COOL ROBOTS • SUMO ROBOTS • ROBOT NAVIGATION

SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

www.servomagazine.com

MAY 2007

Tell Your Robot To Get A Grip!

- Robot Spider Crabs
- Ultrasonic Microsensor Comparison
- Fabulous Robots With Prefab Parts
- Build The Ultimate Remote Control
- Rummaging In The Robot Reliquary

U.S. $5.50 CANADA $7.00

JAMECO ROBOT STORE

Perfect projects for kids of all ages!

robotic kits

chassis

servos

passives

optos

integrated circuits

semiconductors

muscle wires

connectors

motors

test equipment

software

books

Robotic Kits *and* Components...
...there's something for everyone!

Robotic kits help you and your child to experience and learn about perception and control using a variety of sensors and actuators. Challenge yourself with Jameco's selection of fun and interactive kits! You or your child can assemble the kits and then enjoy endless hours of discovery.

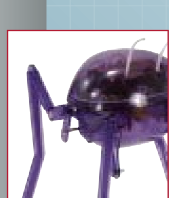
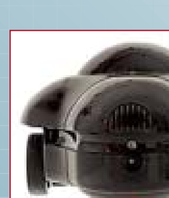
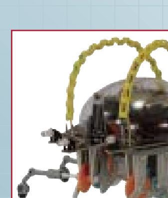
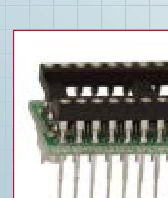
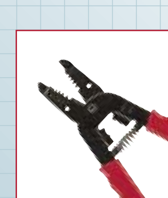
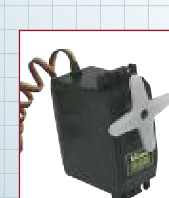
Check out our unique selection of robotic kits at **www.RobotStore.com!**

- **Robot Insects & Animals**
- **Solar Robots**
- **Listening, Touching & Seeing Robots**
- **Hackable Robots**
- **Programmable Robots**
- **Educational Kits**
- **Legged and Wheeled Platforms**
- **OctoBot Survivor Kit**

At Jameco's RobotStore you can get the world's most complete robotic offering—*all in one place!* You'll find kits for all ages and experience levels, along with gear boxes, servos and chassis, for those who are ready to take on the extreme.

WWW.ROBOTSTORE.COM/SRD
1-800-374-5764

Call for your free catalog today!


SERVO MAGAZINE

Features & Projects

36 **Haptics, Telepresence, and TeleRobotics**
by Bryan Bergeron
An introduction to the features and design challenges associated with haptics by way of a telerobotic gripper.

41 **Cool Robots**
by Ralph Lorenz
Dull, dirty, and dangerous missions are perfect for these "square" robots.

44 **Seeing With OpenCV**
by Robin Hewitt
Part 5: Implementing Eigenface.

51 **Build the Ultimate Remote Control**
by Michael Simpson
Create a custom control system that can be connected directly to your robot via a wireless Zigbee module.

58 **Ultra-Sonic Microsensor Comparison Project**
by Kyle Haston
This ECE undergraduate shares the results of his class project.

64 **ROBOGames Prep**
by David Calkins
This month: Sumo Robots.

Columns

08 **Robytes** *by Jeff Eckert*
Stimulating Robot Tidbits

10 **GeerHead** *by David Geer*
Lurking in the Shadows: Leeds Works With Robot Spider Crab

14 **Ask Mr. Roboto** *by Pete Miles*
Your Problems Solved Here

19 **Different Bits** New!
by Heather Dewey-Hagbord
The True Beginner's Guide to the Spartan3E Starter Kit

68 **Twin Tweaks**
by Bryce and Evan Woolley
Rummaging in the Robot Reliquary

72 **Robotics Resources**
by Gordon McComb
Fabulous Robots With Pre-Fab Parts

76 **Appetizer** *by Chris D. Odom*
Environmental Sensing in a Robotics Curriculum

79 **Then and Now** *by Tom Carroll*
Robot Navigation

ENTER WITH CAUTION!

28 **The Combat Zone**

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

05.2007
VOL. 5 NO. 5

SUMO — PAGE 64

Departments

06 Mind/Iron
07 Bio-Feedback
24 Events Calendar
26 New Products
27 Robotics Showcase
43 Robo-Links
62 *SERVO* Bookstore
82 Advertiser's Index

ULTIMATE CONTROLLER — PAGE 51

Mind / Iron

by Camp Peavy

Do you need an android? Maybe not today, but someday you might. Do you need a car? A cell phone? Internet access? Of course you do, but until recently the latter were luxuries or not even available. In the 1900's, you did not even need an automobile; horse-and-buggy was the standard. Then all of a sudden, you needed a car. If you were to describe to someone in the 1950's some of the products of today, they'd think you were crazy ... except they would wonder what happened to the flying cars and robots. Functional universal robots and, in particular, androids have been desirable throughout history; real robots like this will be the culmination of the computer revolution and, believe it or not, we might even have flying cars.

Why do we need robots? We need robots to bring us stuff, to put stuff back, to carry stuff, and to do the things that we do not want to do like laundry, dishes, trash and garbage disposal, hedges, farming, security, and even baby-sitting. Sure, we have washers, dryers, and trash compactors and they are robots to a degree, but I'm talking about eliminating the human factor; providing the physical interface between the dryer to the closet and from the dirty clothes hamper to the washer/dryer. For dishes, the process would be from the cupboard to the table; from the table to the dishwasher; and from the dishwasher back to the cupboard. Not only will this robot be good for bossing around, but with voice recognition, synthesis, and access to the web it'll be a great companion with instant answers to any question; an automated conveyance system designed to look and act like a person. Not necessarily intelligent, but infinitely trainable and therefore useful. Besides, "intelligence" — like beauty — is in the eye of the beholder.

With the current pace of technology propelled by the Internet, it is inevitable humanoid robots will become commonplace in factories, warehouses, offices, and nursing homes (at least the one in which I plan on staying). They will do for physical objects and services what computers and the Internet have done for information and communications; there will be huge productivity gains. In time, the robotics industry will dwarf the computer industry as service jobs are eliminated and robot operators and technicians proliferate in the same way bookkeepers and used-car salesmen became programmers and computer technicians. Along the way, there will be hundreds of must-have toys, gadgets, thing-a-ma-bobs, and appliances but you can bet the culmination of all of these activities will be a human-form slave-machine. After the initial growth phase, you will even have to choose between the upscale Apple® analog or the standard "PC" version; it's a vicious cycle.

You will be hard-pressed to resist the temptation to buy one of these handy humanoid devices; to be waited on hand and foot by a tireless and cheerful servant-machine. The humanoid robot will become the ultimate how-did-I-get-along-without-this device; the ultimate remote control. Can you imagine actually having to get up to change channels?

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, Kyle Haston, Bryan Bergeron, David Calkins, Kevin Berry, Michael Simpson, Robin Hewitt, Ralph Lorenz, Steven Kirk Nelson, Camp Peavy, Jeffrey Scholz, Chris Odom, Brian Benson, Evan Woolley, Bryce Woolley, Chad New, Heather Dewey-Hagborg

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

By the 22nd century, going into the kitchen to make coffee will seem just as strange.

If you want to take part in this great adventure, the best place to start is a robotics contest. There's nothing like a deadline to force you to create. And the best robotics contest in the world is taking place in San Francisco on June 15, 16, and 17: RoboGames 2007 (**www.robogames.net**). It may not be too late (four weeks ... plenty of time!); registration closes on 5/16/2007. If you're interested, go here NOW: **www.robogames.net/registration.php**. If it is too late, start building now anyways. You are never more than a few months from some kind of robotic event. Go to **www.robots.net/rcfaq.html** for the latest listings. If you're in the San Francisco Bay area at least go to the show; it's ROBOtastic!

Subscribe to *SERVO Magazine* and submit an article. Don't just be a reader! Participate! Build something! This is a fun, educational, and potentially practical hobby with big-time potential. It is a slow-grow industry as the learning curve is both deep and wide, but timing is everything. Besides, after the robots take over, it will be too late. Remember, the journey is the reward. Where else can you have an impact on a technology primed to change the world so radically?

Looking forward ... Camp **SV**

BIO-FEEDBACK

Dear SERVO:

This email is in regards to the article "Seeing With OpenCV, A Computer Vision Library," by Robin Hewitt. First, I loved the article and I am anxiously waiting to see the the remaining parts of the series — great job! Second, I thought it might be useful for your other readers that don't know much about C/C++ to find out that Microsoft offers Visual C++ Express for free on their website (**http://msdn2.microsoft.com/en-us/visualc/aa336404.aspx**).

Greg Vowles
Senior Engineering Technologist
University of Toronto

Dear SERVO:

I am starting to build an AI robot using the Lispworks program designed for the Leaf project and saw the article about waypoint navigation with DarkBasic Pro. I have DarkBasic Pro and am looking for the code called robot path finding.

Fred Miller

*Response: The Dark AI extension pack for DarkBasic Pro — available online from **www.thegamecreators.com** — includes several waypoint examples. The Developer Forum has additional examples of waypoint navigation and other examples that are directly applicable to robotics.*

Bryan Bergeron

Robytes

by Jeff Eckert

Amphibious Bot Mimics Salamander

Salamandra Robotica, transitioning from walking to swimming, on Lake Geneva. Photo by A. Herzog, courtesy of Biologically Inspired Robotics Group, EPFL.

This month's offering in the category of artificial slippery things is Salamandra Robotica, created by researchers from Ecole Polytechnique Federale de Lausanne (EPFL, **www.epfl.ch**) and Inserm (**www.inserm.fr/en/inserm/**). It is basically a robotic model of a salamander's locomotion system, aimed at figuring out how the critters crawl and swim and to demonstrate "how robots can be used to test biological models, and in return, how biology can help designing robot locomotion controllers." A detailed explanation can be found at **birg.epfl.ch/page65446.html**.

Oh, and in case you haven't heard, EPFL has been offering downloadable "Talking Robots" podcasts for a year or so, the latest of which is an interview with Metin Sitti on nanomaterials for robotics. To access it, visit **lis.epfl.ch/index.html?content=resources/podcast/index.html**.

Knowing Your Cenote from a Hole in the Ground

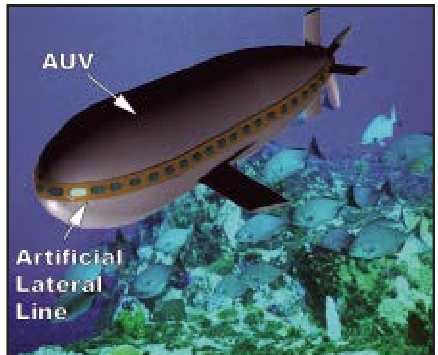
The DEPTHX team poses with the probe in front of the cenote La Pilita during a test run. Photo courtesy of Jackson School of Geosciences, University of Texas at Austin.

As of this writing, scientists have returned to the world's deepest known sinkhole, Mexico's Cenote Zacatón, to resume tests of a NASA-funded robot called DEPTHX (for Deep Phreatic Thermal Explorer, if you must know). The bot, funded by NASA, is designed to survey and look for life in one of Earth's most extreme regions and potentially in outer space.

On the terrestrial side, some of the things DEPTHX will be looking at are some microbes that float in deep water and line the rocks in Zacatón. The relatively unknown microbes are so far from any penetrating sunlight that they must derive energy from another source, such as nutrients originating in hot springs. The scientists surmise that other unknown life forms may be waiting in the 1,000-ft deep hole.

DEPTHX is unlike other deep-sea probes in that it is autonomous, and it creates 3-D maps of the areas it explores and uses the maps to return home. Results from the $5 million project should be flowing in as the exploration continues, so stay tuned to **www.geo.utexas.edu/zacaton/DEPTHX/DEPTHX_home.htm** for details.

Learning from Fish



Artificial lateral line that may improve navigational capabilities of underwater vehicles. Photo courtesy of Chang Liu.

Also set for action in the depths is a system of biologically-inspired sensors that may supplement traditional sonar and vision systems in subs and AUVs. It seems that fish rely on a row of sensory organs along the sides of their bodies — known as the lateral line — for guidance in synchronized swimming, predator avoidance, and picking up prey. This inspired a research team at the University of Illinois at Urbana-Champaign (**www.uiuc.edu**) to develop an artificial version.

