should respond as follows: >>3,7:*APOS=359722

The '*' indicates it is a result and not a command, just in case the sender might interpret that as a command, instead of in the case of a multi-master setup.

I hope this gives you an idea of how it works. I hope to put up some more complete documentation on my website soon.

---

distance and is very easy to use. The EZ1 automatically measures the distance to an object and can return the distance in inches as a serial message, an analog voltage, or as a pulse width signal. We will use the analog voltage method for CIRCBot using one of the analog-to-digital converters (ADC) built into the ATMega48 microcontroller.

In simple terms, the EZ1 works by sending out a short burst of sound at 42 kHz. The sound bounces off of an object and returns to the sensor, just like an echo you may hear in a large room. The EZ1 sensor measures the amount of time it takes for the sound to return and uses this information to calculate the distance to the object in inches. The EZ1 allows us to measure the distance to an object from 6" up to 254" and detect an object within 0"–254".

FIGURE 1. LV-MaxSonar-EZ1

Second Microcontroller

As mentioned in the last article, we are running low on available I/O pins on our ATMega48 microcontroller. We could squeeze the servo and sonar sensor

Description	Qty
ATMega48-20P MCU	1
LV-MaxSonar-EZ1 Sonar Sensor	1
Breadboard 400 contact (small)	1
GWS S03NXF Standard Size Servo	1
Wire Jumper Set	1
WR636 Multi-Color Stranded Wire Set	1
.1" OC Breadboard Male Header	1
.1 µF Capacitor	2
220 µF Electrolytic Capacitor	1
Plastic Shelf Clip	1
L4931 5V LDO Regulator	1
1N4001 Diode	1
.1" OC Male Header Pin Strip	1

FIGURE 2

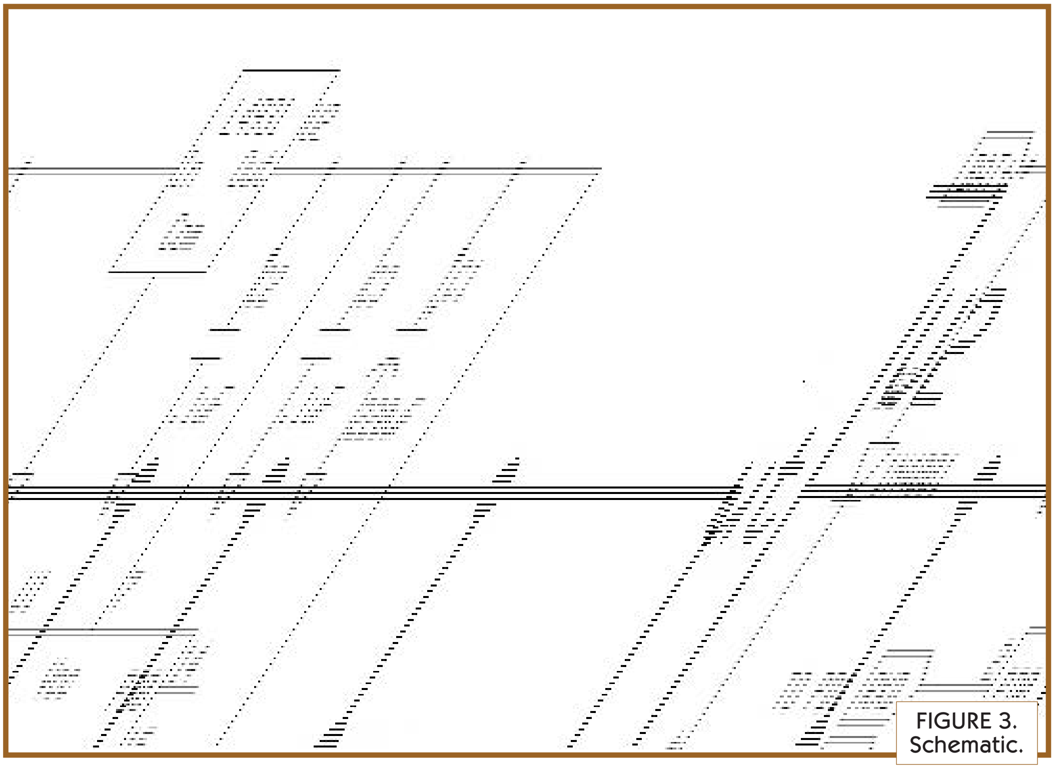


FIGURE 3. Schematic.

Item	From	To	Length/Notes
Small Orange Jumper	A3	Lower GND3	.3"/8 mm long
Small Orange Jumper	A14	Lower GND14	.3"/8 mm long
Small Orange Jumper	J15	Upper +V15	.3"/8 mm long
Small Yellow Jumper	A13	Lower +V13	.4"/10 mm long
Small Yellow Jumper	J13	Upper GND13	.4"/10 mm long
Small Yellow Jumper	J26	Upper GND26	.4"/10 mm long
Small Yellow Jumper	H18	H22	.4"/10 mm long
Small Green Jumper	C5	C10	.5"/13 mm long
Gray Jumper	G17	G25	.8"/20 mm long
Gray Jumper	J16	J24	.8"/20 mm long
Long Green Jumper	Upper GND29	Lower GND29	Cut one of the Long Green Jumpers to 1.8"/45 mm, insulation length
Long Red Jumper	Upper +V30	Lower +V30	Cut one of the Long Green Jumpers to 1.8"/46 mm, insulation length
Long Yellow Jumper	D7	F23	Cut one of the Long Yellow Jumpers to 1.9"/48 mm and bend as shown in Figure 4
ATMega48 MCU	Pin 1 at E7	Pin 15 at F20	Pin 1 has a small recessed dot by it
220 µF Capacitor	Upper +V24	Upper GND24	Gray stripe indicates Ground
.1 µF Capacitor	Upper +V18	Upper GND18	Cut excess length from leads
.1 µF Capacitor	Lower +V18	Lower GND18	Cut excess length from leads
1N4001 Diode	C1	C4	Lead with Silver band goes in C4
L4931 +5V Regulator	Pin 1 Lower +V1	Pin 2 Lower GND1	Pin 3 A1, Pin 3 Lead will need to be bent to reach A1

FIGURE 5. Breadboard Assembly.

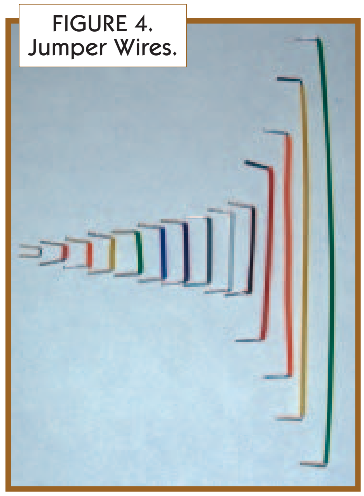
onto the existing microcontroller, but that wouldn't leave room for anything else. Instead, we will add a second ATMega48 microcontroller and connect it to the existing one using serial communications. The first microcontroller can offload controlling the servo and processing the data from the MaxSonar-EZ1 onto the second microcontroller. By adding a second microcontroller, you effectively double the I/O pins, double processing power, and provide room for future expansion.

We will use a serial communications protocol called MojoBus. MojoBus is a simple, text-based standard that allows for multiple devices to talk to each other and is easy to implement on microcontrollers (see MojoBus sidebar).

Parts in Kit

Figure 2 shows the list of parts in the kit from Wright Hobbies for this month. Figure 3 shows the schematic for the second microcontroller, the servo, and the sonar sensor. Note that we will only make four connections to the existing circuit: battery power, ground, serial transmit, and serial receive.

FIGURE 4. Jumper Wires.



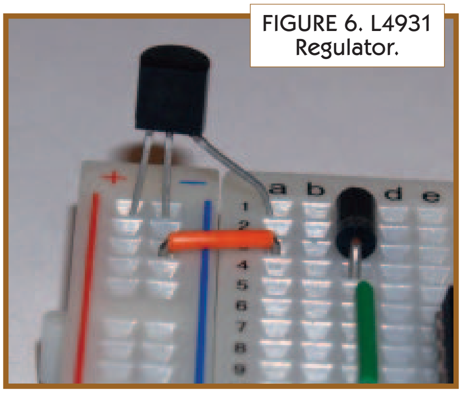


FIGURE 6. L4931 Regulator.

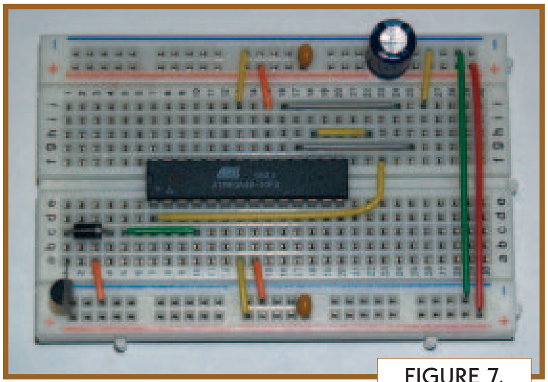


FIGURE 7. Completed Breadboard.

We will use a separate voltage regulator for the new breadboard. The L4931 is a low-voltage dropout voltage regulator which provides 5V of regulated power from as little as a 5.5V supply. Our battery pack is 6V, so this will help provide stable 5V power to the MaxSonar-EZ1 and the second ATMega48 microcontroller. The 1N4001 diode will reduce the noise that can be caused by the servo.

Assembly

We will start by populating the breadboard as it is easier to insert the parts into the breadboard before putting it on the robot. This month, we will use jumper wires designed for breadboards. They are color-coded by size and are bent to match the hole spacing in the breadboard as you can see in Figure 4. Using these will save time since only three wires will need to be cut to populate the breadboard.

Follow the instructions for wiring the breadboard in Figure 5. The lengths shown are the insulated portion of the wire and do not include the legs that insert into the breadboard. Refer to Figure 6 to see how to bend the lead on the L4931 regulator to fit the breadboard. A completed breadboard assembly appears in Figure 7.

Mounting the Breadboard

Remove the cover from the mounting tape on the back and place the breadboard next to the LCD breadboard on top of the bot. Line up the edges with the other breadboard, leaving a one-hole gap between them to allow the wires to pass through as shown in Figure 8.

Place a jumper from Upper GND30 on the new breadboard to Lower GND30 on the LCD breadboard. The blue jumper should be about the right length, but you can use a different one if needed. This jumper provides a common ground for all the breadboards. The new breadboard will also need power from the battery. Take a piece of the stranded red wire and solder a pin cut from the male header strip using the same technique as last month. Plug the pin into H60 on the main breadboard and run the wire through one of the holes in the upper plate between the two breadboards. Extend the red wire to E1 on the new breadboard, cut it to the proper length, solder another pin onto the end of the wire, and insert the pin into E1.

Next, cut two pins from the male header and solder the green and yellow stranded wires to them. Plug the header pins into the main breadboard placing the yellow wire into A32 and the green wire into A33. Run the wires up through one of the holes between the upper breadboards and extend the wires to B8 and B9 on the new breadboard. Cut the wires to length, solder a pair of male header pins onto the ends, and plug the header pins with the green wire plugging into B8 and the pin with the yellow wire plugging into B9. The wiring of the new breadboard is complete.

Mounting the Sonar Sensor on the Servo

To get the most out of our MaxSonar-EZ1, we will mount it onto a servo. The servo can rotate 180 degrees, allowing the EZ1 to view to the sides and in front of the robot.

First, open the servo package and remove the large round servo horn. This will be the base that we use to attach the EZ1. Locate the plastic shelf bracket that came with your kit. This will be used to hold the EZ1 in place. Remove the round plastic post that protrudes from the long side of the bracket using your cutters. Cut it slowly and rotate the bracket as you cut.

Once it's removed as shown in Figure 9, take one square of foam mounting tape and cut it in half. The half piece should be the right size to mount on the long side of the plastic bracket (the side where you removed the plastic post). Leave the paper on the exposed side of the tape for now. Take the other half of the foam tape square and cut it in half again. Place this tape on the face that is perpendicular to the long side as shown in Figure 10. Again, leave the paper on the exposed side for now.

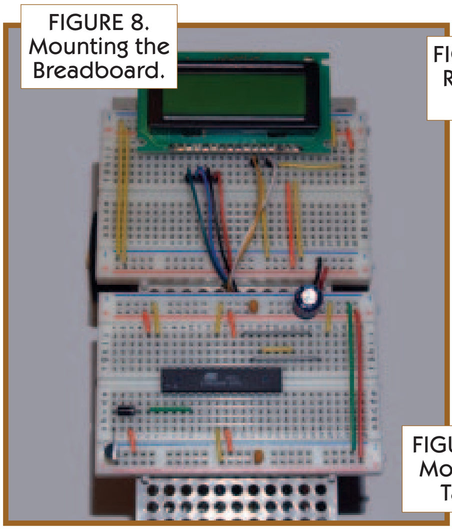


FIGURE 8. Mounting the Breadboard.

FIGURE 9. Remove Post.

FIGURE 10. Mounting Tape.