The artificial lateral line is an integrated array of microfabricated flow sensors that detect changes in water pressure and movement. Each sensor is integrated with MOS circuitry for on-chip signal processing, noise reduction, and data acquisition. The largest array built so far has 16 flow sensors with 1 mm spacing, and each sensor is 400 m wide and 600 m tall.

In tests, the line was able to localize a nearby underwater vibrating source and could detect a hydrodynamic wake (such as the wake formed behind a propeller-driven submarine) for long-distance tracking. The project was funded by the US Air Force Office of Scientific Research and (as usual) DARPA.

And Hunting for Birds

1944 photo of ivory-billed woodpecker.

Reverently referred to as the "holy grail of birdwatching," the ivory-billed woodpecker was thought to be extinct since the 1940s, but in 2004 some biologists claimed to have sighted one in Arkansas. There was a lot of skepticism, but then last September some scientists from Auburn and Windsor Universities not only spotted the elusive creatures in Florida, they made some audio recordings of the birds' distinctive double knock.

Since 2004, the bird has been sighted more times than Elvis at Dunkin' Donuts, and a slew of birdwatchers and websites have been trumpeting its return. However, many ornithologists remain skeptical.

To try to resolve the matter, a group of academics, with the assistance of the Arkansas Game and Fish Commission, have set up a robotic camera system near the Cache River National Wildlife refuge that will keep its eye out for the elusive woodpeckers. The system, powered by a 69 kV transformer, shoots at 22 frames per second with two to three megapixels of resolution per frame. It also performs real-time high-res video analysis to track flying birds.

Admittedly, it could take years to catch one on video, if it happens at all. But enthusiasm is high. If you want to join the madness, a good starting place is **www.birds.cornell.edu/ivory**, where you can learn more about the subject, support Cornell University's mobile search team, or report a sighting. You can even get an ivory-bill T-shirt for a donation of $100. Such a deal!

Robotic camera system. Photo by M. David Luneau, Jr., courtesy of US Fish and Wildlife Service.

Victorian Robots?

If you're interested in the ancient history of robotics (veritable or otherwise), drop by **www.bigredhair.com** and click on the "robots" link. There you will find "the most extensive collection of images and information on Victorian-era robots to be found in the whole World Wide Web." This includes information on Boilerplate (a prototype soldier), The Electric Man, The Steam Man (noted as the world's first robot), and The Automatic Man.

Clicking on any of the images will bring up a detailed narrative and additional illustrations, all of which seem to be the creation of artist Paul Guinan. Too bad I didn't spot this in time for the April issue ... *(Note that these are all fake. — Ed.)* SV

GEERHEAD

by David Geer

Contact the author at geercom@alltel.net

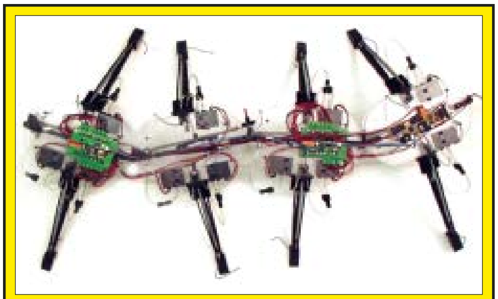
Lurking in the Shadows

Leeds Works With Robot Spider Crab!

Shadow Robot Company (London, UK) and its technical director, Rich Walker, have a lot of experience designing, building, demonstrating, and performing (performance art, that is) robots. Shadow is a reliable source of research robotics. Researchers count on Shadow's equipment, like the Dexterous Hand — arguably the most capable robot hand in the world — and the air-pumped robot muscles.

More to the point, Shadow has designed and constructed more than one set of robot spiders already, including the Zephyrus and Zephyrus Two. The robot spider crab will be similar and yet unique.

Close-up of spider robot technology.

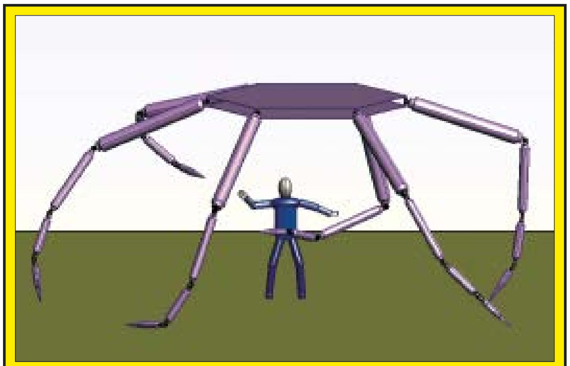


Do the Robot Spider Crab Dance!

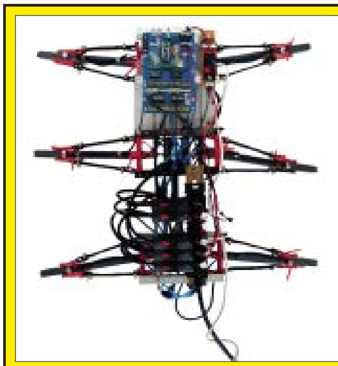
The six-legged, 3 m by 4 m crab will be dancing in a new performance project of the Faculty of Performance, Visual Arts, and Communications at Leeds University. A single crab arm will be constructed and interfaced with a virtual simulation of five other limbs. The single leg will suspend from the ceiling and dance with human performers by use of its multiple air muscles.

On a grander scale, all this is part of the Emergent Objects project, which uses "performance knowledge to explore and articulate the emergent nature of the interface between technological objects and the human, which they believe is fundamental to the development of new design thinking and practices," according to a recent Shadow media release.

Artist's rendering of potential robot spider crab configuration.



Close-up of six-legged spider robot technology.



Robot Spiders, Materially Speaking

Shadow has formed spider robots from wood, PolyMek, Delrin, aluminum, polycarbonate, and a variety of other materials. According to Walker, with the spider crab, part of the emphasis is on constructing a robot that is very big and "soft" that can interact directly with humans, so the material choices become very interesting. "One of the prototype parts that we are putting together at the moment is made out of the lagging [casing] used for hot water pipes," he says.

Because the robot spider crab will interact with people, there will be many sensors incorporated into it, though what those will be has not yet been determined.

The spider crab's robotic muscles

All photos are courtesy of Shadow Robot Company, Inc.

will be the very same air muscles created by Shadow for the development of all their robots. "The Air Muscle," says Walker, "is a soft, compliant actuator that makes it easy to construct bio-mimetic robots, because it has very similar movement characteristics to a biological muscle."

Gathering for demo of Shadow robot spiders, predecessors to the coming robot spider crab, which will be similar.

Two larger air muscles next to a human hand.

Shadow's air muscles are as small as a penny or as large as a stick of pepperoni. These actuators are lightweight, simple, and experimenter-friendly (soft, no "stiction," easily controlled, and significant in power), according to Shadow.

Air muscles imitate real muscular expansion and contraction by providing a "pulling force." The roboticists actuate the muscles using compressed air; the muscles contract by more than a third of their expanded length.

Air muscles can move levers. With one air muscle, a lever can move to a set angle because the muscle contracts to a given point. In a single muscle configuration, a spring must be used to return the lever to its original position. Two muscles can pull the lever in two directions "with considerable force." In fact, with a power-to-weight ratio of as much as 400:1, a 30 mm muscle can bend a nail, according to its makers.

Rubber tubing and a strong plastic "netting" outer layer comprise the air muscles. The muscle actually contracts (shortens) rather than elongating when filled with air. Air muscles — now 25 years old — are available from the company website (see the Resources sidebar).

According to the company, one of the slightest air muscles, 6 mm in diameter, has "the strength, speed, and fine stroke of a finger muscle in a human hand," while a large muscle of 50 mm can pull down a brick wall. Air muscles require a pneumatic system with valves.

Computer Technologies Not So Crabby

Shadow robot computer systems run Debian Linux with Real-Time Application Interface (RTAI) technology for managing real-time applications. Many of the embedded controllers are built around PIC18 microcontrollers.

"The original Zephyrus ran on a BASIC Stamp, but we found they were very fragile. Eventually, we moved to using a PIC16 on the robot itself," says Walker. Zephyrus-Two used Controller Area Network (CAN) modules similar to those used for the Dexterous Hand; so, Shadow was able to use the CAN protocol from the Hand project for the second iteration of Zephyrus.

"We use our own protocol over CAN that is designed around our experience of what is necessary for robots," says Walker. At a higher level, according to Walker, they SSH [secure shell] in and issue a command.

The robotic control layer software is an in-house blend of the Shadow Robot Company. "The University of Leeds' AI people will be working on some software for the higher-level behaviors, but we don't know much about that yet," says Walker. Most

A small air muscle preparing to bend a nail.

The same air muscle, bending that nail!

ROBOTS AND DANCERS TEACH EACH OTHER NEW STEPS

At a conference on research in performance, according to Rich Walker, technical director, the Shadow Robot Company (Shadow), the company and performance artists from Leeds University demonstrated interactions between dancers and robots using two versions of the Zephyrus spider robot.

Using the original six-legged and an eight-legged spider robot, the dancers gave the audience — mostly advanced senior researchers and practitioners in performance studies — an intriguing display of the potential outcomes of this kind of interaction and research. "Watching an experienced dancer engage with the movements of a simple, yet flexible, eight-legged robot was eye-opening," Walker says.

Walker, Shadow, and the performance artists at Leeds were ready to take the interactive research to the next level. By using Shadow's air muscles, which are flexible in creating almost any size and nature of robot limb and other movement, they will be able to build a larger robot to perform the research tango, so-to-speak. And, that's how they arrived at the robot spider crab project.

RESOURCES

The Shadow Robot Company
www.shadowrobot.com

Shadow Robot media releases, including news about the Robot Spider Crab project
www.shadowrobot.com/news/press.shtml

Scheduled events where you can catch a glimpse of Shadow Robots live
www.shadowrobot.com/news/events.shtml

Robot demo TV clips
www.shadowrobot.com/news/tv.shtml

Need Shadow Robots or engineering for your research or production project? Surf here:
www.shadowrobot.com/shop.shtml

The Faculty of Performance, Visual Arts, and Communications
www.leeds.ac.uk/paci/index.html

Shadow Robot programming makes use of things like Shell Script, the C language, and PIC assembler.

Robots and People — Let's Dance

One aspect of research toward a robot spider crab is the interaction between human dancers and existing robots. Shadow and the Performance Robotics Research Group (PRRG) have been collaborating for years now. The dance/interaction portion occurred as part of a one-week workshop held at the School of Performance and Cultural Industries at the University of Leeds.

An air muscle about the size of a penny.

"One of the sessions," says Walker, "consisted of having two dancers, Liz and Paul, work with two robots: a dinosaur and a six-legged walking robot, Zephyrus. Each dancer embodied the nature of the robot they were working with."

Here are the research findings, according to Walker:

• It was possible for the dancers to explore the space of possible kinematics of the robot. "We asked Liz if Zephyrus could stand up, and she was able to work out a movement pattern that would lead to Zephyrus sitting on its back-end. To do this with the robot would have taken many hours of re-design and re-work, and still might not have led to the desired results."

• The researchers presented the dancers with new movement patterns and kinematics. "In trying to embody a dinosaur with a long neck and tail, or a six-legged robot with no "knee" or "foot," each dancer was challenged to reach into new areas of their own movement capabilities. This produced some very innovative dance work from the dancers."

Faux Crab Legs Anyone?

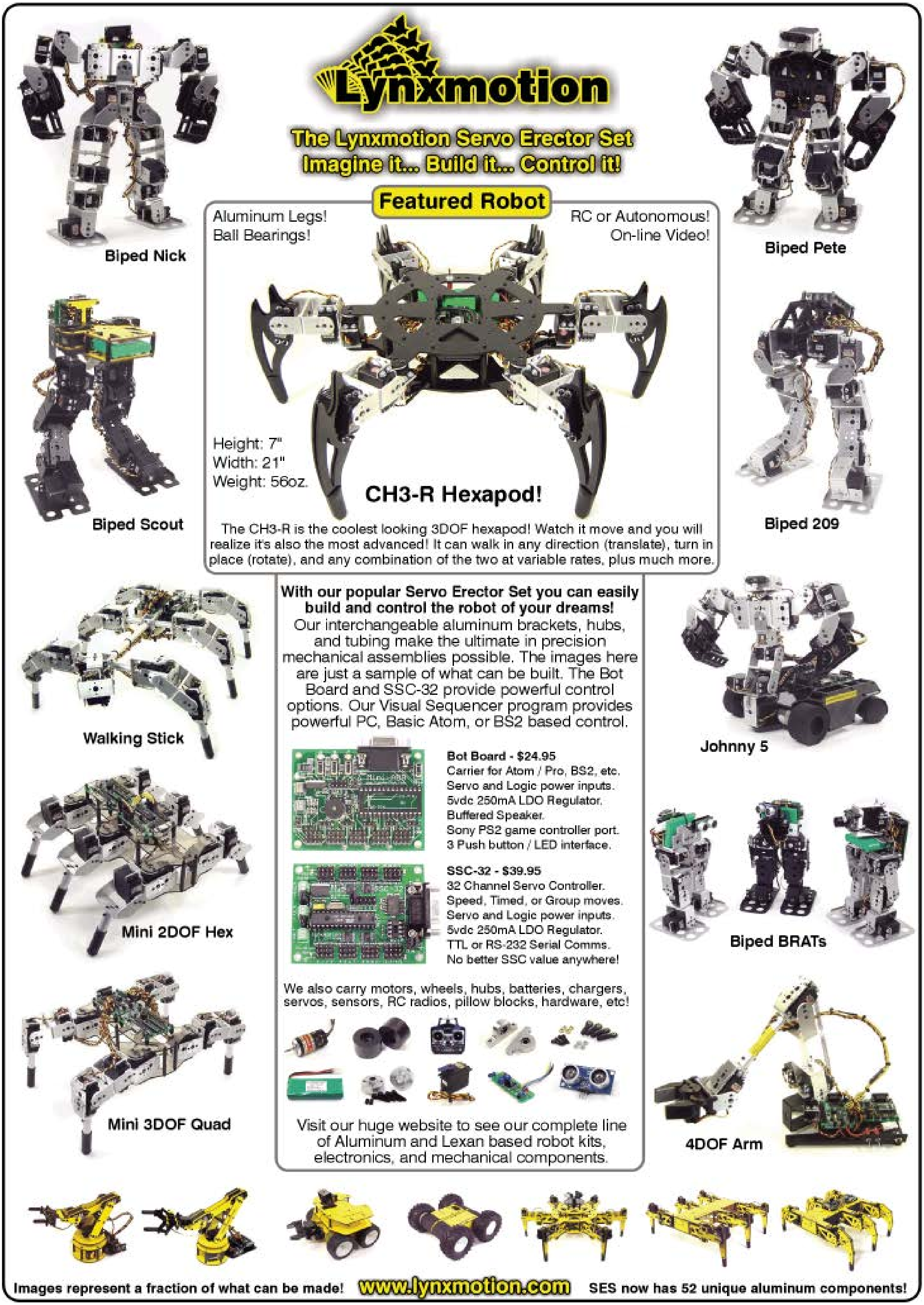
Shadow is building one leg of the 3 meter tall (or more) spider crab. Gazebo, part of the Player/Stage project, will model the other legs, according to Walker. The project is funded through a grant from organizations like the Arts and Humanities Research Council (UK). The grant monies for research total 300,000 (GBP). The project ends with the actual installation and performance in 2007. **SV**

ROBOT MUSCLES NOT JUST FULL OF HOT AIR!

Most of the Shadow Robot Company's robots employ its air muscle technology, which uses compressed air to contract the muscle. These muscles have several characteristics that make them desirable for actuating robot parts like levers.

Air muscles can weigh in at as little as 10 grams (about 0.35 ounces), are cheaper than other actuators, and provide an "immediate response" to their actuation. They are also flexible and powerful.

The muscles are well-suited to "weight-critical" applications and "natural" movement, as well as continued operation when twisted or bent.


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!

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

This is Part 2 of the answer to a question from the April '07 column. I've repeated the question for reference.

Q. I am an electronics teacher at the Orleans Career and Tech Center in Medina, NY. We have entered the Monroe Community College SUMOBOT competition for the past four years and have done very well. They have videos from the competition at their website at **www.monroecc.edu/depts/eng&phy/highschl.htm**.

I have run into a brick wall for parts! Specifically, compound gears that are Mod 1. That is to say that the number of teeth is equal to the diameter in millimeters +1 mm. I have used a gear that has a 40-to-10 tooth combination. They have an outer diameter of 41 mm; inner gear is 11 mm. I could use other combinations, but they must be Mod 1. By the way, LEGO gears and Kelvin ** gears P/N 990174 are Mod 1. My usual supplier is NELNICK.COM, but they won't answer the phone or emails and I fear they're out of business or at least out of town!! Here is the site: **http://nelnick.com//nelnickrobotics/index.php?cPath=21&osCsid=d5f00cb4eedc88af3ebeb383ea78f5e4**.

So, here's the question ... where can I find any Mod 1 compound gears? Or, better yet, how about molding my own? I have old gears or single gears I could make a mold from!

Thanks for any info!

— Bill Leggett
Medina, NY

A. Last month, I talked about gear specification, some places where you can get gears, using two smaller combination gears to get the same gear ratios, and fabricating a combination gear from two other gears. This month, I will be addressing how you can cast a copy of one of the gears you already have.

It turns out that casting a copy of a gear is a fairly straightforward process. All you need is the gear that you want to make a casting of, a mold, and some casting compound. I don't have any Mod 1 gears lying around, so for this demonstration, I will be making a casting of a #25 plastic sprocket that I happen to have (see Figure 1). The procedures will be the same with your gear, or any other part you want to make.

For the casting materials, I chose HobbyMold 150 to make the silicone mold of the gear, and HobbyCast 110 to make a polyurethane casting of the gear. Both of these products can be obtained from HobbyCast.net, a division of Freeman Manufacturing & Supply Company (**www.hobbycast.net** and **www.freemansupply.com**). These products are very easy to use, and they have an excellent video tutorial library showing how to cast many different types of parts, including CNC machining plastic molds. They are a one-stop shopping place for all of your casting materials and support.

The HobbyMold 150 is a two-part silicone rubber with a tensile strength of 650 psi and a flexible shore A hardness of 30. The HobbyCast 110 is also a two-part urethane compound with a tensile strength of 3,300 psi, with a rigid shore D hardness of 69. Figure 2 shows the two pound kits for both the HobbyMold 150 and the HobbyCast 110.

Figure 1. #25 plastic sprocket to be casted.

Figure 2. HobbyMold 150 and HobbyCast 110 molding and casting compounds.

Figure 3. Close-up view of the sprocket's mounting set screw.

Figure 4. Set screw hole covered with modeling clay.

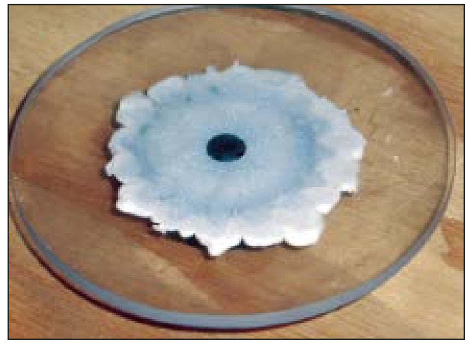


Figure 5. Bottom view of the sprocket mounted to the base of the mold frame using modeling clay.

The following is a description of the steps that I went through to make a casting of the sprocket shown in Figure 1, and is based on the videos I watched on the HobbyCast and Freeman websites. The first thing that needs to be done is preparing the sprocket for casting. Figure 3 shows a close-up view of the set screw that is used to secure the sprocket to a shaft. This needs to be covered up so the sprocket can be removed from the silicone mold without damaging the mold. Some modeling clay was pressed into the set screw hole on the outside surface and on the inside shaft bore surface, and smoothed out to match the contour of the surfaces (see Figure 4).

The next step is to make a mold frame for the sprocket. The videos shown on the HobbyCast website show the base of the mold being made from wood, and a simple paper cup was used for the sides. The Freeman website shows the sides made from small pieces of wood. These are very simple and inexpensive construction methods. For this demonstration, I chose to use clear plastic so that certain steps could be visualized easier. My mold frame consisted of a three inch diameter polycarbonate tube and a 1/8-inch thick polycarbonate disk (all available at local hardware stores). The disk was traced out from the I.D. of the tube and cut from a flat sheet of material.

Next, we need to secure the sprocket down to the base of the mold frame. HobbyCast recommended using some modeling clay for this. Figure 5 shows the bottom view of the sprocket mounted to the base of the mold frame. Make sure that you press down hard on the sprocket to squeeze out any excess clay. You want to keep this layer as thin as possible so that the thickness of the sprocket teeth is not significantly increased.

Using an X-Acto™ knife, remove excess clay from around the teeth and the sprocket's bore (see Figure 6). Figure 7 shows the bottom view of the cleaned up sprocket. Figure 8 shows the base of the mold frame placed inside the polycarbonate tube to establish the sides of the mold frame.

The HobbyMold silicone rubber compounds are mixed at a 10:1 ratio of Part A and Part B, on a weight basis. Based on the volume of the mold frame, I estimated that I would need about 80 grams of the silicone rubber (based on the volumetric yield of the silicone rubber 21.3 cubic inches/pound). The videos show how to estimate these numbers. For simplicity, I used 100 grams of the Part A compound, then added 10 grams of the Part B catalyst to it, and started mixing (see Figure 9).

The HobbyCast urethane is prepared in a similar method, except that its Part A and Part B mixing ratios are 1:1 on a volume or weight basis.

Figure 6. Trimming excess clay from between the sprocket teeth.

Figure 7. Bottom view of the sprocket after all of the excess clay has been removed.

Figure 9. Weighing the Part A and Part B compounds on a digital postal scale prior to mixing.

Figure 8. Polycarbonate tube used to make the sides of the sprocket's mold frame.

Figure 11. Slowly allow the silicone rubber to flow around the sprocket's teeth.

Figure 12. Silicone rubber surrounding the sprocket.

Figure 10. Start with slowly pouring the silicone rubber at the lowest point in the mold.

Figures 10 through 12 show the silicone rubber being poured into the mold. The pouring should be done with a very thin stream starting at the lowest point in the mold (Figure 10), then slowly allowing it to envelop around each sprocket tooth. The slow pour helps to keep bubbles from forming on the surfaces of the teeth. When the silicone was near the top of the sprocket, a thin stream was directed down one side of the sprocket's bore to slowly fill the hole so no air bubbles are trapped inside the bore (see Figure 13). If a bubble forms inside the bore, then the mold can become useless because the mold will tear when demolding and there won't be a complete hole when making the final castings. Figure 14 shows the mold completely filled with silicone rubber. Once this is done, let the mold cure for at least 16 hours before beginning the demolding process.

Figure 15 shows the bottom of the mold after the silicone rubber has cured. Notice the clay pattern has not changed from Figure 7. Figure 16 shows the polycarbonate mold frame removed from the silicone rubber mold. Notice that some of the clay remained on the sprocket surface. Figure 17 shows the sprocket being removed from the silicone rubber mold. At this point, you may need to use a thin piece of metal — like a sewing needle — to push between the sprocket's bore surface and the mold so that mold doesn't stick to the sides of the bore when lifting the sprocket

Figure 13. Slowly filling the sprocket's bore from one side of the bore.

Figure 14. Completing the mold casting process.

Figure 15. Bottom view of the clear plastic mold prior to removing the mold frame.

Figure 16. The silicone rubber mold removed from the mold frame.

Figure 17. Removing the sprocket from the mold.

Figure 18. Sprocket completely removed from the silicone rubber mold.

out of the mold. Don't force it, or you will tear the sprocket's bore off the mold. Work it out slowly. Figure 18 shows the sprocket removed from the silicone rubber mold.

After cleaning the silicone rubber mold, the mold is ready for casting the urethane sprocket. Figure 19 shows a couple of small plastic measuring cups. (Actually, these cups come from a liquid medicine. These make excellent 30 ml disposable cups. I save them all the time.) Since my digital scale only has a five gram minimum resolution, I decided to measure the Parts A and B HobbyCast urethane by volume instead of by weight since I estimated that all I needed was about 10 ml of urethane to fill the sprocket cavity. Figure 20 shows a Popsicle stick mixing the two compounds together.

As with the silicone rubber, the urethane is poured into the mold at its lowest point (Figure 21), then slowly poured (Figure 22) until completely filled (Figure 23). The urethane needs to cure for a minimum of two hours before removing from the mold.

Figure 19. Disposable 30 ml medicine measuring cups.

Figure 20. Mixing Part A and Part B of the HobbyCast urethane by volume.

Figure 24 shows the cured urethane sprocket in the silicone rubber mold. Figure 25 shows the urethane sprocket being removed from the mold. The same cautions need to be observed in regards to the sprocket's bore sticking to the mold. A mold release compound can reduce the friction between the urethane and the silicone rubber. Figure 26 shows the sprocket removed from the mold, and Figure 27 shows the urethane sprocket next to the original plastic sprocket.