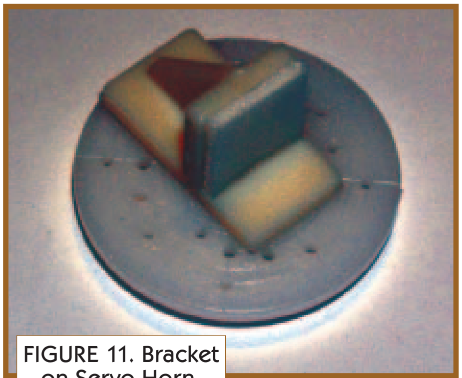


FIGURE 11. Bracket on Servo Horn.

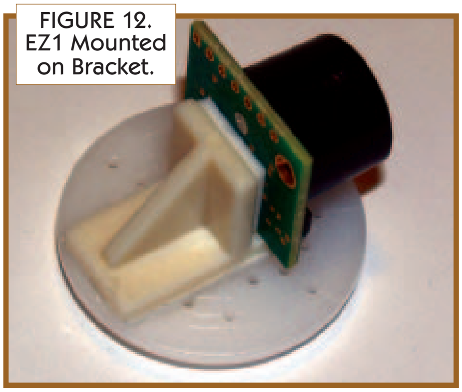


FIGURE 12. EZ1 Mounted on Bracket.

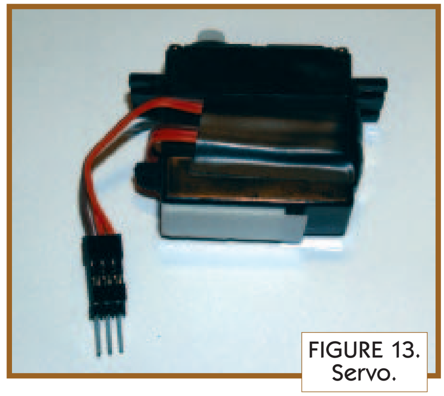


FIGURE 13. Servo.

Remove the tape cover from the long side of the bracket and place it on top of the large round servo horn. Center the bracket on the servo horn and position it so the angled edge of the bracket is even with the edge of the servo horn as shown in Figure 11.

Mount the EZ1 with the line of holes on top to the smaller piece of tape as shown in Figure 12. Make sure the holes on the EZ1 are not obstructed by the tape or the bracket.

Remove the small servo horn from the servo by removing the small screw in the center while holding the servo horn. *Do not hold the servo by its case while removing the screw as this may damage the gears inside the servo.* The servo connector is much longer than we need. Instead of cutting the connector off, leave it wrapped around the case and secure it in place with a piece of black tape.

Take another square of foam tape and cut approximately a quarter of it off so that the remaining piece is about the same width as the servo body. Place the tape on the bottom of the servo in line with the center of the servo horn as shown in Figure 13. Cut three pins from the male breadboard header (long pins on both sides of the header) and insert the pins into the servo connector.

Remove the paper and place the servo on the upper plate against the new breadboard as shown in Figure 14.

Plug the servo connector into the new breadboard placing the brown wire into B3, the red wire into B4, and the orange wire into B5.

Refer to Figure 15 as you complete the next step. Cut a piece of red, black, and white stranded wire about 7"-8" long. Strip about 1/4" of insulation off the end of the wires. Insert the red wire into the +5 hole on the MaxSonar-EZ1 from the front (the side with the black housing) and solder it on the back side (the side with the writing). Solder the black wire into the hole marked GND in the same manner. Solder the white wire into the hole marked AN. Run the three wires through the top mounting hole on the EZ1 to provide strain relief on the wires. Cut two pins and one pin from the male pin header. Solder the red and black wire to the two pins and solder the white wire to the single pin.

Place the servo horn with the EZ1 mounted on it on top of the servo. Position the horn so the EZ1 is facing to the front and push the horn onto the servo. The horn will fit snug enough to hold without the mounting screw (we may need to reposition it later to center the sensor). Plug the red and black wires into the Upper GND12 and Upper +V12. The red wire connects to +V12 and the black to GND12. Plug the white wire into J12. Your CIRCBot should look like Figure 16.

Conclusion

That completes the construction portion of this month's work. Please visit **www.wrighthobbies.net/guides** for the remainder of the instructions.

This is also the last article of the series. Your new CIRCBot is a valuable learning tool for robotics. We encourage you to experiment, make changes, and try out new ideas. Please visit our forums and share your experiences and perhaps you can pick up a few new ideas while you are there. SV

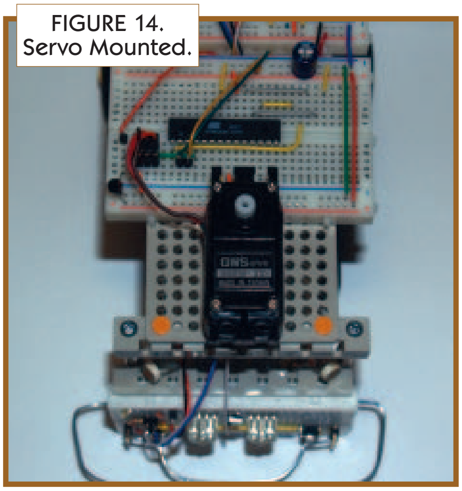


FIGURE 14. Servo Mounted.

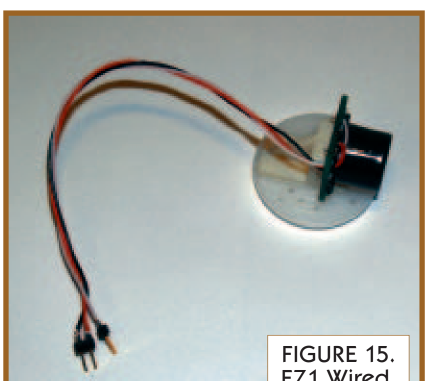


FIGURE 15. EZ1 Wired.

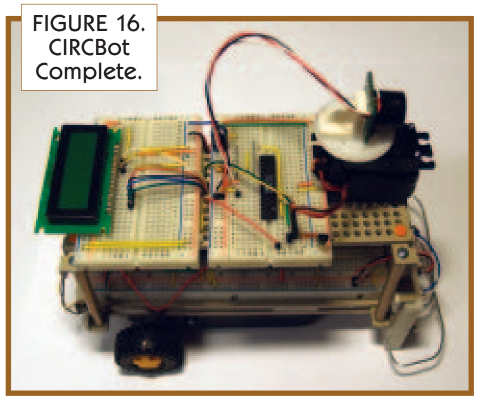


FIGURE 16. CIRCBot Complete.

Continued from page 35

University of Illinois at Urbana-Champaign, IL
Check the website for the details of this year's contest.
http://dc.cen.uiuc.edu

9-10 **National Robotics Challenge**
Veterans Memorial Coliseum, Marion, OH
In addition to Sumo and maze-solving events, this student competition includes two unusual ones: a robotic workcell event and a pick-and-place event.
www.nationalroboticschallenge.org

10 **CIRC Central Illinois Bot Brawl**
Peoria, IL
Includes several classes of autonomous Sumo and remote-control vehicle destruction.
www.circ.mtco.com

17-18 **Manitoba Robot Games**
Winnipeg, Manitoba, Canada
Events may include both Japanese and Western style Sumo, mini-tractor pull, and Atomic Hockey.
www.scmb.mb.ca

24 **Boonshoft Museum Robot Rumble**
Boonshoft Museum
Dayton, OH
The Robot Rumble is a Vex Challenge event following the FIRST rules.
www.boonshoftmuseum.org

31 **Penn State Abington Fire-Fighting Robot Contest**
Penn State Abington, Abington, PA
Regional for the Trinity Fire-Fighting contest.
www.ecsel.psu.edu/~avanzato/robots/contests/outdoor/contest05.htm

31 **Penn State Abington Mini Grand Challenge**
Penn State Abington, Abington, PA
Autonomous outdoor ground robots must navigate on and off-road around the campus, avoiding obstacles.
www.ecsel.psu.edu/~avanzato/robots/contests/outdoor/contest05.htm

April

1-8 **Africa Cup International Robotics Competition**
Pretoria, South Africa
Events include obstacle race, wall climbing, Sumo, and robot soccer.
www.nydt.org/home.asp?pid=713

CES 2007 in Review

by Ted Larson

As usual, in the second week of January, over 100,000 technology lovers converged on Las Vegas, NV, for the 40th annual Consumer Electronics Show (CES). The show goes on from 8 am to 5 pm for four days, and even then, it is almost impossible to see everything. CES takes up 1.8 million square feet of trade show space, spanning all of the major convention centers in Las Vegas. So much walking is involved for the attendees, there are areas with people giving leg, foot, and back massages for a mere $15+. There are lots of 108" plasma screens to see, and enormous, million dollar booths filled with the latest consumer gadgets, and technology-specific zones overflowing with emerging electronics, including — of course — robotics.

Robotics Trends has sponsored and run the Robotics Tech Zone at CES for the second year in a row. Robotics Trends also runs Robo Business, RoboNexus, and the new RoboDeveloper Conference to be held in Santa Clara, CA at the end of September 2007 (**www.roboticstrends.com**). With just a few exceptions, most of the attending robots and robot companies were on display in the Robotics Tech Zone, or in nearby booths.

Asimo (**world.honda.com/ASIMO/**) was there, although not where you would expect to find him. He was tucked away at a Honda booth in the mobile audio section of the Las Vegas Convention Center. He was a bus or monorail ride away from where all the other robots were, and if you didn't know he was there, you would miss him. I wasn't able to catch his show this year, although I heard from friends who made the trek over to seek him out, that they showed demos of him doing his usual stair climbing, waving to the adoring fans, and new for this year, jogging/running, which was quite a treat. As an exhibitor last year, I was able to get over before the exhibits opened and see them taking him through his power-up testing, which was quite interesting to watch. Because I was the only spectator, I easily got some up-close photos, video, and a nice Asimo lapel pin to commemorate my visit.

iRobot unveiled their newest product, the iRobot Create (**www.irobot.com/create**). I thought the iRobot Create was one of the more exciting new robots I saw at CES this year. This is a perfect platform for hardware hackers who have chopped into the iRobot Roomba floor cleaner. It is essentially a Roomba, with the vacuum cleaner component removed from the robot, leaving plenty of room for hardware of your own creation. At the $129.99 suggested price, this is right in the sweet-spot of most hobbyists and educational research robot builders. Plus, you get all the industrial design expertise of iRobot, and its rugged durability.

For an additional $59.99, you can pick up a Command Module sporting an Atmel ATMega168, with a complete software developer suite included. I spent some time talking to Helen Greiner, Co-Founder and Chairman of iRobot, and she is clearly excited about the potential of the iRobot Create to ignite great ideas in robotic researchers, hobbyists, and educators by giving them a rock solid platform with which to start.

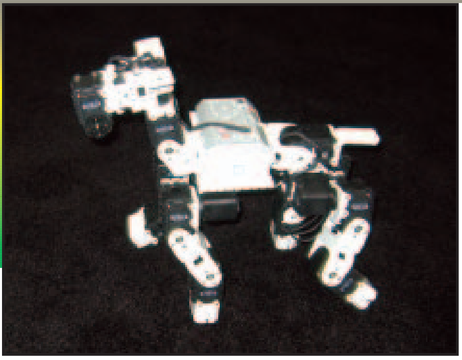
Just like all their other floor cleaning robots, the iRobot Create can interact with Virtual Wall units, as well as the charging station. Other than the Octobot, this is the first, inexpensive (sub $150), self-charging, capable hobby robot, I have seen on the market. It has a few modes of operation: passive, safe, or full. It can automatically detect "cliffs (dropoffs)" or wheel drops, and if the battery is on charge or not. You can either override it, or let the base be smart about things for you.

Other features are driving by position, velocity, and turn radius, controlling the LEDs on the robot, a music synthesizer that plays 16 note songs using MIDI commands, scripted command sequences, and full control over reading battery charging status. It has approximately 20 sensors on board for reading

Helen Greiner with iRobot Create.

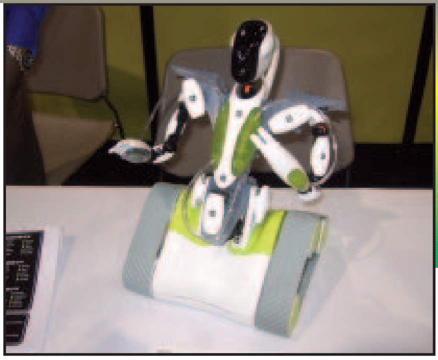
Hamster Roomba!

MP3 Player Bot.



Bioloid Puppy.

R2-D2 Mobile Entertainment System.



Spyke WiFi Spy Robot.

bumps, wheel drops, speed, the buttons on top of the robot, the IR remote control (optional), and battery status.

It is an impressive collection, for the price. They showed off a few Roomba hacker creations, including an impressive unit designed to allow a hamster ball to control where the Roomba would drive, and a beer fetching robot, with a robotic arm mounted to it. I have lots of good ideas of what I could do with one of these, probably enough for a whole other article.

JETRO, the Japan Economic Trade Organization (**www.jetro.com**), was there showing off one of their member companies robots — a wheel-shaped robot with an MP3 player in it — that drew quite a crowd.