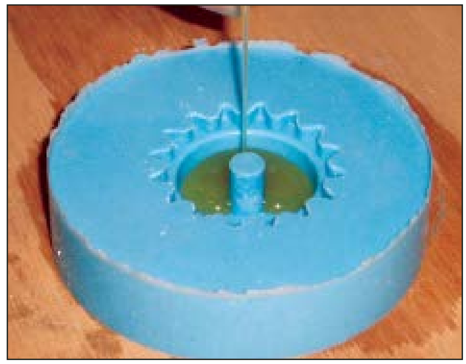


Figure 21. Pouring the HobbyCast urethane into the mold starting at the lowest point.

Figure 24. Cured urethane sprocket in the silicone rubber mold.

Figure 23. Sprocket mold filled with HobbyCast urethane.

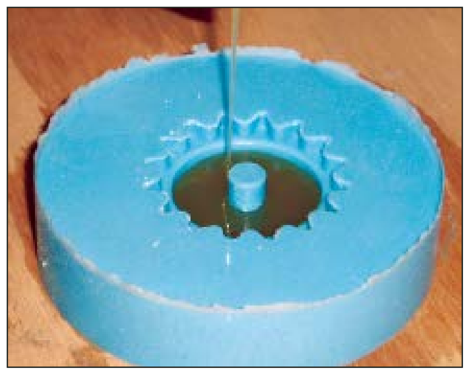


Figure 22. Filling the mold with the urethane.

Figure 27. Side-by-side comparison of the new and original sprockets.

Figure 26. Urethane sprocket removed from the mold.

Figure 25. Removing the urethane sprocket from the mold.

Figure 28. Using an X-Acto knife to remove the burrs created from the clay mounting process.

Figure 29. Marking the location for the shaft mounting set screw.

Figure 30. Drilling the set screw hole.

Figure 31. Tapping the set screw hole.

Figure 32. Close-up view of the tapped hole in the side of the sprocket.

Figure 33. Final sprocket with a #25 roller chain.

Figure 28 shows some post casting cleanup work that is needed to smooth out the rough spots created by the clay when the original sprocket was mounted to the base of the mold frame. An X-Acto knife will work well for carving off the burrs.

The last step is putting the set screw mounting hole into the urethane sprocket. The clay that was used to cover the hole in the original sprocket can be easily seen in the new casting. A small starter hole is poked into the center of the impression with an X-Acto knife (see Figure 29). A number 21 drill is used to drill the tap hole in the side of the sprocket using the mark from Figure 29 for alignment (see Figure 30). A #10-32 tap is used to tap the hole for the set screw (see Figure 31). Figure 32 shows a close-up view of the tapped hole. Note how smooth the threads are inside the tapped hole.

Figure 33 shows the final sprocket with the shaft mounting set screw installed along with the #25 roller chain wrapped around it. Prior to using the sprocket, it should be allowed to fully cure to obtain maximum strength. This takes about seven days.

As you can see, casting a sprocket is a fairly straightforward process. Once the mold is made, dozens of parts can be made from the same mold. These same steps can be used to cast the gears that you are having a difficult time finding. If you ever get a chance to try the casting approach, send a note to the *SERVO Magazine* BIO-Feedback telling us how it went and what we should be careful about so that we can all learn from our collective experiences. **SV**

DIFFERENT BITS

THE TRUE BEGINNER'S GUIDE TO THE SPARTAN3E STARTER KIT

The Spartan3E.

by Heather Dewey-Hagborg

Different Bits is a column which looks at the ways in which the traditionally software oriented domains of Artificial Life and Artificial Intelligence can be transposed to embedded hardware. From genetic algorithms to heuristics and neural networks, we will be examining ways of bringing algorithms inspired by biology to electronic circuits.

Field Programmable Gate Arrays (FPGAs) have recently become a hobbyist friendly medium. Offering hundreds of I/O channels, faster clock speeds, and true parallel processing, they provide an exciting alternative to microcontrollers. With free development software from Xilinx and a Spartan3E starter kit available from Digilent, Avnet, or NuHorizons, you can get started for less than $200.

Unfortunately, the learning curve is steep and online resources are often confusing for beginners. I recently purchased a Spartan3E starter kit from Digilent and found the process of simply getting my first LED blinking to be tedious. Board specific documentation was scarce, abstract, and sometimes blatantly incorrect. This guide is my attempt to remedy this situation for future hobbyists interested in learning about FPGAs.

This tutorial will walk you through the steps to get you up and running with the Spartan3E. All you need is the kit and a PC running Windows XP. Consider this a very thorough "Hello World" for the FPGA.

Getting Started

First things first, if you purchase your kit from Digilent, it will be slightly cheaper and will arrive faster than it will from Avnet or NuHorizons because they keep more parts in stock. However, they will not send you the Xilinx installer CD or documentation, meaning you will have to download everything from the web.

With the Xilinx WebPACK development environment running more than a gigabyte in size, it might be worth it to pay the extra money and avoid the interminable download if you have a slow Internet connection. If you do purchase from Digilent, you will need to register and download the Xilinx ISE 9.1i WebPACK development environment online at **www.xilinx.com/ise/logic_design_prod/webpack.htm**.

While you're at it, grab the Spartan3E starter kit user guide: **http://direct.xilinx.com/bvdocs/userguides/ug230.pdf**.

This document will tell you the pin locations of all the devices on your starter board and give a little nformation about using them. It also includes a small section on programming.

The design flow we will be following in this article consists of five steps:

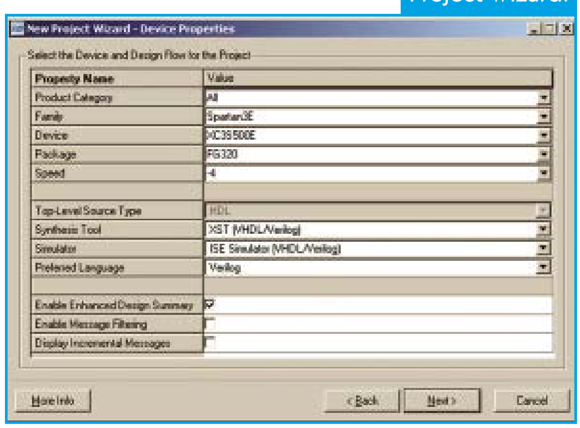
1) Coding (behavioral description)
2) Simulation
3) Pin assignment
4) Implementation
5) Programming

Starting a New Project

Our first project will be an AND gate which takes input from two switches and turns on an LED. Once you have the Xilinx WebPACK installed, open ISE and begin a new project by choosing from the file menu. Getting all the settings right for your first project can be a bit confusing, so we will walk through it step by step.

The first page of

FIGURE 1. New Project Wizard.



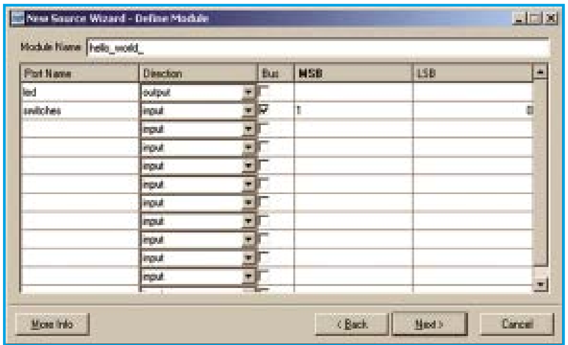


FIGURE 2. New Source Wizard.

the new project wizard will ask you for a name, choose "hello_world" and a directory of your choice. Select HDL for Top-Level Source Type. The second page is the most confusing, for Product Category choose All, for family choose Spartan3E. The Device is XC3S500E and the package type is FG320, both of which can be found written on the FPGA chip on your starter board.

The speed is -4, the synthesis tool is XST, the simulator is ISE simulator, and the preferred language is Verilog. Check the box for Enable Enhanced Design Summary and leave the last two boxes for message filtering and incremental messages unchecked (see Figure 1). On the third page, click the New Source Button.

On the pop-up menu that appears, select Verilog Module and name it "hello_world.v." On the next page, enter "led" as a port name, and change the port direction to output. Then add "switches" as a port name, leave the direction as input, and click the checkbox that says "bus." Enter a 1 in the MSB column and leave the 0 in the LSB column. This specifies how many bits wide our I/O ports are.

When you are finished, your form should look like Figure 2. Click through the wizard, and click yes when it asks you if you would like to create the new directory for the file. Return to the new project wizard and click through until you are finished.

Adding Code

Now it is time to add some code. We are going to create the simplest possible program — a logical AND gate which takes input from two onboard switches and outputs to an onboard LED. In case you are unfamiliar, a logical AND gate has two inputs and one output. It outputs logical low unless both inputs are high, in which case it outputs a logical high (see the truth table in Figure 3). Check out **www.kpsec.freeuk.com/gates.htm** for a simple introduction to logic gates.

INPUT		OUTPUT
A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

FIGURE 3. AND Gate Truth Table.

Double-click on "hello_world.v" in the sources pane to view your new code file. It should look like Figure 4. The words next to // are comments, and you can delete them to compact your code. Four lines of code have been automatically generated for you. The first states:

```
`timescale 1ns / 1ps
```

This line specifies the time units and precision for measurement of delay and time values. Next, it says:

```
module hello_world(led);
```

A module is like a class or function. It has inputs and outputs and can be instantiated by other modules. Our first module is called "hello_world" and has a single output, "led." The following line specifies that led is a one-bit-wide output port and two-bit-wide input port.

```
output led;
input [1:0] switches;
```

We close our module with the "endmodule" line. Note that the first three lines end with semi-colons but the last does not; end commands are never followed by semi-colons.

```
endmodule
```

Let's add a bit more code to create the AND gate. Between the output

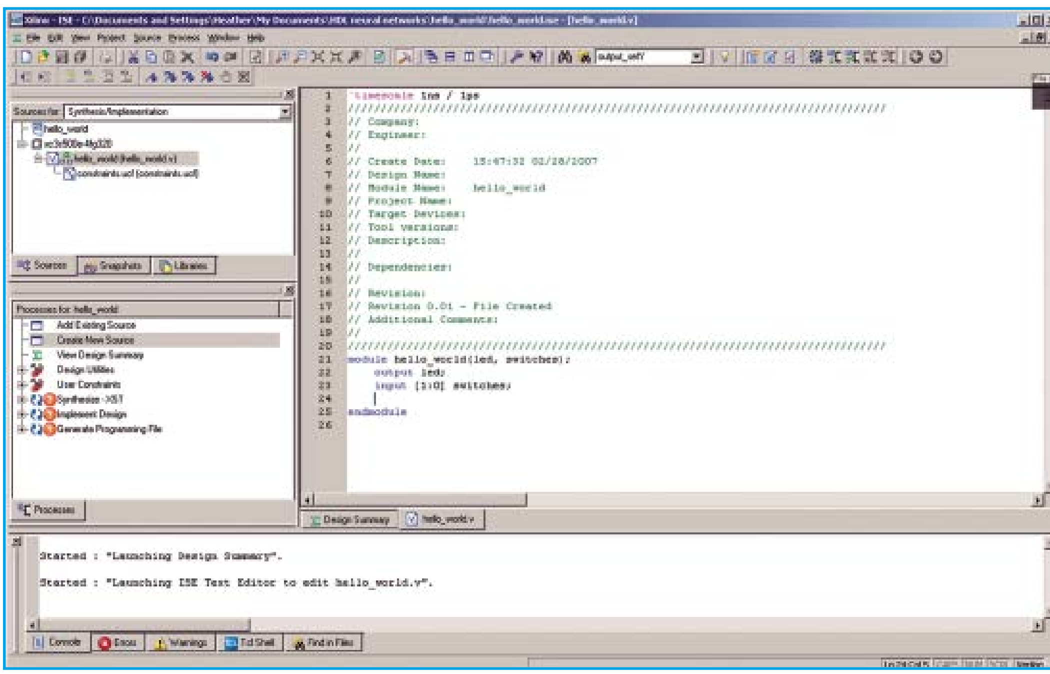


FIGURE 4. Completion of the New Project Wizard.

line and the endmodule line, insert the following code:

```
and
a0(led,switches[0],switch
   es[1]);
```

This line instantiates the built-in primitive AND gate and names it "a0." It sets port led as the output of the gate and switches[0] and switches[1] as the inputs.