Robotis displayed their latest Bioloid (**www.bioloid.com**) educational robot kit. The Robotis booth was quite active with several sample robots running around in front of their display. For you avid *SERVO* readers, you will remember that Bryce and Evan Woolley did quite a thorough review of the Bioloid kit in the January '07 issue.

I had not actually seen a Bioloid kit in-person until now. It is quite an impressive collection of parts. No wonder you can make so many different robots with only one kit. I was most impressed by — and I spent some time playing with — a robotic puppy made from the kit. The demo program they were running was really good. Of course, at the moment I tried to take a photo of the puppy, it suddenly lifted its leg, to simulate peeing. It seemed like the reality factor was in full force.

For those R2-D2 lovers out there, Nikko Home Electronics (**www.nikkor2d2.com**) showed up with an R2-D2 replica that contained a complete mobile home theater inside it: a projection TV, a DVD player, an iPod port, and a VGA port, in case you want to project your PC on the wall or ceiling. The remote control unit for the robot is shaped like a miniature Millennium Falcon, with a pop-out control pad. Their booth was adorned with a live Storm Trooper and Princess Leia to complete the Star Wars theme. For a mere $2,500, you can take one of these units home. They hope to have them in stores later this year.

Meccaro brought their new SPYKE (**www.spykeworld.com**) spy robot kit to show. It is being sold under the ERECTOR brand as a robot Erector set. It sports an impressive feature set such as a WiFi camera with WiFi robot control, IR based auto-parking and recharging, a digital music player, and VOIP phone and webcam capabilities.

Through their consortium of companies, the city of Osaka brought the PLEN (**www.plen.jp**) robot out for display. PLEN is made by Akazawa Co. LTD. PLEN is the cute little humanoid made famous by a YouTube (**www.youtube.com**) video where he is seen riding a skateboard and tossing balls around. PLEN has a Bluetooth interface, so you can control it from your PC or mobile phone. It comes with about 20 different motion programs, and it can run for about 25 minutes on a single charge. PLEN can be programmed over either a Bluetooth or USB connection. In the Osaka booth, there was also an impressive collection of humanoids from Kumotek (**www.kumotek.com**).

Elvis was alive and well in Las Vegas at the WowWee booth at CES (**www.wowwee.com**). Well, not all of him, just from the chest up. He sings and is similarly interactive like the Alive Chimpanzee product they released last year. For only $349, you can have Elvis in your home, ready to entertain at a moment's notice. WowWee had a total of five new product introductions, including WowWee Alive Elvis, Flytech Dragonfly, Roboboa, Robopanda, Roboquad, and RS Media. I

PLEN strikes a pose!

Kumotek Humanoids.

Elvis Lives!

Flytech Dragonfly.

Robopanda.

Flexo the Balancing Robot.

was most impressed with Elvis, the Dragonfly, and the Robopanda.

The Flytech Dragonfly, a remote controlled bug, that reminded me of the wind-up ornithopter toys of old, was regularly demonstrated. It is quite lightweight, with a crash resistant airframe. WowWee representatives were flying it around the convention hall above the booth, and it seemed quite easy to control. On a couple of occasions, I saw it spin into the crowd and bounce off the head of an unsuspecting — yet surprised — CES attendee with no injury whatsoever.

WowWee also had an interactive panda bear, dubbed, Robopanda, which tells stories and interacts with the kids, through loads of capacitive sensors all over its body. It has a plush toy baby bear, with an RFID tag embedded in it, so when you bring it near, it knows to cuddle it and interact with it. I can't wait to see the first autopsy of this guy. It looked like it was just packed full of all kinds of high-tech electronics.

Of course, I can't complete this article without tooting our own horn, and talking about OLogic (**www.ologicinc.com**). OLogic is an outsourced research and development company with a focus on robotics, that I co-founded with Bob Allen. OLogic brought out a few robots used to demonstrate their technology and design services: Flexo, the balancing robot, that chases the color red, Follow-Me Robot, which is a service robot designed to follow you around and carry your stuff for you, IGOR (Indoor GPS OLogic Robot), which demonstrated technology that allows a robot to know where it is exactly in a room using sonar technology, and finally, a robot affectionately called "The Brain."

With 2,700 exhibitors, CES booths are well known for lots of glitz and glamour, and smiling booth models ready to demonstrate products. One of our neighboring booths had a former Oakland Raiders cheerleader, demonstrating their products. I say, who needs a booth model when you have a few robots! All you have to do is switch them on, and you immediately have a crowd around your booth, asking what you do, and how to get more information on your products or services. Flexo and Follow-Me robot were our booth models this year.

The two star attractions in the OLogic booth were IGOR, and "The Brain." IGOR is an autonomous robot that was set up in a warehouse in Monterey, CA, with a live video feed coming to the OLogic booth over the Internet. IGOR uses an ultrasonic positioning system developed by SJ Automation, to determine its position in the warehouse down to 1/10th of a foot. IGOR has some on-board AI that makes him smart enough to navigate to one of three places in the warehouse just by being told where to go.

Another OLogic partner, ComCam International (**www.comcam.net**), provided live video feed and telepresence technology that made it possible to remotely send a command to IGOR from the booth at CES and watch his progress from both an overhead camera, as well as a camera mounted in IGOR's head. Overall, it made for an excellent demonstration of location awareness and remote control of an autonomous robot.

NeuroSky (**www.neurosky.com**) has created the world's first neural-interface for the consumer market, which they demonstrated in the OLogic booth. They have brainwave sensing and interpretation technology, that allows them to read brainwaves, and control something by translating them into a measure of attention, or meditation. Their demonstration included a video game, where you can toss objects around just by focusing on them, or levitate objects by meditating and becoming more calm. It may sound like something out of a Star Wars episode, but NeuroSky's technology is real, and it is here now. "The Brain" robot is a robot OLogic constructed using NeuroSky technology to control it. You can drive the robot around by focusing your attention on making it drive. This is super cool technology that always drew a huge crowd when someone was controlling the robot or the video game. Hopefully, there will be more on this in a future article.

Compared to last year, there were less exhibitors in the 2007 Robotics Tech Zone. However, comparatively speaking, there were definitely more attendees this year interested in robotics and how they apply to consumer electronics. Toward the end of the show, David Calkins, President of the San Francisco Robotics Society of America, RoboGames (**www.robogames.net**) organizer, and frequent *SERVO* contributor came by, and I talked to him about his impressions of CES this year. He took a few pictures of the Robotics Tech Zone and commented, "Just you wait. In 10 or 20 years, this whole hall will be filled with robots and robot companies, and you will be able to say you were there in the early days, when there were less than a dozen." For me, I guess that just about summed it up. SV

About the Author

Ted Larson is CEO of OLogic, Inc., and an active member in the Home Brew Robotics Club of Silicon Valley.

Robotics Showcase

by Dave Calkins

ROBOGAMES PREP:

Balancing Robots: Who Needs Dean?

What goes up, must come down. Gravity always wins, right?

The Segway started the fad. Why use three wheels when you can use two? It's not the most intuitive way to move, but it certainly is cool.

Since the Segway, there have been many robots that have used the two-wheel solution. These balancers are incredibly fun to watch, incredibly hard to make, and — you won't believe it — incredibly inexpensive. In talking with five builders of balancers, all of them built their robots for under $1,000, and several of them did it for around $100.

What's the point? Well, it's not very efficient, but it really forces you to understand the forces acting on a robot at any given time — especially gravity. If you've never built a robot before, building a balancer is probably not the best way to start. But if you have built a few robots, you'll be surprised at how many different ways there are to build a balancer. It's been done with as little as LEGOs, as simple as servos, or as complex as custom motherboards and industrial motors.

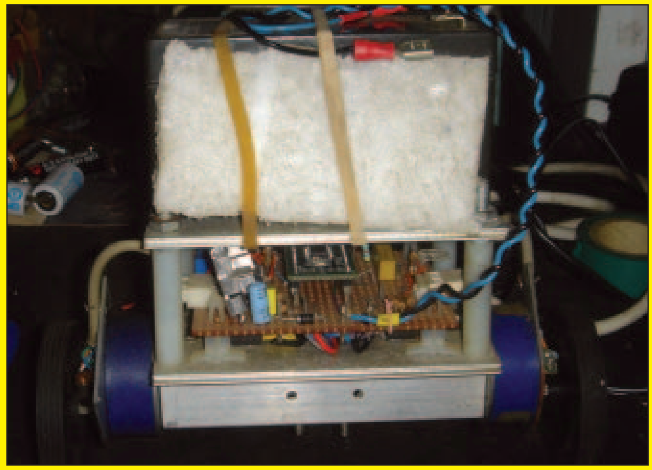
Robin Hartley of New Zealand built a balancer for only $100 — the hardest thing was "getting the accelerometer/gyro filter working correctly," which is what most builders say is the tough part. Building any robot always takes patience, but there are few projects that will test your resolve more than a balancer. When Bob Allen and Ted Larson did their first project, it took several months. But they now have several balancers to their credit, and have won first place in the balancer race at RoboGames the last two years.

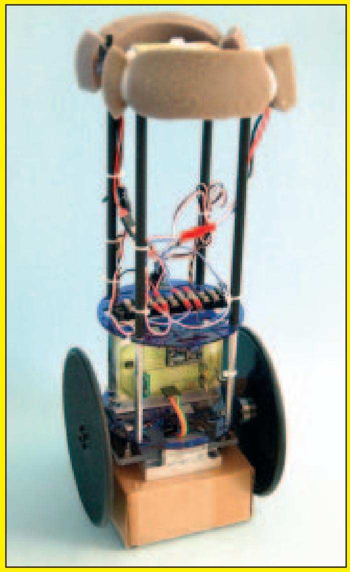
Trevor Blackwell was so enamored of the Segway, he actually built his own. That's certainly taking the balancer concept to the extreme, but hey — it will save you the $3,000 it takes to buy a new Segway. "Although the Segway has several exotic components, mine is built from common low-tech parts like wheelchair motors and RC car batteries. The parts, even at small quantity retail prices, cost less than half of a genuine Segway. It also doesn't need complex or high-performance software. The first version was written in Python and used serial ports to talk to the gyroscope and motor controller. The current software, now in C running in an onboard eight-bit microcontroller, is only 500 lines of code."

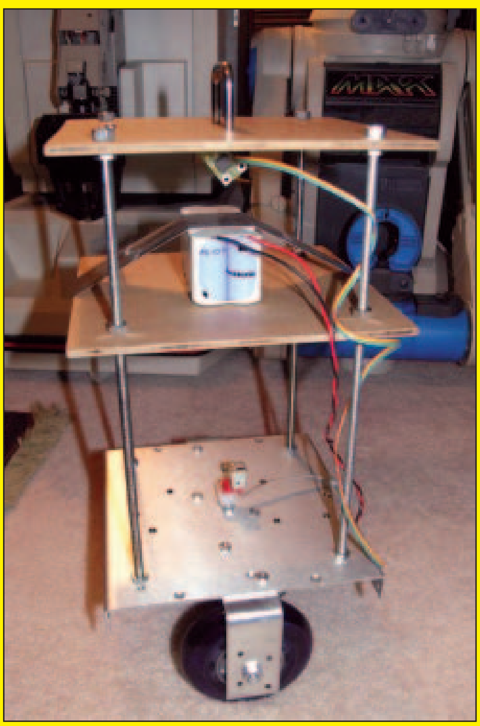
"You don't need high-tech low-inertia motors for adequate responsiveness. Regular old copper-wound motors work pretty well even though they have a lot of rotating mass that acts like a flywheel. This might actually help with handling a bump, as the inertia helps keep the wheels spinning up the incline. You don't need low-backlash gearboxes, either. The conventional non-precision spur gear units give about 1/8" backlash at the wheel diameter. You can feel a tiny clunk sometimes when the torque reverses, but it's hardly noticeable. They do make some gear whirring noise which is noticeable indoors."

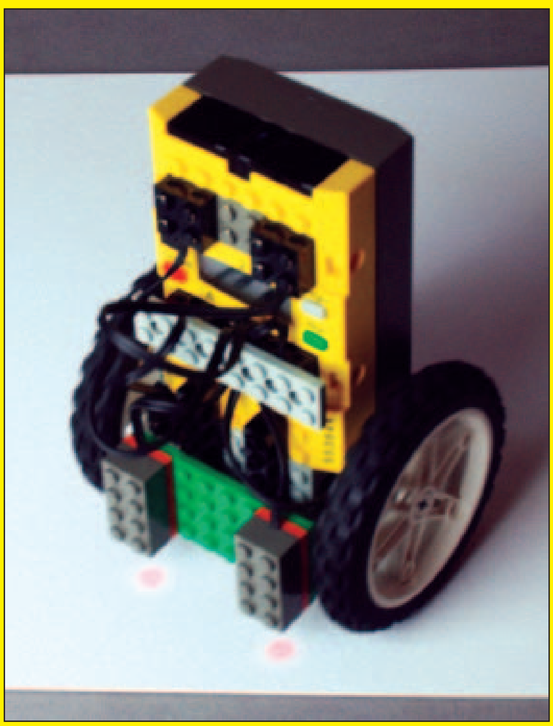
At the upper end of the cost spectrum, Steve Judd spent around $500 on his balanacer "Isorropos." He used a BasicX-24P, a Sparkfun IDG300 gyro, Dimension Engineering's DE-ACC3MD accelerometer, and some Hitec HSC5995-TG servos to get his bot running. The hardest thing about building the balancer was "integrating the hardware and software into a working system. The hardware by itself was trivial but the software, while simple in concept, took many revisions to get right. In the end, I had to revise the hardware and software together to get everything working. Things I discovered in testing the software triggered hardware changes, which then required more software revisions. This is an iterative process and you just have to be willing to take the time to work through all the issues. Start with a simple balancer that uses distance

"The Segway started the fad. Why use three wheels when you can use two? It's not the most intuitive way to move, but it certainly is cool."









sensors to determine tilt. After that works you can move on to the world of accelerometers and gyroscopes."

Steve adds "get the gyro/accelerometer integration working on a test jig before trying to integrate it into the robot." When picking parts, "try to estimate the maximum speed your drive train will be required to move, and double it. Also consider the power it will take to accelerate the robot when shoved. Then size the motors, drivers, and batteries appropriately. I don't want to make light of the importance of the tilt and rotation sensors, which are also quite important, but if the robot doesn't have the necessary authority (as in torque and speed) to keep itself upright, it doesn't matter how good the sensors are."

For only about $150, Gary Dion got cabin fever and built Zeke, the Balancing Snowman. There's no reason that your balancer shouldn't have personality! "The snowbot idea came simply from my desire to have the robot at our open house for the kids to see (and hopefully spark their interest in robots), and the imagination of my co-workers. We save piles of white Styrofoam from the boxes our computers arrive in, and it made a lot more sense to glue and carve the great construction material than to throw it away. Some of the best stuff in life is free."

"The hardest part of building a balancer is that since it is an inverted pendulum, you're always fighting gravity. And in order to correct for the robot's tendency to fall if left unpowered, you need to know two things: it's absolute angle (or tilt) and the rate of change of that angle (how fast it's tilting). There are many sources of books and online with information on solving the inverted pendulum problem. The challenge I found in building my robot was closing this control loop with the hardware I had on hand. The equations I started with all dealt in values such as angles and torques, while my microprocessor only knew of analog values from the gyroscope and accelerometer and a PWM signal to command the drive wheels. I spent most of my time tuning these relationships between the hardware values and the information they represented in the real world (i.e., determining what PWM value produces a certain torque at the wheels)."

If you don't have any money, or aren't very experienced at programming, you can do what Steve Hassenplug did — use LEGOs! If you've got a Mindstorms kit and a pocket flashlight, you've got everything you need to make a working balancer. Steve shares his code with everyone — if you want to build a robot in a day, just go to **teamhassenplug.com** and download the code that Steve wrote to make your own "legway."

"Once I was able to make it balance, it was a simple task to make it move forward and follow a line. Many people have asked if it will work on a slope. Actually, it will, for the same reason it will work on different color surfaces. LegWay constantly attempts to adjust its balance point. I've been able to put it on a table, and tilt the table, and LegWay continues to maintain its balance. To move forward (for line following), LegWay actually sets the motors to run backward, causing a tilt, which it automatically corrects, by moving forward. When one sensor is over the line, it stops that motor, and LegWay balances using only the other motor, causing it to turn."

LEGOs, servos, stepper motors, or wheelchair motors — like any other robot project, you've got an incredible range of options. But more than any other project, all of the roboteers who've made a balancer think it's one of the coolest things they've done. Once your robot stands up on its own, it's a lot like seeing your baby take its first steps — there's an incredible sense of pride. So what are you waiting for? Whether it's for RoboGames, a local robot event, or just the pleasure of defying gravity (which should be a strong motivator itself), building a balancer is a project within everyone's abilities and budgets. SV

BALANCER RULES

A balancing robot is a two or one wheeled robot that can maintain its balance without any external support (yes, you can have just one wheel!).

No robot may weigh more than 50 pounds, nor may it use an internal or external combustion engine. The robot must fit inside a 3' x 3' x 3' cube for the entire duration of its run.

There are two classes of balancing robot race: Autonomous and Remote Controlled.

- Autonomous — Once the robot starts the course, it cannot be touched by an operator, or be remote controlled, or it will be disqualified.

- Remote Controlled — Balancing must be autonomous. Driving can be controlled by an R/C unit.

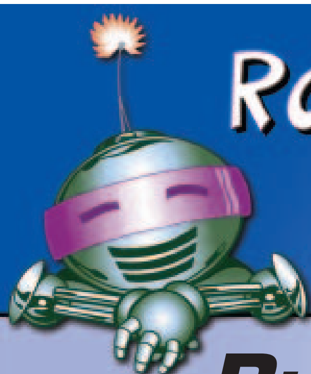
The robot starts behind a white starting line with the finish lines approximately 20 feet from the starting line. There are two white finish lines, placed 24" apart. Clock stops when robot remains stationary with its wheels or legs between the two lines for more than 10 seconds.

Obstacles may be placed on the course, at the discretion of the robot owner. Each obstacle traversed will apply a scoring multiplier to the robots time to complete the course.

Course boundaries will be determined prior to race. Robots that stray outside of the course boundaries on a given run will have twice the time outside the boundary added to their score.

Each robot is given three tries to complete the course. The best score from each of these tries will be used as the final score.

Each robot will have a maximum of 10 minutes to complete the course.



ROBOTICS RESOURCES

Tune in each month for a heads-up on where to get all of your "robotics resources" for the best prices!

BY GORDON MCCOMB

Bushings, Bearings, and Other Robotic Baubles

Bushings, bearings, sprockets, belts, chain, gears — they're all members of a family known as power transmission. In a nutshell, power transmission is anything that transfers power or movement from one device to another. In you car, the engine turns, which drives a transmission. That transmission may be connected to a drive shaft or a transfer case, which transmits power to the wheels. Along the way are various gears, bearings, bushings, collars, retailing clips, and other components commonly used to keep moving mechanisms, well, moving!

In this month's Robotics Resources, we'll look at sources that offer bushes, bearings, and other power transmission components. This tends to be the stuff of high-end budgets, so we'll mix in resources for industrial-strength and consumer-grade.

It's not uncommon for sellers of power transmission components to provide you with lots of technical details, but no prices! This is typical in the industrial supply business, where finding just the right part is more critical than pricing. As we amateur robot builders are typically long on dreams and short on cash, you'll want to specifically ask for a pricing sheet so you can compare costs.

A Quick Overview of Common Power Transmission Components

We could fill a book on everything power transmission and, in fact, if you are truly interested in the subject, your public library will have a number of industrial engineering books which you can review or borrow. And, sellers of power transmission components routinely publish design guides that provide excellent how-to information. Much of it is free, and available on the Internet. Be sure to check out the Sources listed later, and browse the documentation available on the company websites.

Gears, Bearings, and Couplers

• *Gears* are the mainstay of power transmissions, and are primarily used in robotics to reduce the speed and increase the torque of the wheel drive motors. Because of the mechanical precision required to properly mesh gears, most amateur robot builders do not construct their own gear assemblies, unless they are simple ones.

• *Bearings* are used to reduce the friction of a spinning component, such as a wheel or idler, around a shaft. Several bearing constructions exist, with ball bearings being the most common. Bearings can be mounted directly to a device, which requires precision machining and a press to securely insert the bearing into place.

• *Bushings* serve the same general purpose as a bearing, except a bushing has no moving parts. (Note: Some people also call these bearings or dry bearings, but I prefer to use the term bushing in order to differentiate them from components with moving parts.) The bushing is made of metal or plastic, and is engineered to be "self-lubricating."

• *Couplers* are used to directly connect two shafts together, thus obviating the need for any kind of gear or belt. Couplers come in two general styles: rigid and flexible.

Flexible Linkages with Belts and Chain

• Also called synchronization belts, most *timing belts* for small mechanisms range from 1/8 to 5/18 inch in width, and sizes from just a few inches in diameter, to several feet in diameter. Material is usually neoprene, with metal or fiberglass reinforcement. Belts are rated by the pitch between "cogs" or teeth, which are located on the inside of the belt.

• Timing belts are used with matching timing belt pulleys or *sprockets*, which come with either ball bearing shafts (used for idler wheels) or with press-on or set screw shafts for attaching to motors and other devices.

• *V-belts* have a taper V shape, and are used to transfer motion and power from a motor to an output, when synchronization of that motion is not critical (because the belt could slip). V-belts are often made with metal or fiberglass reinforced rubber, and are very sturdy.

• V-belts are used with V-grooved *pulleys*. By changing the diameter of the pulley, it's possible to alter the speed and torque of the output shaft in relation to the drive shaft.

• *Endless round belts* are used to

transfer low-torque motion. The belt looks like an overgrown O-ring and, in fact, is often manufactured in the same manner. Other endless round belts are made by fusing the ends of rounded rubber (usually neoprene). Some belt makers provide splicing kits so you can make custom belts of any length.

• *Ladder chain* resembles the links of a ladder, and is used for fairly low-torque and slow speed operations. Movement of a robotic arm or shoulder is a good application for ladder chain. With most chain, links can be removed and added using a pair of pliers. Special toothed sprockets engineered to match the pitch (distance from link to link) of the chain are used.

• *Roller chain* is exactly the same kind with bicycles. For smaller-scale machinery such as most robots, the chain isn't as big. Roller chain is available in miniature sizes, down to 0.1227 inch pitch (distance between link-to-link). More common is the #25 roller chain, which has a 0.250 inch pitch. For reference, most bicycle chain is #50, or 0.50 inch pitch. Sprockets with matching pitches are used on the driver and driven components.

Roller chain comes in metal or plastic; plastic chain is easier to work with, and links can be added or removed. Many types of metal chains are pre-fabricated using hydraulic presses, and require the use of "master links" to make a loop.

Supplementary Components

• *Idlers* (also called idler pulleys or idler wheels) take up slack in belt- and chain-driven mechanisms. The idler is placed along the length of the belt or chain, and is positioned so that any slack is pulled away from the belt or chain loop. Not only does this allow more latitude in design, it also quiets the mechanism.

• A flexible *cable*, made of plastic or metal, transfers power/movement by spinning within some protective sheath. The speedometer cable on older model cars is a good example.

• *Lead screws, acme screws, ballscrews*, and similar components convert rotational motion to linear motion. They are commonly used on CNC equipment, for example, to convert the rotation of a servo or stepper motor to the linear motion of a drill bit in the X and Y planes.

• *Motion control* products such as transfer tables and slides are manufactured subsystems that incorporate a number of power transmission components. Rather than build your own X/Y table, for example, you might purchase one as a subsystem, and use it in your design. Several companies that specialize in motion control products are included in the Sources section.

Shopping for Power Transmission Components

There are several online sources well-suited to the amateur robot builder. These include Stock Drive Products, Small Parts, Reid Tool and Supply, and MSC Industrial. What helps these companies stand out is the online ordering feature, where you can look for the component you want, add it to your shopping cart, and pay online. Many of the resources listed here are aimed at the industrial user, where it is common to order from a printed or web catalog that has no prices. You contact the company, get a quote, and arrange payment. For small $20-$50 orders, you'll find the online shopping cart experience much faster and economical.

Be prepared for some sticker shock when browsing products from different companies. A gear is not a gear is not a gear. Prices for what looks like the same type of gear can vary several thousand percent — from a buck for a surplus plastic gear, to $50 for a machined stainless steel version.

As you're looking, be sure to match the component to the job. If you're building a heavyweight combat robot, you'll probably need sturdy steel or machined aluminum gears and other power transmission components. You can expect the prices to be hefty, depending on the size and quality of the products.

If you're building a smaller carpet rover, it can probably use plastic gears, most (but not all) of which are injection molded. This makes them less expensive as they can be mass produced. You can then decide if you want the gear to be press-fit, if it should have a metal insert, and other options. Costs vary depending on the extras.

Be sure to check out surplus sources for power components. We've listed these several times in the recent past; be sure to check back issues of *SERVO*. While you may not always get exactly what you want when buying surplus, the cost savings can be enormous. It is often possible to tailor the design to fit the components available to you, rather than the other way around.

Sources

Following are online sources for bushings, bearings, gears, belts, and other power transmission components. As always, be sure to also check the inventory of our regular robotics specialty advertisers. While power transmission may not be their main business, it's not unusual for them to carry a small line of popular-sized gears and other products.

Anaheim Automation
www.anaheimautomation.com

Industrial motor control systems (motors, controllers, tachs); X-Y tables.

Applied Industrial Technologies
www.appliedindustrial.com

Industrial bearings, linear slides, gears, pulleys, pneumatics, hydraulics, and other mechanical things. Also hosts Maintenance America, online reseller of industrial maintenance supplies and general industrial supplies (wheels, casters, fasteners, and a lot more), tools, paints, and adhesives.

Applied Motion Products
www.appliedmotionproducts.com

Applied Motion is a major manufacturer of industrial motion control products, including stepper and servo motors, gear heads, and motor control electronics. Products are available through distributors, though they are also quite common on the surplus market. For this reason, you'll want to visit the website for its technical reference materials.

Bearing Belt Chain
www.bearing.com
Local and online retailer of bearings (linear, roller, taper, pillow, etc.), belts (including V and timing), sprockets, and chains. Large inventory.