It is time to check your code. In the processes pane, expand the menu under "Synthesize – XST" and double-click "Check Syntax." The little wheel should spin and eventually a green checkmark shows up, affirming the impeccable syntax of your code. Next double-click "Synthesize – XST." If the green checkmark does not show up, step back through the code above and make sure you don't have any spelling or capitalization errors.

If your code looks perfect but you are still getting errors, try clicking the errors pane in ISE and following the links listed there. Often, links for specific errors are missing, so if this still doesn't help, try deleting your entire project, rebooting your computer, and starting again from scratch in a different directory on your computer.

Simulation

Now we are ready to simulate our Verilog code module by creating a testbench file. In the sources pane, change the drop-down menu from "Synthesis/Implementation" to "Behavioral Simulation." In the processes pane, double-click "Create New Source." Choose "Verilog Test Fixture" as the module type and name it "test.v." When the wizard finishes, the test fixture should automatically open. If it doesn't, double-click "test.v" in the sources pane.

Let's add some code to simulate sliding our switches on and off. Under the comment which says "//add stimulus here," type:

```
#10 switches = 1;
#10 switches = 2;
#10 switches = 3;
#10 switches = 0;
```

This stimulates our hello_world model with every binary combination of two bits and a delay of 10 in between so we can see how it affects our output. Double-click "Check Syntax" in the processes pane and when the green checkmark shows up, double-click "Simulate Behavioral Model." This feature is sometimes a little buggy. If it says that you already have an active simulation open and asks if you want to close it, click "No."

Now click on the Simulation tab in the display pane and you should see a waveform like in Figure 5. You can zoom in to the waveform by pressing the + magnifying icon in the toolbar. Zoom in until you can see the measurement units for one clock cycle. Note that the output for our LED cycles high only once for 10 ns. Remember, this is because we created an AND gate which is only high when both inputs are high, or when our switches are set to binary 3 in our testbench. The length of time comes from setting our delay timescale to 1 ns in our first line of code, and including a #10 delay on each input change in our testbench.

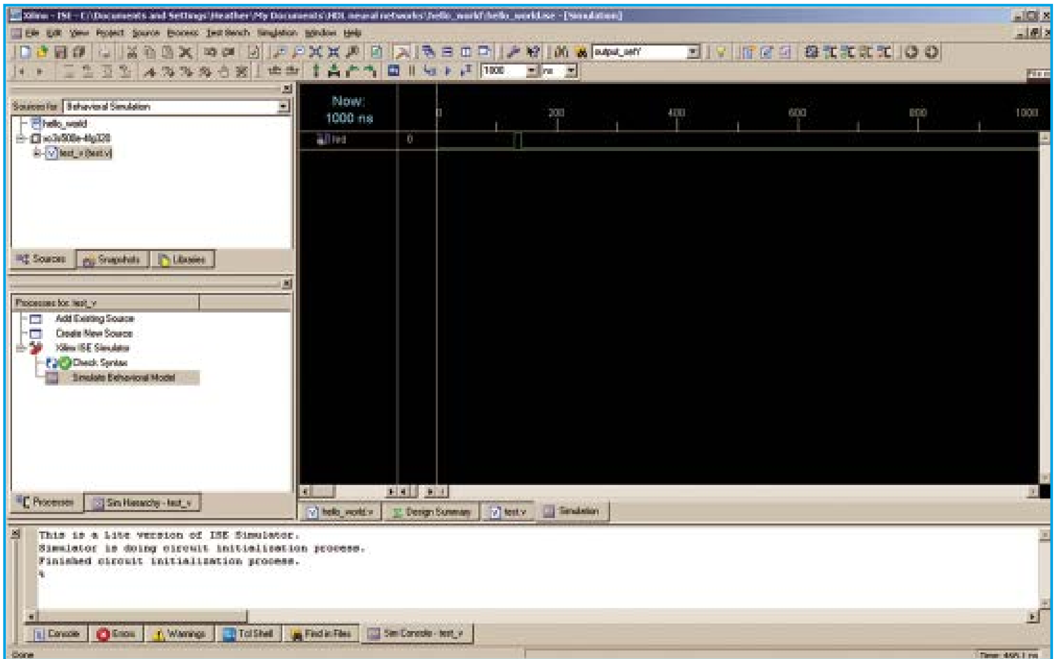


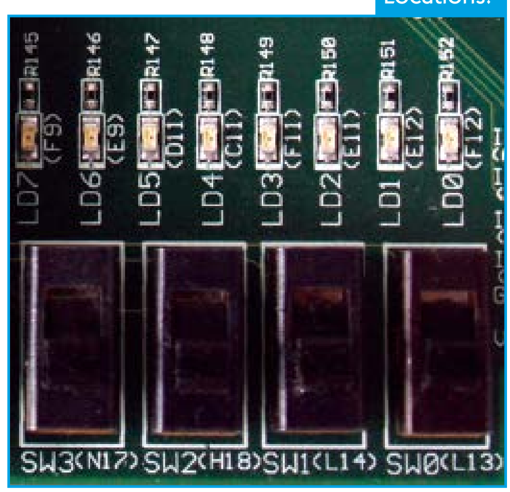
FIGURE 5. Testbench Waveform.

Pin Assignment

With the simulation running correctly, we are ready to assign our I/O pins. Change the view back to Synthesis/Implementation and double-click "Create New Source" in the processes pane. This time, choose "Implementation Constraints File" as the type and name it "constraints." Click through the wizard, then back in ISE, expand the menu under hello_world.v in the sources pane. You should see a file named "constraints.ucf." Click on it and then double-click "Assign Package Pins" in the processes pane.

Xilinx PACE opens up showing a view of the FPGA architecture on the right and a

FIGURE 6. LED/Switch Locations.



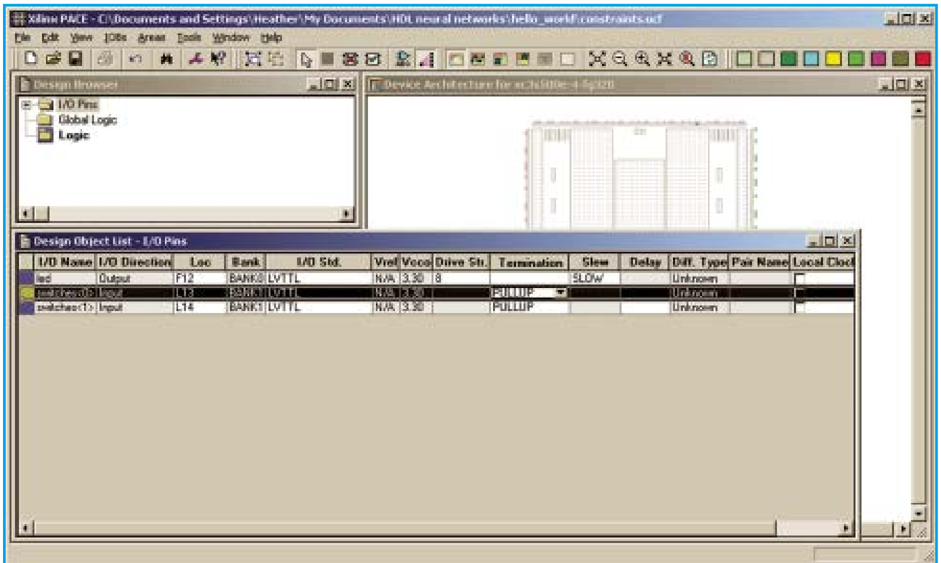


FIGURE 7. Package Pin Assignment in PACE.

design browser and object list on the left. Notice that there are three objects in the list: the LED and both switches. This is where we will assign which pins we want the code to use for our input and output ports.

If you look at your starter board (or Figure 6), you will notice that each LED has an ID printed next to it (LD0-7) and a location written in parentheses below it (F12, E12, etc.). Type "F12" in the Loc field for the row with the led output listing. This will be the rightmost LED on the board. Finish your pin assignment by typing "LVTTL" in the I/O Std. field, "SLOW" in the slew field, and "8" in the DRIVE Str. field. These specifications can be found for each component in the Spartan3E starter kit user guide.

Switch IDs and locations are also printed on the board. Enter "L13" and "L14" in the location columns for switches[0] and switches[1]. These are the rightmost switches on the starter board. Their I/O standard is also LVTTL and their termination is "PULLUP." When you are finished, your Design Object List

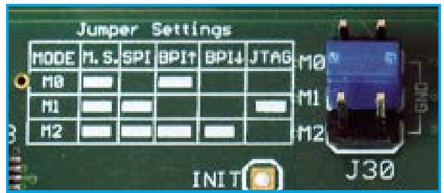


FIGURE 8. Jumper Settings for JTAG Configuration.

should look like Figure 7. Save the file as XST default and close PACE.

Implementation

This step is easy! Returning to ISE, click on hello_world.v in the sources pane and double-click "Implement Design" in the processes pane. The wheels spin and eventually each category in the expanded menu (Translate, Map, Place, and Route) should have a green checkmark next to it. Don't worry, this sometimes takes a while even for simple designs like ours.

Programming

Finally, we are ready to program the FPGA. Unlike microcontrollers, FPGAs do not have onboard program memory. This means that unless you download the program to an external memory device, the chip will lose the code every time it powers up. This section will walk through both techniques: first temporary programming of the FPGA directly via JTAG using the onboard USB port, then programming the Platform Flash PROM and configuring the FPGA to boot from it.

On the top of the starter board, you will notice three jumpers, M0, M1, and M2. These configure the FPGA startup mode. Remove the jumper on M0 and M2 leaving only a jumper on M1 (see Figure 8). Plug in the power supply for

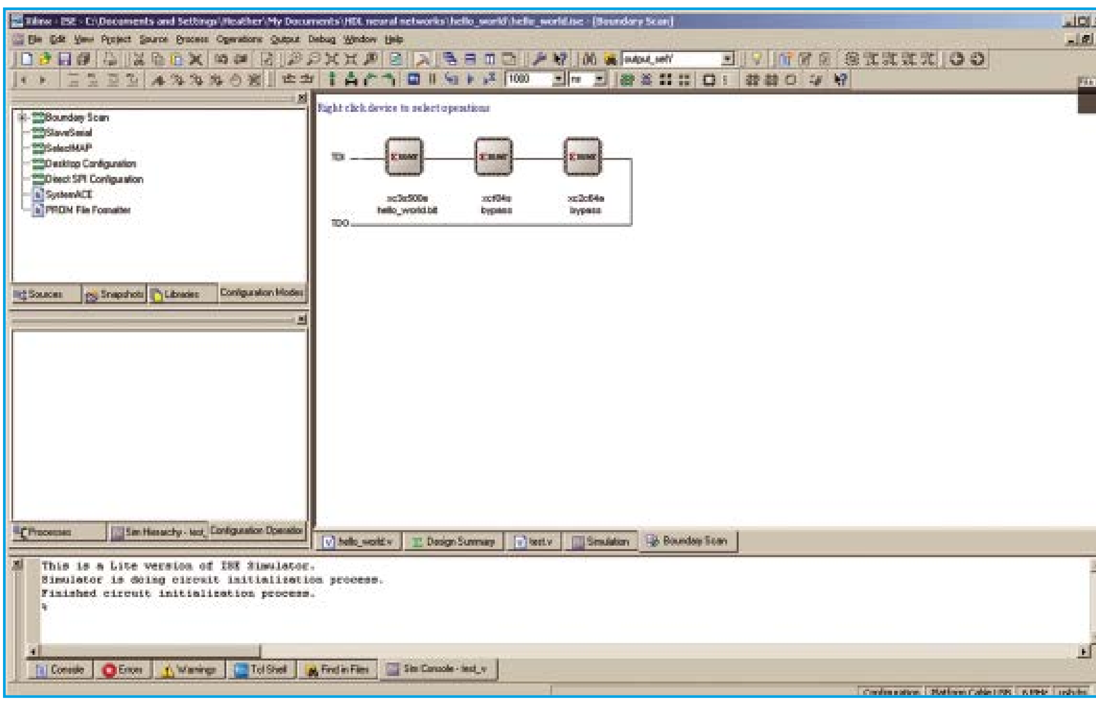


FIGURE 9. Boundary Scan.

the board and switch it on.

Then connect a USB cable between the board and your PC. Annoyingly, I have found that every time I plug my board into my computer, the Windows new hardware wizard starts up. I have to go through all the steps each time or the Xilinx programming application cannot see my device. Once the hardware wizard is finished, return to ISE and double-click "Generate Programming File" in the processes pane. When the wheel stops spinning and the green checkmark appears, double click "Configure Device [iMPACT]."