Bearing Headquarters Co.
www.bearingheadquarters.com
Industrial bearings (all types), couplings, clutches, belt drives and rollers, gears, conveyor rolls and chain, sprockets, and chain.

Belt Corporation of America
www.beltcorp.com
Belts, and not the kind you wear. BCA offers timing belts, woven endless belts (can be useful to construct robot tank treads), natural rubber and neoprene stretch belts, and endless round belts.

Boca Bearing
www.bocabearings.com
Boca specializes in small and miniature bearings for such applications as radio control vehicles, inline skates, power tools, small appliances, fishing reels, and of course, robotics. Bearings are listed by size, type, and general application. Be sure to check out their engineering section, with some one dozen helpful technical backgrounders on using bearings.

Boston Gear
www.bostgear.com
Gears, yes, but also bearings, transmissions, clutches, pneumatics, and assorted other power transmission and actuation products. The company also offers free literature, maintenance manuals, and operating instructions for their products.

BRECOflex Co., L.L.C.
www.brecoflex.com
Manufacturer of belts: timing belts, profiled belts, flat belts, pulleys, belt tensioners, and slider beds.

Danaher Motion
www.ballscrews.com
High-end, high-quality ballscrews, threaded rod, linear stages, and other motion mechanicals. The site also provides a number of useful technical articles on designing with and using motion mechanicals.

Drives, Incorporated
www.drivesinc.com
Drives, Inc., makes and sells roller chain and "attachment products," as well as chain for conveyors. The chain is available in sizes from #35 (slightly smaller than bicycle chain) on up to A2060, which has a pitch of 1-1/2 inches. So-called attachments include mechanical clips that seat into the chain — ideal for making heavy-duty tracked robots.

Dura-Belt, Inc.
www.durabelt.com
Makers and sellers of round urethane endless belts (O-rings), quick-disconnected twisted belts, flat belts (in different thicknesses and widths), groove sleeves for round belts, idlers, and belt splicing kits.

Dyna-Veyor
www.dyna-veyor.com
Maker of plastic conveyor belt chain, sprockets, idlers, and related conveyor components. Intended mainly for the food industry, the components can also be used in the design of tracked robots.

Emerson Power Transmission Manufacturing
www.emerson-ept.com
Mondo major manufacturer of power transmission and motion products.

Electronics Parts Center
www.electronicsic.com
Specializes in replacement/service parts for electronics products (power supplies, monitors, TVs, you name it). Look up parts by part number or function. Includes mechanical VCR parts, such as rollers, gears, and belts. This is one way to get mechanical components for cheap, though your engineering selection is somewhat limited.

Gates Rubber Co.
www.gates.com
Gates is a major supplier of belts for timing and power transmission, for both industry and automotive applications. Among the most useful belts (for robotics) in their line: synchronous ("timing") belts — they keep parts of a mechanism working together; Vectra and V-Belts — belts with trapezoidal shape that work in V-shaped pulleys

Go Kart Supply
www.gokartsupply.com
Parts for go karts and mini bikes (and therefore for the plus-size robots out there), including bearings, drive sprockets and chain, axles, wheels, replacement tires, control cables, and clutches.

Golf Car Catalog
www.golfcarcatalog.com
All replacement parts for golf cars, including motors and batteries.

Helical Products Co., Inc.
www.heli-cal.com
Helical flexible couplings. Many different sizes, styles, and materials.

Huco Engineering Industries Ltd.
www.huco.com
Manufacturer of flexible couplers. Products include three-part couplers with replaceable wear elements, one-piece couplers, and plastic universal joints.

igus GMBH
www.igus.de
Makers of polymer (plastic) bearings, chain, linear slides, and other mechanicals.

JJC & Associates
www.jjcassociates.com
Custom and standard drive components. Belts, timing belts, pulleys, gears, plastic power drive components, collars and clamps, rollers.

Karting Distributors, Inc.
www.kartingdistributors.com
Though intended for go karts, the company's bearings, axles, sprockets, chain, and other mechanical components are useful on larger robots, especially those intended for mortal combat.

Kerk Motion Products, Inc.
www.kerkmotion.com

Linear mechanicals; leadscrews, acme nuts, linear rails, spline rails.

Lovejoy, Inc.
www.lovejoy-inc.com

Lovejoy manufactures a line of affordable flexible couplers. These are designed to connect a motor drive with some driven device, like a pump or a wheel. Because they are flexible, the coupler allows the shafts of the driver and the drive to be slightly out of whack from one another, and yet they won't tear each other apart.

Manufacturer's Supply, Inc.
www.mfgsupply.com

Chain saw, motorcycle, and engine parts. Includes wheels, chain, bearings, axles, snowmobile treads, and a lot more. Check out the Go-Kart page at **www.GoKartParts.com**.

Maryland Metrics
www.mdmetric.com

Something of a one-stop-shop, Maryland Metrics carries bearings, linear bearings, fasteners, rods, gears, pneumatic and hydraulic fittings, and a variety of power transmission items. Good assortment of technical info.

MSC Industrial Direct Co., Inc.
www.mscdirect.com

Online catalog retailer of industrial products, including a large assortment of power transmission components, large to small.

Minarik Corporation
www.minarikcorp.com

Full line of mechanical (bearings, shafts, gears, chain, etc.); electronics (PWM drives, sensors); online ordering, plus many local warehouses throughout the US.

Nook Industries
www.nookindustries.com

Nook specializes in linear mechanicals (lead screws, acme screws), linear bearings, worm gear actuators, and ballscrew products.

Nordex, Inc.
www.nordex.com

Gears, miniature instrument bearings, shafts, Geneva mechanisms, fasteners, ball (linear and rotary) slides, brakes, clutches, couplings, assemblies, enclosed gear trains, and many other related precision components.

NSK
www.nsk.com

Power transmission — bearings, bushings, gears, sprockets, and more. Extensive technical details provided on the site, including online engineering calculators.

Parker Hannifin, Inc.
www.parker.com

A multi-faceted manufacturer of industrial automation components. Their main lines include linear actuators; pneumatic rotary drives; motors; structural framing; linear tables and slides; gearheads; gantry robots. Most products are available only through distributors, though some (like Compumotor division motors) can be purchased directly.

PIC Design
www.pic-design.com

Precision mechanical components, motion control mechanicals, X-Y translation tables, lead screws, belts, pulleys, and gear products.

Power Transmission.com
www.powertransmission.com

Power Transmission.com helps you find suppliers of gears, motors, bearings, clutches, couplings, speed reducers, and other components that transmit mechanical power.

Putnam Precision Molding, Inc.
www.putnamprecisionmolding.com

Manufactures and sells the Plastock line of mechanical drive components. Products include: timing belt pulleys, timing belts, chain sprockets, roller chain, and spur gears.

Reid Tool Supply Co.
www.reidtool.com

Reid is an all-purpose industrial supply resource. They carry tens of thousands of items, including bearings, gears, linear shafts, lead screws and nuts, ball screws and ball nuts, multi-directional rollers (omniwheels), ball transfers and ball casters, light- to heavy-duty casters, machine framing (reseller of 80/20, Inc.), fasteners of all kinds, and much more.

Rockwell Automation
www.rockwellautomation.com

Rockwell Automation manufactures a broad line of automation electronics and components: stepper motors, servo motors, encoders, linear motion products — about 500,000 items in all ... a little too many to list here.

Secs, Inc.
www.prosecs.com

Gears, sprockets, bearings, shafts, couplings — you get the idea.

Seitz Corp.
www.seitzcorp.com

Plastic gears, gears, motion control mechanicals.

Serv-o-Link
www.servolink.com

Serv-o-Link is the source for power transmissions with precision plastic gears, chain, and sprocket drives. The products are injection molded, so they're less expensive than machined gears in Delrin or metal, yet they are precise enough for many robotic applications. The snap-lock chain link design allows any pitch length; master links are not necessary, nor are special tools.

Servo Systems Co.
www.servosystems.com

Servo Systems Co. is a full service motion control distributor and robotic systems integrator. The website contains copious descriptions and technical data on their industrial components. The company also sells motion mechanicals, such as linear stages. Be sure to check out their "surplus bargains" pages for affordably-priced servos and other gear.

Small Parts, Inc.
www.smallparts.com

Small Parts is a robot-builder's

dream, selling most every conceivable power transmission part, from gears to sprockets, chain to belts, and bearings to bushings. Product is available in a variety of materials, including brass, steel, and aluminum, as well as nylon and Delrin. Rounding out the mix is a full selection of raw materials: metal rod, sheets, tubes, and assorted pieces, as well as a huge assortment of fasteners.

Stock Drive Products
www.sdp-si.com

SDP is a manufacturer and seller of power transmission products — gears, bearings, bushings, shafts, sprockets, chain, and dozens of other categories. They specialize in the smaller scale stuff that is most useful in amateur robotics.

Specialty Motions, Inc.
www.smi4motion.com

Specialty Motions makes and sells (online or through distributors) pro-grade linear motion mechanicals: leadscrews, acme screws, ballscrews, linear bearings, guides, rails, and more. The website contains plenty of tech docs on building and using linear motion gear, and the online e-store provides spec info and (usually) a picture or mechanical drawing.

TS Racing, Inc.
www.tsracing.com

Go kart and motorbike racing supplies, also suitable for use for robots (racing or non): bearings, axles, sprockets, sprocket hubs, wheels (five and six inch diameter), and assorted mechanical parts.

Vaughn Belting
www.vaughnbelting.com

Vaughn Belting is a local distributor of rubber, nylon, steel, and plastic timing belts, conveyor belts, and other belts used in industry.

W.M. Berg
www.wmberg.com

W.M. Berg, Inc., manufactures and distributes precision mechanical components, including gears, rotary bearings, pulleys. belts, hardware and fasteners, linear bearings and slides, couplings, flexible ladder chain (useful as miniature robot tracks), and roller chain (both plastic and metal).

Wholesale Bearing & Drive Supply
www.wbds.com

Online sales of bearings and other power transmission components. **SV**

ABOUT THE AUTHOR

*Gordon McComb is the author of the best-selling Robot Builder's Bonanza and Electronics for Dummies. In addition to writing books, he operates a small manufacturing company dedicated to low-cost amateur robotics (**www.budgetrobotics.com**). He can be reached at robots@robotoid.com.*

The *SERVO* Store

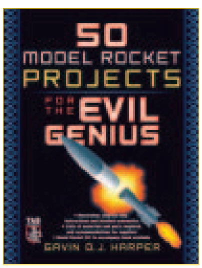
Mind Candy For Today's Roboticist

SERVO CD-ROM

Are you ready for some good news? Along with the first 26 issues of *SERVO Magazine*, all issues from the 2006 calendar year are now available, as well. These CDs include all of Volume 1, issues 11-12, Volume 2, issues 1-12, Volume 3, issues 1-12, and Volume 4, issues 1-12 for a total of 38 issues all together. These CD-ROMs are PC and Mac compatible. They require Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the discs. **$24.95 – Buy 2 or more at $19.95 each!**

50 Model Rocket Projects for the Evil Genius

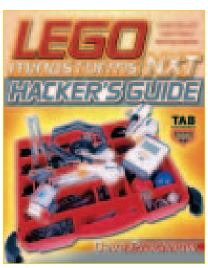
by Gavin D. J. Harper



Yes, as a matter of fact, it IS rocket science! And because this book is written for the popular Evil Genius format, it means you can learn about this fascinating and growing hobby while having run creating 50 great projects. You will find a detailed list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions. **$24.95**

LEGO MINDSTORMS NXT Hacker's Guide

by Dave Prochnow



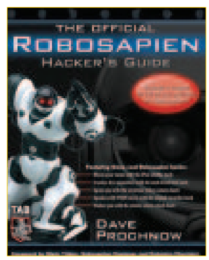
Here is an awesome next-generation collection of LEGO MINDSTORMS projects that enables you to build and program a real working robot in just 30 minutes! New technologies and expanded sensor capabilities make it easier than ever to add a level of sophistication to robotic and architectural creations. This cutting-edge guide describes new advances that make LEGO MINDSTORMS NXT such a great robotics resource. The book explains the all-new NXT intelligent brick ... the interactive servo motors with rotation sensors that align speed for precise control ... the ultrasonic sensor that allows robots to "see" by responding to movement ... the improved light and touch sensors that let robots detect color and feel ... and much more. **$24.95**

SERVO CD-Rom

Are you ready for some good news? Along with the first 14 issues of *SERVO Magazine*, all issues from the 2005 calendar year are now available, as well. These CDs include all of Volume 1, issues 11-12, Volume 2, issues 1-12, and Volume 3, issues 1-12, for a total of 26 issues all together. These CD-ROMs are PC and Mac compatible. They require Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the discs. **$24.95 – Buy 2 or more at $19.95 each!**

The Official Robosapien Hacker's Guide

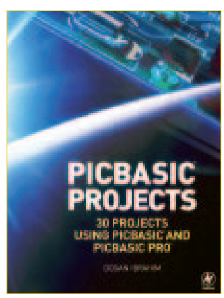
by Dave Prochnow



The Robosapien robot was one of the most popular hobbyist gifts of the 2004 holiday season, selling approximately 1.5 million units at major retail outlets. The brief manual accompanying the robot covered only basic movements and maneuvers — the robot's real power and potential remain undiscovered by most owners — until now! This timely book covers all the possible design additions, programming possibilities, and "hacks" not found anyplace else. **$24.95**

PIC Basic Projects

by Dogan Ibrahim



Covering the PIC BASIC and PIC BASIC PRO compilers, *PIC Basic Projects* provides an easy-to-use toolkit for developing applications with PIC BASIC. Numerous simple projects give clear and concrete examples of how PIC BASIC can be used to develop electronics applications, while larger and more advanced projects describe program operation in detail and give useful insights into developing more involved microcontroller applications. **$29.95**

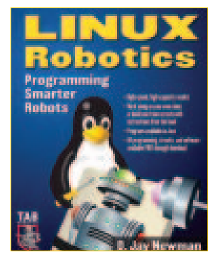
We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

SERVO CD-Rom

Are you ready for some good news? Starting with the first *SERVO Magazine* issue — November 2003 — all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 1, issues 11-12 and Volume 2, issues 1-12, for a total of 14 issues. The CD-Rom is PC and Mac compatible. It requires Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the disc. **$29.95**

Linux Robotics

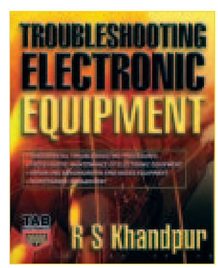
by D. Jay Newman



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need, plus complete training in doing original programming. **$34.95**

Troubleshooting Electronic Equipment

by R. S. Khandpur

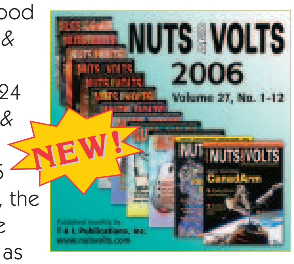


From cell phones to medical instruments to digital and microprocessor based equipment, this hands-on, heavily illustrated guide clearly explains how to troubleshoot, maintain, and repair all types of electrical equipment. The author covers all the essentials such as necessary tools, soldering techniques, testing, fundamental procedures, and mechanical and electrical components. **$49.95**

To order call 1-800-783-4624 or go to our website at www.servomagazine.com

Mind Candy For Today's Roboticist

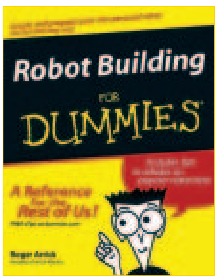
Nuts & Volts CD-ROM



Here's some good news for *Nuts & Volts* readers! Along with all 24 issues of *Nuts & Volts* from the 2004 and 2005 calendar years, the 2006 issues are now available, as well. These CDs include all of Volumes 25, 26, and 27, issues 1-12, for a total of 36 issues (12 on each CD). These CD-ROMs are PC and Mac compatible. They require Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the discs. **$24.95 – Buy 2 or more at $19.95 each!**

Robot Building for Dummies

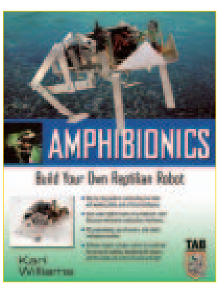
by Roger Arrick / Nancy Stevenson



Ready to enter the robot world? This book is your passport! It walks you through building your very own little metal assistant from a kit, dressing it up, giving it a brain, programming it to do things, even making it talk. Along the way, you'll gather some tidbits about robot history, enthusiasts' groups, and more. Do it the Dummies' way — explanations in plain English, "get in, get out" information, icons and other navigational aids, tear-out cheat sheet, top 10 lists, and more. **$21.00**

Amphibionics

by Karl Williams



If you're a robotics hobbyist with a flair for creativity, here's your opportunity to join the revolution and advance robotic evolution. *Amphibionics: Build Your Own Biologically Inspired Reptilian Robot* leads you step-by-step through four amazing projects which, when completed, will leave you as master of a gang of robot reptiles that jump, slither, walk, swim, and respond to their environment! **$19.95**

Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

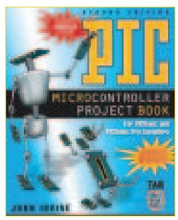
Insectronics

by Karl Williams

This complete project book delivers all the step-by-step plans you need to construct your own six-legged insect-like robot that walks and actually responds to its environment. Using inexpensive off-the-shelf parts, hobbyists can "build a better bug" and at the same time have loads of fun honing their knowledge of mechanical construction, programming, microcontroller use, and artificial intelligence. **$19.95**

PIC Microcontroller Project Book

by John Iovine



The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, in the four years that have passed since the book was first published, the electronics hobbyist market has become more sophisticated. Many users of the PIC are now comfortable shelling out the $250 for the price of the Professional version of the PIC Basic (the regular version sells for $100). This new edition is fully updated and revised to include detailed directions on using both versions of the microcontroller, with no-nonsense recommendations on which is better served in different situations. **$29.95**

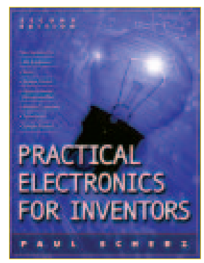
Take This Stuff and Hack It!

by Dave Prochnow

Transform common household items into really cool stuff. You don't need to be an electronics genius to get started turning everyday items into high-performing wonders. With how-to guru Dave Prochnow's step-by-step directions and fully illustrated plans, even beginners can hack their way to a high-tech home, cooler toys, and less yard work. Certain to fire your imagination and start you plotting new, original, and even more creative wonders you can make from ordinary household items, *Take This Stuff and Hack It!* is the perfect gift for your inner inventor. **$27.95**

Practical Electronics for Inventors

by Paul Scherz



This intuitive, applications-driven guide to electronics for hobbyists, engineers, and students doesn't overload readers with technical detail. Instead, it tells you — and shows you — what basic and advanced electronics parts and components do, and how they work. Chock-full of illustrations, *Practical Electronics for Inventors* offers over 750 hand-drawn images that provide clear, detailed instructions that can help turn theoretical ideas into real-life inventions and gadgets. **$39.95**

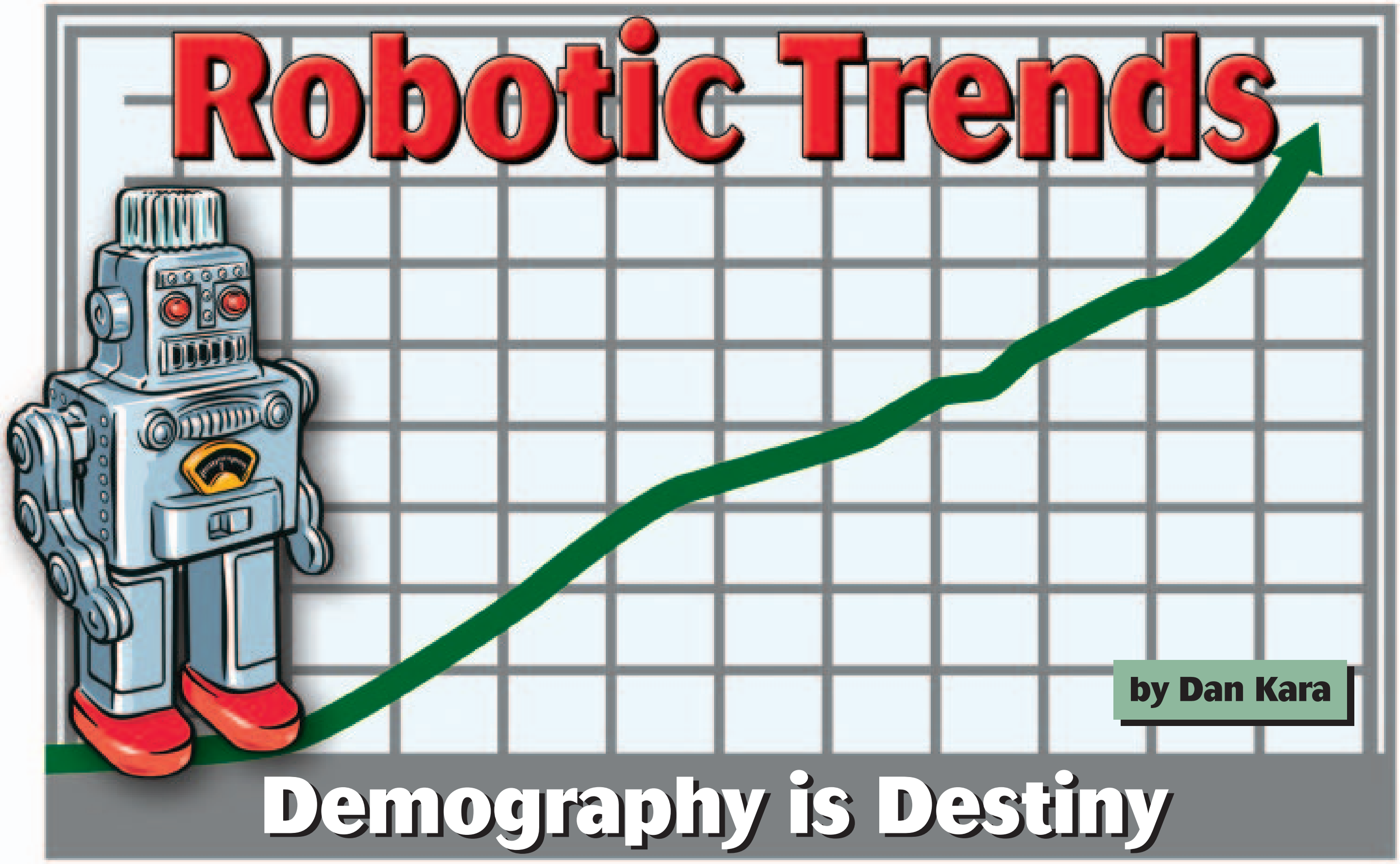
$20 each
15% off if you buy two or more.

From HomoSapien to RoboSapien

Before R2D2 there was R1D1

SERVO Magazine T-Shirts

Get your very own limited-edition *SERVO Magazine* T-shirt. Shirts come in sizes S, M, L, and XL, and are available in either black or white. All shirts are 100% pre shrunk cotton.



Robotic Trends

by Dan Kara

Demography is Destiny

When it comes to choosing a robotics career or looking for opportunities in the robotics market, your time would be well spent looking for areas where demand will be high and money will be available.

In past editions of *SERVO*, we have addressed the subject of robotics as a career, with an eye toward those areas in robotics that are expected to demonstrate rapid growth in the near future. I believe that one of the key factors for determining which robotics markets are expected to expand — and hence where to place your robotic career bets — is what I call the 'forced hand.' Educational robotics and military robotics — both hot robotics markets at this time — are the result of the forced hand. For example: The forced hand for military robotics is casualty reduction, with the added bonus of cost savings and enhanced functionality. Governmental mandates also help. Congress has asked the Defense Advanced Research Projects Agency (DARPA) that "one third of military land vehicles be unmanned by 2015." The 'forced hand' for educational robotics is the need for states and nations to increase science and engineering proficiency of their students (the next generation workforce). Our future might depend on it.

As you can see from the examples above, the forced hand usually involves governmental directives backed up by an infusion of taxpayer money. It is also usually the result of consumer 'pull.' That is, consumers — in this context, I mean the government, the state, or individuals — persuade vendors to produce products and services to meet their immediate needs. Educational robotics kits or military 'bots that perform needed tasks while keeping their human operators out of harms' way, provide examples of consumer pull at work. Vendor 'push' — the opposite of consumer pull — occurs when solution providers push products and services on to consumers that they have not called for. The introduction of personal computers in the 1970s serves as an example.

The forced hand also involves an element of certainty or inevitability. That certainty often includes guaranteed monies to be spent and a fixed timeline for completion. The Year 2000 date change problem is the best example of this. The computer problem — also known as the "millennium bug" — resulted when programmers, trying to save space, used two digits to represent the year instead of four. Thus, the year 2000 would be written as 00, and date-dependent systems could mistake this as 1900 when making calculations using the date. This would cause computer systems to malfunction, issue errors, or shut down altogether. The result was that millions of systems and billions of lines of COBOL code (primarily) had to be reviewed and reengineered before the stroke of midnight December 31, 1999 — no matter the cost, no matter the manpower allotted to it.

Of course, some things in life are more certain than others. The Year 2000 date changeover was a certainty,

> **The 'forced hand' usually involves governmental directives backed up by an infusion of taxpayer money."**

but there are even greater degrees of certainty. Consider, for example, Ben Franklin's belief that "In this world nothing can be said to be certain, except death and taxes." I can't speak for taxes, but death is as certain as it comes. People, viruses, galaxies, and possibly even the universe (or universes according to some theorists) are born, grow, and eventually die.

Personally, I cannot think of any direct relationship between robotics and taxes, but I cannot say the same for robotics and death, or at least death's opposite — life. Right now you might be asking yourself what does life and death have to do with careers and opportunities in robotics? Read on!