The iMPACT window opens up and offers you some configuration options. Choose "Configure Devices Using Boundary-Scan (JTAG)." The software will immediately try to connect to your board. If it fails to connect, you will need to quit iMPACT, unplug the USB cable, and try a different port or try going through the Windows new hardware wizard again.

Next, iMPACT will ask you to select a configuration file from a file browser. Choose "hello_world.bit," the programming file we generated in the last step. You may receive an iMPACT warning 2257; just click OK. You will then be asked for configuration files for the xcf04s platform Flash and the xc2c64a CPLD; just click bypass for both. Your screen should now display the setup you see in Figure 9.

Click on the icon for xc3s500e representing the FPGA, then right click and select "Program ...". Click OK on the page that pops up without changing anything. A progress window will appear followed by a message stating that programming has succeeded. You did it! The chip is finally programmed. Slide switch 0 and switch 1 on the starter board up and the LED will turn on. Slide one or the other down and the LED turns off (see Figure 10).

If you turn the power off and then turn it back on, the program will disappear from the FPGA. To give our AND gate some persistence, let's program the onboard platform Flash. Turn the power on your starter board to the off position and add back the jumpers for configuration headers M0 and M2. All three headers should now be in place

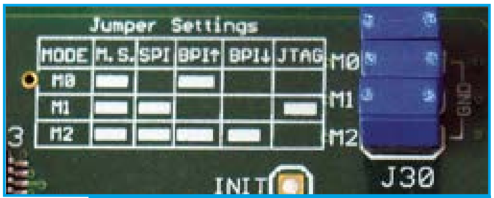


FIGURE 11. Jumper Settings for Platform Flash PROM Configuration.

(see Figure 11).

Double-click "Generate PROM" in the processes pane and then "PROM File Formatter" in the boundary scan configuration mode pane which opens up. Name the file "hello_prom" and set the directory location to your working directory for the project. Click next and check the box to "Auto Select PROM." Click through the wizard and it will ask you to start adding device files. Click OK and choose the hello_world.bit file. This is the only file you need to add, so select no when it asks if you want to add additional device files. Click on the icon for xc3s500e and double-click "Generate File" in the Configuration Operations pane. A message will flash up saying "PROM File Generation Succeeded." Now click on boundary scan in the Configuration Modes pane and right-click on the PROM icon to assign the MCS file you just generated.

When it asks you to select which PROM you are using, choose xcf04s. Right-click on the icon of the PROM again and choose "Program." Click OK in the properties dialog that opens, making sure the checkbox for "Load FPGA" is checked. A progress dialog opens, followed by a "Programming Succeeded" message on the screen.

That's it! Now you can turn the board on or off as much as you please and the AND gate will persist.

Wrap-Up

This concludes our introduction to the Spartan3E starter kit. Now that you understand how the Xilinx settings and configuration work, you can get to the fun stuff. Try expanding the program we wrote to include more switches and LEDs. Add different types of gates for more complicated logic and try constructing a multiplexer and then an adder (see Gerard

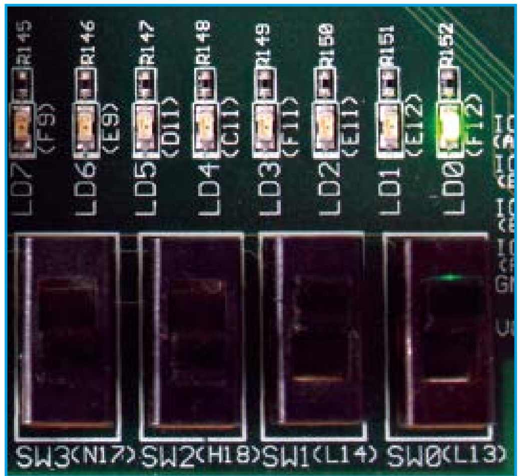


FIGURE 10. Programming Successful! When both switches are up, it should turn the LED on.

Fonte's "Programmable Logic" column in past issues of *SERVO* for more details).

I highly recommend purchasing a Verilog book if you are new to programmable logic. The language is quite different than C and often unintuitive if you are coming from a microcontroller background. You can get a used book; just make sure it covers the 2001 Verilog standard. I also recommend picking up a mating breadboard from Digilent (the part number is FX2BB). It snaps right into the FX2 connector on the starter kit board, making all of the I/O available for prototyping. Have fun! **SV**

RESOURCES

For more details on the topics covered here plus related information, check out these resources:

Spartan3E starter kit user guide: **http://direct.xilinx.com/bvdocs/userguides/ug230.pdf**

Spartan3E datasheet: **http://direct.xilinx.com/bvdocs/publications/ds312.pdf**

Application guides: **www.origin.xilinx.com/xlnx/xweb/xil_publications_display.jsp?iLanguageID=1&category=1211393&sGlobalNavPick=&sSecondaryNavPick=**

Digilent, Inc., supplier of all kinds of starter kits and accessories: **www.digilentinc.com**

NuHorizons, authorized Xilinx distributor (sell the starter kit): **www.nuhorizons.com/**

Avnet, authorized Xilinx distributor (sell the starter kit): **www.avnet.com/**

Fun FPGA projects: **www.fpga4fun.com/**

EVENTS CALENDAR

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

Looking at the event list for May, you'd have to conclude it's international robot competition month. We've got events in Sweden, Germany, Canada, Israel, Switzerland, Portugal, France, and Malyasia. There are also a couple of US competitions scattered through there — like Chibotica in Chicago, IL and NATCAR in Davis, CA.

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

May

4 **Robot-SM**
Sweden
Autonomous Sumo and mini Sumo event. There's no English version of the website, so if anyone can pinpoint the location a little more precisely, let me know.
www.robotsm.se

4 **SPURT**
Rostock-Warnemunde, Germany
Robots race on the official SPURT track.
http://spurt.uni-rostock.de

5 **Eastern Canadian Robot Games**
Ontario Science Centre, Ontario, Canada

This event includes mini Sumo, line-following, walker race, and a new event for beginners called Search and Rescue.
www.robotgames.ca

8-9 **Haifa Robot Competition**
Haifa, Israel
Autonomous robot events for university students, high school students, and amateurs.
http://math.haifa.ac.il/robotics/competition

11-12 **SwissEurobot**
Yverdon-les-Bains, La Marive, Switzerland
This is a regional version of the main Eurobot competition which will be held on May 16-20 in France.
www.swisseurobot.ch

12 **Chibotica**
Museum of Science and Industry, Chicago, IL
Line-following, maze-solving, Sumo, and a robot talent show.
www.chibots.org

12 **Western Canadian Robot Games**
Alberta, Canada
This event includes robot Sumo (western rules), mini Sumo, walking robot triatholon, robot art contest, Atomic Hockey, and a full set of BEAM events.
www.robotgames.net/robot_games.htm

21 **NATCAR**
UC Davis Campus, Davis, CA
Very high-speed autonomous line-following.
www.ece.ucdavis.edu/natcar

NEW PRODUCTS

MOTORS

Digital Robot Servo

Hitec announces the release of their new HSR-5980SG considered by many to be the most powerful servo for the money. It features 417 oz/in of torque at 7.4 volts, super strong, wear-resistant steel gears, and HMI digitial feedback protocol. For further information, please contact:

Hitec Tel: **858 • 748 • 6948** Website: **www.hitecrcd.com**

Planetary Gear Head Motor Servos

Lynxmotion has joined forces with members of the Open Servo team to provide two very powerful planetary gear head motor servos. The first of the two servos is based on a 22 mm planetary gear head motor with 355 oz/in of stall torque. The second servo will use a 32 mm planetary gear head motor with 945 oz/in of stall torque. Both motors have all metal gear trains for maximum power transfer and durability. The new servos are Servo Erector Set ready with no additional brackets required. The control boards will retain all of the Open Servo functionality with I²C control or TTL level serial control. Normal servo pulse control and a trimmer pot to adjust the center position will be added later. The control boards are all digital and use powerful HEXFETs in the output stage. They will also be available separately in 5 amp and 10 amp versions for use in homebrew servos. The control firmware will remain completely open source. These new products will be available in the third quarter of 07.

For further information, please contact:

Lynxmotion Website: **www.lynxmotion.com**

PLATFORMS

Base Rotation For Servos

Lynxmotion is adding a new product that falls under the "Why didn't anyone think of this before?" category. It's an injection molded base rotation for standard size servos. It is made from heavy duty black ABS material and incorporates five 6 mm ball bearings to support the deck. The sturdy construction can easily support a 10 lb payload. The base rotate measures 4.00" at the base, 3.75" at the top, and is 1.88" tall. The top deck has the Servo Erector Set hole pattern so there are several options as to what can be mounted to the top. Some applications include robotic arm base rotate, heavy duty pan and tilt, panning a large sensor array, or even a waist rotate for a Johnny 5 style robot. Any of the most popular Hitec standard size servos can be used. The new base rotate will be available with or without servos, starting at $19.95. The new BR-KT will be available in the second quarter of 07.

For more info, please contact:

Lynxmotion Website: **www.lynxmotion.com**

Robotics Showcase

COMBAT ZONE

Featured This Month

Participation

28 *Propane/MAPP Torch Safety* by Jeffrey Scholz

29 *Fingers and Toes* by Steven Kirk Nelson

Feature

30 *Robots That Don't Break* by Brian Benson

Technical Knowledge

32 *Chains — Putting the Growl in Your Drive Train* by Steven Kirk Nelson

Events

32 *Results — Feb. 12 - Mar. 11*

35 *Upcoming — May and June*

Product Review

34 *LED Lighting Systems* by Chad New

PARTICIPATION

Propane/MAPP Torch Safety

● by Jeffrey Scholz

Propane and MAPP torches are not only used by plumbers to solder pipes together, robot builders use them as well for hardening weapon blades and for heating up titanium to make bending the material into the desired shape easier. Along with the warnings included in your manual, robot builders have a few more things to watch out for:

• Reflected heat — bending titanium into a concave plow? Did you remember that solar ovens are also concave in shape? I've been rudely reminded of this phenomenon several times; it's one you need to watch out for. Even if the piece is straight, you can still reflect heat at yourself accidentally. Remember to angle the torch away from your face and your other hand.

• Reflected heat probably won't burn you, but if it concentrates onto metal touching you (like a metal watch), it can get very hot. Take your bling off before using the torch.

• Hitting a red-hot piece of metal with a hammer throws off sparks, so do not pound any harder than you need to. Again, sparks probably won't seriously hurt you, but it can make you react suddenly and unsafely.

• Your vise is attached to a wooden workbench. Sufficient heat is not likely to reach the wood to cause it to ignite. However, there's


Propane and MAPP torches.

always the possibility of pointing the torch at the wood carelessly (especially when handling the piece you're working on), so dampen the wood with water to retard ignition. Or, you can buy commercially-available flame retardant sheets to cover the wood.

- Some heat-treating requires you to dunk the part in oil or water while it is red hot. However, the part loses its "redness" quickly after you stop heating it. Loosening the part from the vise, grabbing it with pliers, sticking it in oil, and swirling it, all while holding a lit torch in the other hand is not exactly a kosher practice. My first bit of advice is that you not clamp the blade in a vise; it takes too long to remove and the vise wicks away heat. You can hold the part in pliers, but it often bends under its weight while hot and soft. Your best bet is to heat the part on firebrick and slide the part into the cooling solution. Have a partner handy to take the torch away from you while you do this. SV

Fingers and Toes

● by Steven Kirk Nelson

You know, it's funny. Sometimes you get involved with things that are beyond your control. If you're lucky, you both survive them and learn how to respect them, as well. What doesn't kill you, usually teaches you when to run (or at least duck).

Some folks say that herding cats is difficult. They should try herding combat bots and robot builders. They can be just as independent and their wit and claws are usually sharper and stronger. Over the last 10 years or so, I have built many machines, competed in many events, been a judge, written rules, been an EO (Event Organizer), built arenas and helped out at many events. Boy, it's been a heck of a ride.

One of the most difficult jobs I have been assigned to in all of this happy mayhem is working as a safety officer or the head wrangler at robotic combat events. Consider this text the obsessions of an event wrangler.