Another 'Boomer' Turns 50 Every Seven Seconds

The most glaring example of the forced hand as a driver for robotics market growth involves neither the military nor educational systems. After all, the world's best militaries have proven to be very proficient at what they do even without the benefit of robots and robotic technology. Educators, too, will continue to educate with or without robots. Certainty in these cases is less than 100%.

As noted above, there is a 100% certainty that everyone reading this column will die at some time, but what matters to us for this discussion is when. In 2004 in the US, the life expectancy at birth reached a record high of 77.9 years, up from 75.4 years in 1990 and this figure continues to increase.

More importantly, however, the proportion of the population older than age 65 in the United States, as well as in Japan and many other industrialized nations, is increasing at a faster rate than the total population. For example, in the United States from 1950 to 2005, the total population increased from 151 million to 296 million, representing an average annual growth rate of 1.2%. During the same period, the population 65 years of age and over grew on average 2.0% per year, increasing from 12 to 37 million persons (see Figure 1). The population 75 years of age and over grew the fastest (about 2.8% per year), increasing from four to 18 million persons. Projections indicate that the older age groups will continue to grow more rapidly than the total population.

By 2029, all of the baby boomers (those born in the post World War II period 1946–1964) will be age 65 years and over. As a result, the population age 65–74 years will increase to 10% of the total population by 2030, while the population 75 years and over will rise to 9% in the same timeframe and continue to grow to 12% in 2050 (see Figure 2). By 2040, the population age 75 years and over will exceed the population 65–74 years of age. In Japan, matters are even more extreme. It has been estimated that nearly 19% of the 130 million people that live in Japan are age 65 and over, and this figure is expected to rise to 40% by 2055.

Not Your Father's Retirement

Along with the certainty of aging, comes the equal certainty of eventual disability. This natural part of the human experience, however, does not diminish the right of people to live independently and pursue meaningful activities. Just ask the 'boomers — a generation built on high expectations. You can bet that boomers will not age in the same way as previous generations. Older adults will increasingly choose to continue to live at home rather than be in assisted living facilities or in nursing homes. They will demand

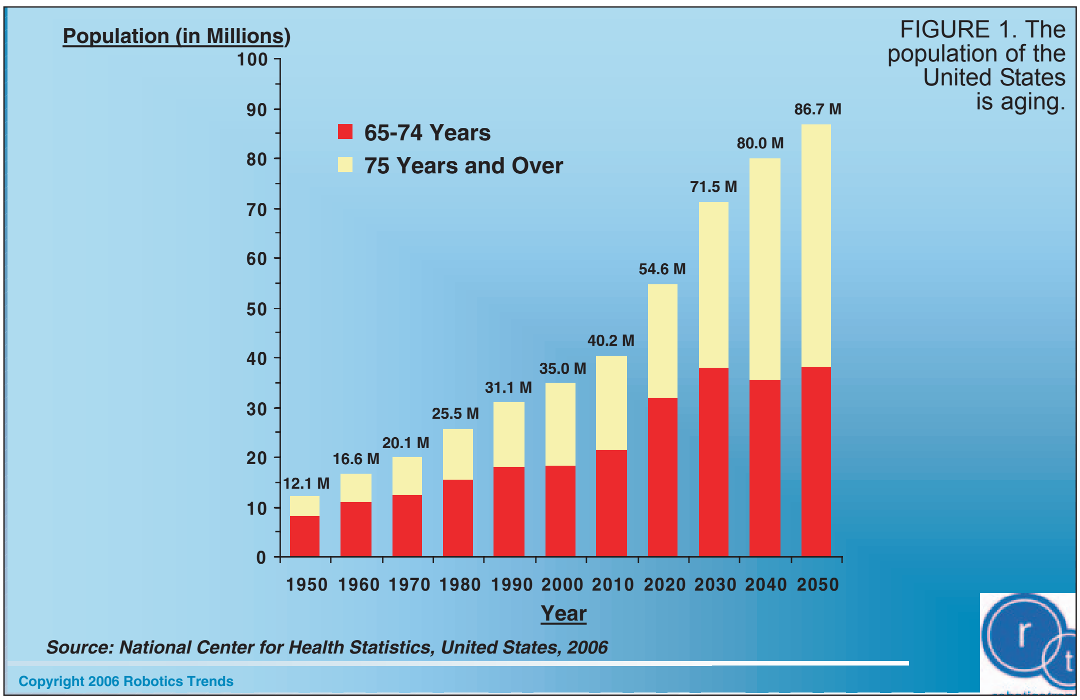


FIGURE 1. The population of the United States is aging.

that advances in technology be leveraged to overcome the disabilities associated with aging. That includes robotics technology that allows seniors to live independently, and exercise control over their life through the use of technologies that minimize their dependence upon their family and medical institutions.

Power and Money

According to the insurance industry's MetLife Mature Market Institute, when the last baby boomer turns 65 in 2029, the generation will control more than 40 percent of the nation's disposable income. In addition, the huge number of baby boomers will guarantee that their political voice will be heard. After all, even senators and congressmen get old. Beyond that, the government understands that it is cost-effective to support the independence of their aging population in as many of the aspects of their lives as possible, whether by supporting their continued employment, independent living, access to transportation, and so on.

So, what does all this mean? Well, it means you have a growing population that historically has been disinclined to compromise on lifestyle issues, unwilling to age gracefully, and backed by piles of disposable income and political clout. In terms of robots and robotic technology, this creates a demographic perfect storm for products and services that allow seniors to live fuller, more independent lives, for a longer period of time, compared to their counterparts in previous generations.

Assistive Technology

It would be in error to equate medical robotics technology, including surgical robots, hospital, and pharmacy automation products, designed for use in the hospital or senior center, with products that are designed for use in the home. We are not speaking of medical robotics technology, but about home-based assistive technology.

The term 'assistive technology' is defined as any device or technology that is used to increase, maintain, or improve functional capabilities of individuals. Assistive technology for use in the home can take many forms, some of which — such as vision and reading aids — do not involve robotics technology directly. Many other assistive technology products, however, are dependent on, or would greatly benefit from, the incorporation of intelligent robotics technology. Examples include:

- *Socially Assistive Robots* — These are robots or robotic technology that focus on social interaction, rather than the physical interaction between the robot and the human user. They can act as cognitive aids providing reminders and task instruction, or assisting in decision making, medication management, or scheduling. They can also act as reasoning systems with a friendly face that interact with the elder, monitor activity, detect abnormal situations, and respond to those situations (contacting caregivers).

- *Daily Living Aids* — Robotics technologies in this class involve physical interaction with individuals providing support for day-to-day daily living activities including: clothing and dressing, eating and cooking, home maintenance, and toileting and bathing.

- *Mobility and Transportation Aids* — These are robotics technologies that assist individuals to move about independently. Products include: scooters and power chairs, wheelchairs, and vehicle conversions.

- *Prosthetic and Orthotic Devices* — These are replacement, corrective, or supportive devices designed to replace a missing portion of the body, prevent or correct physical deformity or malfunction, or support a weak or deformed portion of the body.

Demography is Destiny

When it comes to choosing a robotics career or looking for opportunities in the robotics market, your time would be well spent looking for areas where demand will be high and money will be available. It is all the better if the 'forced hand' in terms of mandated spending (the government) comes into play, or if other social factors force the issue. Does assistive technology qualify? It is a certainty. **SV**

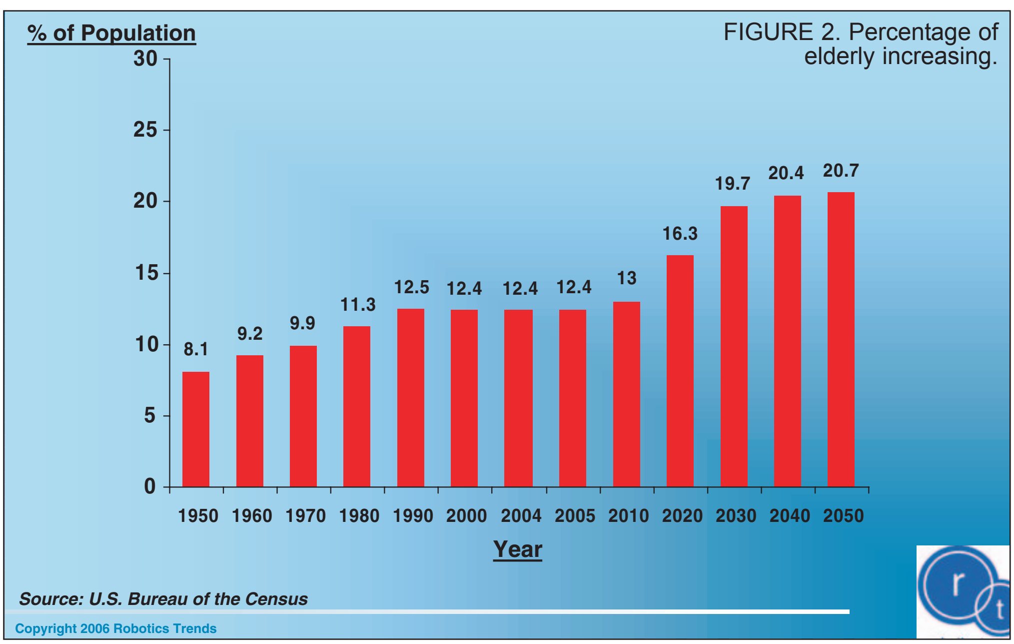


FIGURE 2. Percentage of elderly increasing.

Dan Kara is President of Robotics Trends, the producer of the RoboBusiness (**www.robobusiness2006.com**) and RoboNexus (**www.robonexus.com**) conferences, and publisher of Robotics Trends (**www.roboticstrends.com**), an online news, information, and analysis portal covering the personal, service, and mobile robotics market. He can be reached at dk@roboticstrends.com

Robots 2.0

by Helen Greiner, Co-founder and Chairman of iRobot Corp.

Everyone has great ideas about what they want a robot to do. But building a robot that can actually move, let alone complete the desired mission, is much harder than even the most dedicated hobbyists dreamed.

Not anymore. iRobot recently launched a robot that puts hobbyists on the fast-track to creativity. It's the iRobot Create Programmable Robot.

With Create, you can start programming your robot from day one. You can learn about sensing capability and even design your own payloads. This new robot lets you get to the innovative part early and with a head start on programming.

The inspiration for Create was the iRobot Roomba Vacuuming Robot which was unveiled in 2002. iRobot later released the Roomba with a new serial port and open interface which attracted inventive tinkering by robot enthusiasts. Some developers turned the robot into physical variants of video game favorites Frogger and Pac-Man. Other developers tweaked Roomba to move houseplants into sunlight. Another turned it into a webcam on wheels.

Roomba was also used by some of the nation's top research labs to tackle some of the most difficult problems in robotics, like swarm behavior and localization schemes. There was even a book written about using the Roomba to develop new robot ideas.

This feedback demonstrated the need for a standard platform that prevents robot developers from having to start from scratch. With Create, robot enthusiasts no longer have to connect motors to wheels, design motor drives and servo control systems, attach sensors, implement drivers, or get an operating system running.

Founded in 1990 by MIT roboticists, iRobot specializes in designing practical robots. To date, the company has sold more than 2 million vacuuming robots, and more than 800 iRobot PackBot robots are deployed worldwide and are credited with saving soldiers' lives. iRobot has invested millions of dollars and countless hours to develop robot technology, and translated this blood, sweat, and tears into the affordable, programmable platform the industry was asking for.

Many have pointed out the parallels between the robot industry and the PC industry. In the beginning, if you were interested in computers, you purchased and assembled a kit. Now, if you are interested in computers, you work on the software, solving computer problems, developing solutions that people can use on their computer, rather than building a computer from scratch.

While many robot enthusiasts like to spend time building, we believe the more fun and challenging problems are robot problems: What should the robot do, and how should it do it? What sensors should it carry and what actions should the robot take?

The Road to Create

iRobot Create features 10 built-in demos and 32 built-in sensors that allow you to control the robot and experiment with robotics. An open cargo bay and 25-pin expansion port let you add your own sensors, grippers, wireless connections, computers, or other hardware. The fully documented serial protocol provides full access to sensors, actuators, and on-board scripting functionality. Create is supported under Windows XP via serial port.

A variety of methods and programming languages can be used to control Create. Access to all of the robot's sensors and actuators over a serial port is provided via the iRobot Roomba Open Interface. Beginning users can observe the robot's behavior in one of 10 demonstration modes or program the robot directly by downloading short scripts with any basic terminal program. More advanced users can write custom software using a variety of methods that take advantage of the robot's "streaming sensor data" mode for more control of the robot. Highly advanced users can write programs in C or C++ for the Command Module that attaches to the robot for completely autonomous behavior.

The Command Module contains an eight-bit, 20 MHz (Atmel ATMega 168) microcontroller enabling full programmability of iRobot Create's motors, lights, sounds, and sensor readings. The Command Module plugs into Create's cargo bay connector and provides on-board control with programs you write in C or C++. It has four DB-9 expansion ports for adding your own hardware. It also includes a Command Module CD that contains three example programs and the compiler, USB cable, and Quick Start Guide.

Join the Community

So far, we have seen creative hobbyists use Create to build robots that can clean up clutter, get soda cans from the refrigerator, and even design traditional Indian Rangoli art.