Robotic combat is a very unique sport. The goal for most that build machines is to test their imagination and intelligence by building an unbeatable and deadly machine. Once it's built, their goal becomes proving their prowess to anyone foolish enough to challenge them in the arena. With this in mind, a good machine is very dangerous and the builder is determined to demonstrate its capabilities. For many of us, this is what the sport is all about. That's all good for the builders, but it is what makes life difficult for the EO and their event staff.

The Wranglers Job ...

The wrangler is responsible for the safety of everyone involved at the particular event. This job can never be taken lightly and it can require a forceful attitude from time to time. Basically, the wrangler's job is to get the machines in and out of the arena as quickly as possible while keeping everyone safe (including them). The wrangler may also have to enter the arena to separate robots that are powered up, stuck on something, or even put out machines that are on fire. I'll tell you what, this can be a bit scary. The wrangler is putting his life in the hands (or twitchy fingers) of the builders and their technology.

What Works for Me

Here are a couple procedures that have always worked for me.

Power-up Procedure:

1) Be kind and friendly to the builders.

2) Check for a radio frequency clip on every machine.

3) Assess the robot design and danger level.

4) Assign robot to an arena position depending on its danger level. Let the builders do their own loading /unloading.

5) Never rush a builder during power-up (within reason) to avoid hasty mistakes.

6) Power-up the least dangerous robot first. Use your best judgment if they are similar in danger level.

7) Allow only one person to power-up one robot at a time. Keep the others outside the arena or behind barricades.

8) Power-up the other robot. Have everyone leave the arena.

9) Do not allow testing or twitching of the bots while anyone is inside the arena.

10) Close the arena doors and allow limited testing of the robots. Time to rock!

Power-down:

1) Do not open the arena doors until builders and bots have calmed down for a few seconds. Often, they are very excited or possibly in shock after a match.

2) Open the arena door and let some fresh air into the arena before allowing anyone to enter.

3) Only allow one builder to enter the arena at a time.

4) Power-down the most dangerous bot first.

5) Have the builders vent their pressurized weapon systems before allowing others to approach the robot.

6) Power-down the other robot.

7) Have the builders shake hands once everything is shut down.

8) Have the builders load up and exit the arena.

I have found as a wrangler its a good idea to remind the builders of these steps for every match. I tend to repeat myself hundreds of times during an event. I also do not change the procedures for any weight class or weapon types. Everyone plays by the same rules. That way, they know what to expect and what is expected of them every time.

Un-sticking Robots

This delightful task is probably the most dangerous job in robot combat. These machines can easily hurt or even kill a person. It is extremely important that the Wrangler has the undivided attention of the builders before entering an arena to un-stick machines. *You must* make sure the builders control their machines. Your life is truly in their hands, which must always strongly be emphasized to them.

Fires

Fires do happen, but they're mostly electrical in nature. I like to keep a CO_2 fire extinguisher handy. Dry chemical extinguishers work as well, but I wouldn't shoot one into a robot unless I absolutely had to since this can make a complete mess.

Safety is everyone's job in this sport. And it should be everyone's number one priority to go home with all of their fingers and toes, even if they have just had their bot kicked! SV

ROBOTS THAT DON'T BREAK

● by Brian Benson

There are many secrets to building a winning robot. There are rules of thumb defining proper weight distribution, frame types, armor types, and materials that span the building community. However, the real key to a winning robot is a robot that doesn't break. It sounds simple and obvious, but it is surprising how many robots lose matches because the robot breaks during an average match.

It's expected that things will break when you fight the really dangerous robots, but not against the tame ones. If you watch carefully during competitions, the veteran builders with a reputation for winning have one thing in common: They are not in the pits after every fight tearing their robot apart and trying to repair it. In most cases, they are simply charging their batteries and looking the robot over. This is because their robot was engineered (not just designed) to survive an average fight without failing from major complications. A well designed and engineered robot will have a better opportunity to win because you'll be able to fully concentrate on strategy and doing well, not just surviving.

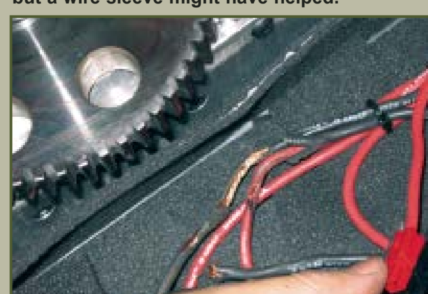
Your Own Enemy

As I said, one of the biggest challenges to building a destructive robot is building one that doesn't break itself. The solution to this comes down to good engineering. It's hard to determine the forces and loads that other robots will exert on your robot but it's easier to find the forces of your own robot. I won't go into the math and physics here, but with some research it's possible to calculate rough estimates. From here, you can determine whether that shaft is really big enough in diameter, if those bolts are going to be strong enough, or if that keyway is going to fail. Take advantage of safety factors; there are reasons they are used in industry. Engineer your robot so that all components will last through at least one competition; this generally means seven or more fights.

FIGURE 1. One pound robot Decidedly Undecided uses a bent steel wedge to deflect spinning weapons.

FIGURE 2. This 220 lb robot used shock-mounted .75 inch aluminum to absorb blows from other robots in order to prevent damage to the drive system and frame.

FIGURE 3. In this robot, the soft wire insulation was no match for getting caught up in the gear. The proper solution in this case is to keep the wires out of the gear, but a wire sleeve might have helped.



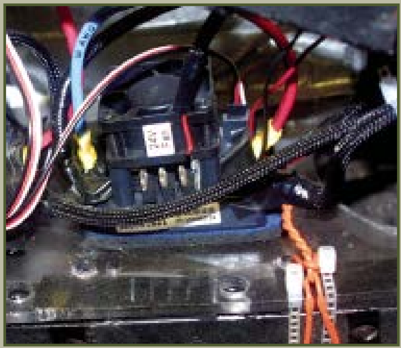


FIGURE 4. This speed controller is shock-mounted on .25 inch Lexan using a cut-up mouse pad.

FIGURE 5. To create a custom radio switch, a simple push button switch, large set screw, and plastic are used.

Design for failure. This means that when you select your components, consider how they will fail and what will happen when they fail. A ball bearing that fails halfway through the match will likely explode or lock up while a bronze bushing that cracks during a match will probably still work for the rest of the fight, just at a decreased efficiency. Try to engineer your robot so it won't fail, and in places you need to make compromises, design it so that when it does fail it doesn't completely compromise the effectiveness of the robot.

Be Tolerant

Next, consider what is going to happen to your robot during a competition. The frame will bend, wheels will get hit, and screws will shear; you need to account for this. Design your robot for low tolerances so that it will be accepting of damage. If you have a frame rail that can take damage, don't make it a key bearing holder for the drive or weapon system. Carefully choose between live and dead shafts. Live shafts have their place, but a dead shaft can double as a frame support and is much more tolerant to misalignment.

Damage Control

Other robots generally try to destroy your robot by transferring as much kinetic energy to your robot as possible. So your goal is to minimize the effect of the energy. There are three ways to do this: absorb the energy, deflect it, or just transfer it to sky miles (flying across the arena). Deflecting the energy is usually the best option. This can be done with a hard armor (like steel) set at a shallow angle as shown in Figure 1. Other robots will just hit it and glance off. The second best method is to absorb the energy. This can be done in a variety of ways; thick aluminum will permanently deform and absorb energy, and rubber will deform and return to its original shape. A good example of the absorption method is seen in Figure 2.

FIGURE 6. The completed switch is ready to mount.

The third method — which I don't recommend — is to transfer the energy into motion. This involves making your robot as solid as you can, so that all impacts yield no damage and your robot is simply thrown across the arena. The problem with this is that your speed controllers, motors, receivers, etc., will then feel the maximum amount of shock. It is hard to build such a strong robot and properly shock-mount your components so you would be better off considering the first two methods.

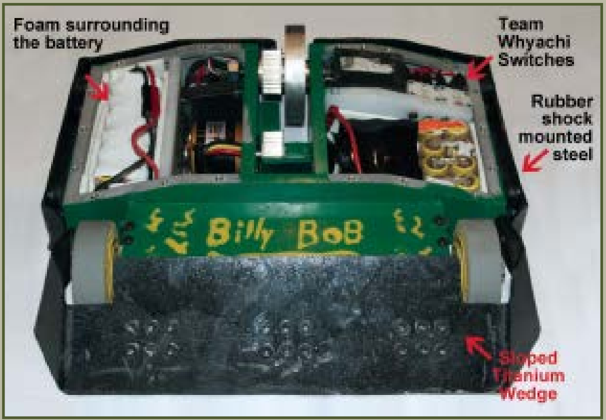
All the Small Things

The worst way to lose is because of a minor issue that could have been avoided with a 30 second fix. This usually translates to problems with the electrical system. Wires make up most of the electrical system and the insulation on the wires can be sliced by sharp frame members or moving parts. If you have a metal frame, multiple instances of this can cause a short. To prevent this, use wiring with good insulation for both rated temperature and durability. Wrapping the wires in a protective sleeve is the best option and might prevent failures like the one shown in Figure 3.

The next failure point is the wire connections. All connections should be crimped and soldered. All connectors that can be disconnected need to be taped or zip-tied so they cannot be disconnected accidentally during a match. Every joint between a wire and motor or other device should be built up with hot glue, Goop, or something similar. This will provide support for the wires when they get pulled on.

This brings us to the main components portion of the electrical system. The receiver is one of the smallest components, but it is the heart of the robot. When it breaks, everything else stops working. The receiver crystal is prone to popping out; solve this with a strip of electri-

FIGURE 7. Billy Bob, a 30-lber, has shock-mounted armor, a hinged titanium wedge, shock-mounted batteries, and electrical components. Billy Bob competed at the North East Robotics Club's Motorama competition going undefeated and winning first place Why? Because it didn't break.



cal tape around the entire receiver to prevent it from coming out. The PWM cables need to be glued into the receiver as described before. You also need to shock-mount the receiver, speed controllers, batteries, and anything else that else that might break (foam works well). If you properly glue, zip-tie, tape, and foam all of your critical components, you will drastically limit your failure points.

The last small thing to consider is power switches. Most builders have lost at some point due to a power switch accidentally turning off during a match. I feel the best option for robots 12 lbs and higher are the Team Whyachi power switch options. (see **www.teamwhyachi.com**). Custom solutions do work (Figure 5 and 6), but off-the-shelf products generally are not suited to this use.

Conclusion

As you can see, there are many factors to consider when designing and engineering a robot. For an example of a robot that has taken into account all of this, see Figure 7. Remember, don't be your own enemy. Be tolerant, control the damage, and remember all the small things. But above all, have fun and be safe! SV

EVENTS

RESULTS — February 12 - March 11

Motorama 2007 was held February 16 in Harrisburg, PA. Presented by North East Robotics Club. Results are as follows:

- *Fairies (150 g)* — 1st: "Deimos," Team Cosmos (ranked #4); 2nd: "Mr. Bigglesworth," Team Udanis; 3rd: "Steve-O," Team PITA.
- *Ants (1 lb)* — 1st: "Switchblade," Team Sawzall (ranked #2); 2nd: "Fender Bender," Team JandA; 3rd: "Absolutely Naut VDD," Team Anarchy Robotics.
- *Beetles (3 lb)* — 1st: "Can O Nuns," HockeyRunner Robotics; 2nd: "Enemy," (ranked #7), NovaRobots; 3rd: "SSOD," jZRobotics.
- *Hobbys (12 lb)* — 1st: "Solaris," Team Cosmos; 2nd: "Darkblade," Team Sawzall (ranked #7); 3rd: "Rants Pants," Not-so-boring Robotics (ranked #1).
- *Open Feathers (30 lb)* — 1st: "Billy Bob," Robotic Hobbies; 2nd: "Sloth," Team Massacre Robotics; 3rd: "Tripolar," Team Brain Damage.
- *Sportsman 30s* — 1st: "Bounty Hunter," Team Hammertime; 2nd: "PITR," Team Javman; 3rd: "Helios Sport," Team Cosmos.
- *Ant Rumble* — "yelo," Team Pinq.
- *Beetle Rumble* — "Destructive Crab," Green Machines.
- *Hobby Rumble* — "Pummel," Robotic Hobbies.
- *Feather Rumble* — "Gnome Portal," Robotic Hobbies.
- *Most Destructive Robot* — Sloth, Massacre Robotics.
- *Coolest Robot* — Diabolical Machine, Team Terror.
- *Best Driver* — Jon Durand, Team Anarchy Robotics.