We hope you will be the next great robot developer, and we challenge you to join the Create community at **www.irobot.com/Create**. To give you some idea of what's possible, check out the projects that were developed at iRobot: **www.irobot.com/sp.cfm?pageid=296**. SV

Then and NOW

ROBOTICS ORGANIZATIONS

by Tom Carroll

Every one of us who read this magazine has some sort of interest in building or learning about robots. Once that interest springs up, whether from childhood or later in life, we start to wonder where we can get more information on the subject of robots. These days it is easy to search the Internet about any subject you desire and get thousands if not millions of hits and links to satisfy your questions. In the past, it was libraries and bookstores that were the sources of robot information.

However, after reading all that you could find, you began to wonder "if there was anybody out there" with whom you could talk about your new interest. You may have questions that were not covered under Internet or library sources and you just felt it would be better to talk with someone who had actually built a robot or worked in the field. After two or more people get together to discuss any subject that interests them, it is natural to search out more people with the same interests.

Robots and robotics is a subject that

Robothon — GYRE Project Robot Team from University of Washington work with NASA JSC with zero G Autonomous Robot.

often revolves around schools and universities so students often are the first to organize clubs of interest. A large portion of the robot organizations that you find on 'comp robotics' lists are organized around schools or meet at a school or college. Places of learning naturally nurture a desire for more knowledge.

Forming a Robotics Interest Group

I'll start this discussion about the history of robotics groups by talking a bit about my experiences. I assume that my early interests in robotics were about the same as anyone else. I enjoyed reading Issac Asimov's robot stories as a kid and searched out any 'robot movie' I could find. There were the occasional robot articles in *Popular Mechanics* and similar magazines to whet my appetite, but I wanted to *meet* people — not just *read* about them.

I was a member of the United States Robotics Society that was based in Albuquerque, NM, but my only communications with them was a newsletter. As a young engineer with Rockwell working on various NASA space projects, I frequently went to JPL in Pasadena, CA, where I met several people who shared a similar interest in robotics. There was also a new robotics magazine that had recently moved to the Pasadena area, *Robotics Age*. In late 1977, several of us got together at the magazine's office and decided to start a robotics interest group in the Los Angeles area and tried to figure out a way to let people know. One person of the core group was a member of a cycling group at JPL and he helped us by printing notices to place in 'computer stores' and a notice in the *LA Times*.

In 1978, we had our first meeting at *Robotics Age's* office in La Cañada near Pasadena (actually a home with a big yard in which to meet). I was amazed at the number of people who showed up. I was the last person to say 'no,' so I became the "temporary" Chairman (for three long years) of the *International Robotics Foundation*.

After finding out just where everyone in attendance lived, we decided to move the meeting location further south to a computer store in Norwalk. After we mailed out a few bulletins to our members, we finally had an official newsletter — *The Robot Builder* — at the end of 1980.

Computer stores in those days were more electronics surplus places that happened to sell some of the Apple and Commodore computers available then. Feeling the word "International" sounded a bit pretentious, we decided to call ourselves the *Southern California Robotics Society*. We later moved to the Norwalk Public Library and even later to Cal State University Long Beach. Some of our meetings at the library had over a hundred people in attendance when we had Heath demonstrating the new Hero robots or a similar presentation, but attendance began to fall off.

In 1989, we began anew out at a CAD-CAM school — *MTI College* — in the city of Orange and changed the name to the *Robotics Society of Southern California*. The RSSC now meets at Cal State Fullerton and is a very active group.

In the early days of the Southern California Robotics Society, I first began to attend the yearly RI/SME robotics conferences that were held in alternate years in Detroit and Chicago. It was at these meetings that I got to meet others who had robotics interests that were outside the basic industrial and manufacturing concerns of the Society of Manufacturing Engineers. My division of Rockwell did not need robots for mass production; we made ones and twos of very complex and expensive products like satellites and space shuttles. My work interests in robotics were to build robots for space applications. From a hobby stand point, I built robots as action props for movies and just to roll around my house to keep me, my family, and guests happy. These other people felt the same way — it was a lot more fun to make and program your own experimental robot than to install a huge Cincinnati-Millicron robot on a car factory's floor to weld car doors.

We tried to meet at every year's conference — usually in a hotel room — to discuss what other individuals and groups were doing across the country. We formed a group, REAL that stood for *Robot Experimenter Amateur League* that was to encompass all groups across the country and was to act as a means of exchange of information between the groups. We liked the 'REAL' part of it to let people know that robots that interacted with people were real.

In those days, robots were either creatures on a movie screen or robots spitting sparks or paint in a car factory. John Gutmann of the Atlanta Hobby Robot Club and I tried to keep this group alive and we actually got out a few newsletters. REAL lasted a few years and slowly sputtered out, though there are some who still refer to it.

Robotics Industry Recognizes Ties to Non-Industrial Robots

In the '80s, when there were many robot companies based in the United States, as well as in Japan, the robot industry as a whole saw the need to expand its horizons beyond the typical factory floor. In those days, non-industrial robots were called *personal* robots as most applications centered on the home to serve individuals. Joe Bosworth, the founder of RB Robot Company (maker of the popular RB5X experimental robot) and Nelson "Nels" Winkless, one of the first experimental robot article and book authors, formed the National Personal Robot Association. In 1984, the NPRA held the first annual International Personal Robot Congress (IPRC) in

Robothon — Start kids early in robotics with LEGO.

Albuquerque. Walking amongst the over 3,000 attendees, I suddenly felt at home with my interests. After attending many RI/SME conferences over the years displaying only the industrial aspects of this new science, I realized that the IPRC people truly grasped robotics for the experimenter.

The news media jumped on the new directions that robotics were taking with headlines such as: "Robots Enter the Home" or "Move Over Rosie <robot maid on the Jetsons>, Robot Servants are Here to Serve You." In a quote from the **megadroid.com** site of a few years ago: *"It was 1984, the height of the Personal Robot Age. Robot companies were springing up everywhere; they were the hot tickets. To showcase their products, the first annual International Personal Robotics Convention was held. All the big players were in attendance. It must have truly been a site to behold. And then, shortly after, it all came crashing down and the era was gone as fast as it started."*

My presentations at the various RI/SME conferences seemed to keep the audience interested as I always spoke about non-industrial aspects, but the over 500 IPRC seminar attendees held onto every word that I and all the other speakers uttered. I distinctly remember Nolan Bushnell of Androbot fame standing before us wearing a white suit (think of a tall, dapper Colonel Sanders) and talking about his robots and the Maxx Steel toy robot while keeping us all enthralled. Wow, could that guy speak!

Robot experimenters wanted to know more and many traveled thousands of miles to be there. Many came to *see* author Issac Asimov of *I-Robot* fame speak in person only to find out that his 'presence' was by teleconference on a large video screen. He was afraid to fly so he declined to be there in person. I later met him at another conference to which he took a train. Nonetheless, everybody went away quite happy at their experiences, except for one dear little old lady who sort of yelled at us as we posed for a group photo in front of all the robots placed in a line at the convention center. "You all should be ashamed of yourselves," she exclaimed. "You're trying to play God by creating all these creatures in man's image. Shame on you."

The next year, the conference was held in San Francisco, CA, and the crowd seemed a bit smaller though there were certainly a lot of home-built robots in attendance. After all, it was next to Silicon Valley — the birthplace of the personal computer. One person had an autonomous *flying* robot (well, a large trash bag filled with helium and a tiny three-axis set of propellers and sensor suite to allow it to roam about). There were other machines that were so unique that they literally 'blew me away' and others just sat there and 'blew' fuses and smoke. Certainly the 'biggies' were there — Androbot, RB5X and Heath, and a handful of the smaller companies — but the initial excitement from '84 seemed to be missing. Megadroid.com's later observation was right. The bubble seemed to have burst on personal robotics, at least for the '80s.

Later conferences were held in conjunction with the annual "Robots" conferences of the RI/SME in Chicago's McCormick Place or Detroit's Cobo Hall, but the excitement had worn off. As pressure would have it, the National *Personal* Robot Association became the International *Service* Robot Association and later the *National Service Robot Association*. This title gave it a worldwide appeal and removed the hobbyist connotation that some felt about the word 'personal.' Besides, too many people made jokes about the conference title: IPRC, though I doubt that the Royal Crown Cola Company found it humorous.

I was invited to join the NSRA board of directors in 1987 by Jeff Burnstein, Managing Director, and we had one of our final meetings at McCormick Place in Chicago that April. Rockwell gladly sent me to the industrial conferences and paid all my expenses but the 'personal' or 'service' facets didn't excite my management, and the same applied to the others in attendance. Once grouped under the 'wings' of the Robotic Industries Association, the NPRA/NSRA was based in Ann Arbor MI, however, it sadly faded away in the early '90s. Many key people in the industrial aspects of robotics became disenchanted with the personal and service aspects in the '80s and early '90s. The US industrial robot companies were 'dropping like flies' or being bought out by European and Japanese companies. This, of course, was before the boom in service robotics in the mid '90s.

New Robotics Clubs Are Formed all Over the US

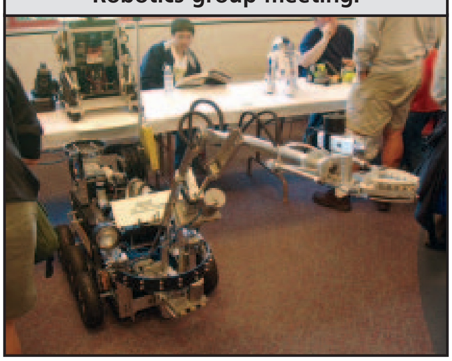
Star Wars and similar movies, combat robots, and sophisticated robot toys have awakened the public to the reality that robots have finally entered the home, or at least the combat arena. Roombas clean our houses as Aibos dance about. Sophisticated robots roam the surfaces of several planets as surgeons on earth use robots for complex operations. Military robots traverse battlefields in Iraq while others fly over the skies of Afghanistan searching for terrorists. LEGO Mindstorms, VEX robotics kits, and Parallax Boe-Bots are the basis of many a science teacher's classroom across the nation.

Robotics enthusiasts watched Will Smith battle computer-generated robots on the big screen and think to themselves, "some day I am going to build a robot like that for *real*." High school kids compete in the many FIRST robot contests generously brought to the world by Dean Kamen, developer of the Segway Transporter.

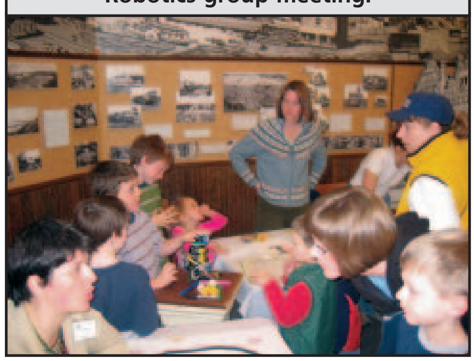
With this new interest in robotics booming, there are literally hundreds of robotics interest clubs across our nation, Canada, and the rest of the world. Some cities have two or three to choose from. Some concentrate on combat robotics while others — like the Bellingham Artificial Intelligence and Robotics Society — have significant interest in AI and autonomous machines. The very large and active Dallas Personal Robot Group has several splinter group meetings a month in their own building and lab in addition to their main meeting, and often serve food to the attendees.

The large Seattle Robotics Society is typical of many groups in that it meets monthly at a technical college. The SRS also has an annual Robothon exhibition and competition at the Seattle Center near the Space Needle to show the public the exciting advances in robotics. Many other groups such as the Portland (OR) group hold similar events. The Trinity College's (CT) fire-fighting robot contest and the Seattle SRS' RoboMagellan competition (a miniature version of the DARPA Grand Challenge)

Large mobile robot at Bellingham Robotics group meeting.



LEGO Mindstorms demo at Bellingham Robotics group meeting.



have been copied by other groups and enjoyed by thousands.

So, you want to join a robotics group. Search the Internet and comp.robotics-type sites for one near you or form your own. You don't have to own your own space like the DPRG or have Texas-sized feasts like those guys; borrow a classroom from a technical school and go to a pizza place afterwards.

You don't have to collect dues; the SRS has one of the largest memberships of any group but most are subscribers of the Internet newsletter and never get to attend a meeting. The SRS obtains operating funds from the Robothon raffles, T-shirt sales, and donations from various companies and individuals.

People always ask me which is the largest group and who was first? The SRS is big but there may be far larger groups such as IEEE's robotics group. Who really knows or cares? As far as who was first, I know my old Southern California group was *not* the first back in 1977; Glenn Norris' *United States Robotics Society* really came first. The way I feel: Albuquerque, NM is the *true* home of the first personal computer — the MITS Altair — *and* the first amateur robotics group — the USRS. It took Bill Gates and Paul Allen camping out in a dingy motel room in Albuquerque to create the first useable personal computer's software. Glenn Norris also struggled to bring about interest in the first personal robots. When Dr. Walter Tunick took the USRS over and moved it up to the Bay Area and changed its name, it really became a new society.

Let's not worry about biggest-best-most food-most noteworthy members and other silly things. Any group is the best for its members. The bottom line is, there is no excuse for you not to be a part of a great robotics group. **SV**

Advertiser Index