The Smackdown in Sactown III was held February 18 in Sacramento, CA. Presented by Sacbots. Results are as follows:

- *Fairyweights (150 g)* — 1st: "Crisp," clamp, Team Misfit; 2nd: "Micro Drive," lifter, Team Misfit; 3rd: "Atom Bomb," drum, Team Misfit.
- *Antweights (1 lb)* — 1st: "MC Pee Pants," drum, Team Fatcats; 2nd: "Dark Pounder," drum, Dark Forces; 3rd: "Unblinking Eye," horizontal spinner, Hammer Bros.

Bay Area Robot Fights 2007 was held March 3 in Tampa, FL. Presented by TeamPyramid. Results for this event were unavailable at press time. SV

TECHNICAL KNOWLEDGE

CHAINS — Putting the Growl in Your Drive Train

● by Steven Kirk Nelson

Regardless of the motor or engine used in powering a combat robot, you have to get the power to the output shafts. There are several ways to do this using gears, belts, friction drives, or just good old chain.

Each method has a different level of efficiency, so it's up to the builder and the application. Chains have certain advantages over belts and gears. One of the most obvious is chain can be cut to length rather easily and when properly installed, does not slip. Another advantage is it is possible to change shaft ratios by changing the sprocket sizes. This lets you adjust the final torque output of the drivetrain to fit your power output needs.

The three commonly-used chain sizes in larger combat robots are #40 roller, #35 non-roller, and #35 high tensile extended bushing or "space chain," such as EK Spaced Silver Pro Chain. Chain size is determined by how much load that it will have to handle.

In all reality, it has been proven that for non-weapon drivetrain use, standard #35 non-roller chain works really well up to about three horsepower; for 4-8 HP, use #35 EK Spaced Silver Pro GoKart racing chain; and for anything over that or a weapon drive, use #40 roller chain. I should mention that we have used the #35 GoKart chain with over 13 HP and at 10,000 RPM with great success. But your mileage and service life may vary.

Fitting Chain

For a properly designed machine, you have to do the math and calculate the shaft centers, loads applied, and type of lubrication used. If you want 15,000 hours of run time with your chain, then you must do the math! A good site is **www.drivesinc.com/roller-PDFs/rollr-precsn.pdf** But there is a mechanical way to build a drive train, as well. Basically, you let the chain tell you what it will handle and where the motors and shaft can be mounted. When building a drive train in the garage, I always start with some wood blocks, bearings, shafts, sprockets, motors, and a box of chain. I just keep sliding the parts around and observe the amount of wrap on the small sprocket. If you can make the chain happy in this way, you can just drill the holes and bolt the bearings and shafts to a frame and your drivetrain is done. (Note: Always insert the key stock in the sprockets and shafts before drilling the mounting holes as they do change chain tension a bit.) You have to always maintain at least a 120 degree (33 percent) wrap on the smaller sprocket. And it is usually not recommended to go over a 7-to-1 reduction in a single stage. For more reduction, use multiple stages through intermediate (jack) shafts.

When fitting a chain, you first mark the links to be removed (soapstone works well). Then, using a chain breaker tool, you push the pins out on each one a little at a time until the side bars come loose. Don't try to fully push a pin all the way out in one operation. Alternate between each pin; this reduces the damage to the chain and the chain breaker tool. Personally, I grind the pins down flush on the link I want to remove, to further reduce the stress on the chain and the tool.

When installing the master link (if used), the closed side of the master link should be facing the direction of rotation for maximum strength and reliability. If the output shaft will turn both directions, then mount the closed side of the link toward the more common direction of use, for example, the forward motion in a pushy bot. (You're not supposed to run *away* from your opponent.)

It is very important to get your sprockets aligned as straight as possible to keep your chain happy. You should use a straight edge to align the sprocket faces to each other. This takes a bit of time and practice. Basically, you want the faces of the sprockets to be true and square, so the chain will ride on the center of the sprockets and not rub on the side bars in operation. Once you get everything straight, you want to install the chain and rotate the drivetrain by hand while checking for binding and crunchy sounds.

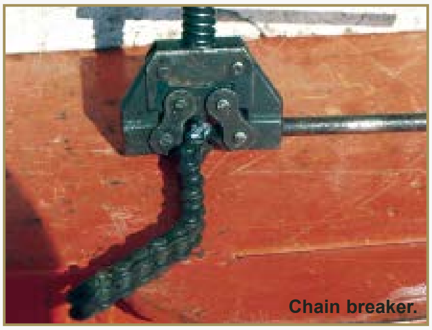
Another thing to double check is the chain tension; #35 chain should have about 3/8" up/down deflection between the shaft centers, and #40 chain should have about 1/2" deflection. Running a chain too tightly wastes horsepower and destroys the chain, bearings, sprockets, and motors. It makes a lot of noise, as well.

Sprockets and chain, oh my!

I am not a believer in the need for added chain tensioners or rubbing blocks. So I simply make my mounts adjustable by drilling multiple holes in the frame and mounting my bearings in the center hole on initial assembly. Once it's all running smoothly, I cut out the slots to allow for chain adjustment in two directions from the original install. Also, it makes it possible to adjust the alignment when needed. Chain tension will change after a few hours (or even minutes) of run time.

Lubrication

Steel chains require lubrication. They are not designed to run dry. I should note that most chains come with a protective lubricant applied to them at the factory. This lubricant is applied to keep them from rusting in the box. To keep your chain working properly, you need to lubricate them with a good grease or oil. With brand new chain, I usually run them for a few minutes with the factory lube. Before I do any real testing, I clean



Chain breaker.

The master link.

Sprockets need to be aligned as straight as possible.

them thoroughly, then use a mild automotive solvent or even kerosene. Do not use carburetor cleaner or gasoline! I just use a soft parts brush and a clean bucket filled with clean solvent.

Adjustable mounts.

After that, I remove the solvent with hot water mixed with a little liquid dish soap and then blow dry them with an air nozzle. Immediately after the chain is blown dry, you must apply a good chain lube to prevent rusting and galling under load. I've had good luck with PJ1 chain lube. Of course, there are lots of brands of both wet and wax type lubes for motorcycles and GoKarts out there.

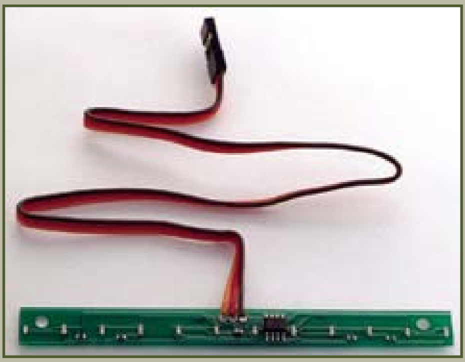
Summing It All Up

- Chains are very strong and do not slip.
- #35 non-roller or #35 HT GoKart chain will handle most drivetrains.
- Maintain at least 120 degrees of chain wrap on your smaller sprocket.
- Single reduction ratios over 7:1 are unreliable. Use a multiple reductions for higher final reduction ratios.
- Lay out your drive chains, sprockets, shafts, and bearings before building the framework.
- Using adjustable mounts is a good idea.
- Align your sprocket faces using a straight edge.
- Thoroughly clean new chains and then lubricate them with a proper chain lube.
- Never run chains dry! SV

PRODUCT REVIEW — LED Lighting Systems

● by Chad New

Rule 6.6 of the Robot Fighting League states that all robots must have a light that is easily visible from the outside of the robot which shows when main power is activated. The power status light is one aspect of most robots that is often overlooked while the robot is being built. Most builders choose to use the status light on their speed controls or run a tape covered monstrosity of an LED from a random area of wiring from within the robot. However, LEDs can add a cool finishing touch to your robot, giving it personality and something to distinguish it from other bots while still satisfying a rule for nearly all robotic events.



At times, LEDs can tend to be bland, with nothing special about them other than the different colors; perhaps they will blink at a given interval or change colors. But even with that, I believe it to still be a little too plain. Dimension Engineering has created a line of products that can solve all of your LED troubles! They offer three different products that not only emit light, but do it in a unique way which is extremely easy to install. The three different systems are: the Easy Light system — which uses a pair of ultra bright LEDs; the Fire Fly lights use two LEDs that glow on and off depending on their setting; and the Sidewinder Light system which harkens back to the '80s TV show "Knight Rider" and the light system that was embedded in its hood. It uses several LEDs that scroll back and forth while also fading in and out.

All three of these LED systems offered are extremely easy to install and use; they come with a standard hobby receiver plug which you can plug into any channel that you are not using to power the unit. You can also attach them to any 5V power source if you do not have an extra channel or are not running a receiver.

The lights also double as a radio signal indicator if you have them plugged into the receiver. If you lose signal, the lights will turn off or glow continuously alerting you to a problem. However, this can be disabled by cutting the signal wire should you not want this feature.

Overall, the lights are very easy to mount. They come with mounting holes so you could secure them with bolts, but I have found that a dab of Shoo Goo provides adequate support for the units. During many events I have yet to have any of the light systems fail, showing their toughness and durability.

So, if you are looking to give your robot a unique touch, give the Dimension Engineering's LED systems a look at **www.dimensionengineering.com**. SV

EVENTS

UPCOMING — May and June

Carolina Combat — This event will take place on 5/4/2007 through 5/5/2007 in Greensboro, NC. It's presented by Carolina Combat Robots. Robots from 150 g Fairyweight to the 120 lb Middleweights will be competing. The big bot arena is a 16 x 32 steel structure with 1/4" steel floor and 1/2" Lexan for the walls. Go to **www.carolinacombat.com** for more information.

HORD Spring 2007 — This event will take place on 5/19/2007 in Strongsville, OH. It's presented by the Ohio Robot Club. The Ohio Robotics Club will be holding its fourth House of Robotic Destruction event at the Strongsville HobbyTown USA, just outside of Cleveland, OH. The ORC insect arena is 4 x 8 in size; halfway through a match two 14" x 14" pits open. Go to **www.ohiorobotclub.com** for more information.

ComBots Cup 2007/Maker Faire — This event will take place on 5/19/2007 through 5/20/2007 in San Mateo, CA. It's presented by ComBots. Antweights through SuperHeavyweights, with a $10,000 Heavyweight prize and a $3,000 MiddleWeight prize! Go to **www.robogames.net** for more information.

Mechwar 10 — This event will take place on 5/19/2007 through 5/20/2007 in Eagan, MN. It's presented by Mechwars Robot Combat. Antweights through their unique Mega 390 lb class. Unique revenue sharing format plus prizes. Go to **www.tcmechwars.com** for more information.

ROBOlympics/RoboGames 2007 — This event will take place on 6/15/2007 through 6/17/2007 in San Francisco, CA. It's presented by ComBots. Combat classes fairyweights through superheavies, plus dozens of non-combat classes. Go to **www.robogames.net** for more information. SV

HAPTICS, Telepresence, *and* TELEROBOTICS

by Bryan Bergeron

RESPONDING TO AN EMERGENCY LOCATOR BEACON, AN AUTONOMOUS TELEMEDIC ROBOT HOMES IN ON A SOLDIER WOUNDED IN BATTLE ...

A human medic, safe in a military base on another continent, takes command of the robot and drags the unconscious soldier out of harm's way. Then, using a video, audio, and touch telepresence interface, he directs the robot's arm to gently manipulate the soldier's bleeding leg. Through visual inspection and carefully prodding and bending the leg, the medic operator determines that the damage is limited to a deep laceration. Using the robot arm as an extension of his own, the medic applies a tourniquet, using just enough pressure to stop the bleeding without damaging the underlying tissue. With the soldier stabilized, the medic returns the robot to autonomous operation so that it can rejoin the swarm of robots combing the active battlefield for wounded.

While this scenario is fictitious, the technology isn't. See the videos of prototype robots in operation on the Army's Telemedicine and Advanced Technology Research Center website **www.TATRC.org**) for a view of the current state of the art in autonomous casualty care robotics. Although many technologies are involved in telemedicine, it is telepresence — the ability to see, hear, speak, touch, and *feel* at a distance — that places the medic virtually at the soldier's side. In particular, it is the ability to feel at a distance through a haptic interface that enables the medic to physically assess the soldier's wound and correctly apply the tourniquet.

This article introduces the features and design challenges associated with haptics by way of a telerobotic gripper that you can add to your robot arm or mobile robot platform.

Force Feedback and Haptics

To appreciate the advantages of a haptic interface over a traditional interface limited to audio and/or video feedback, consider the features of the toy gripper shown in Figure 1. The device serves as an extension of the user's arm, effectively increasing reach by an additional foot or more. Compared to operating a robot arm with a keyboard or joystick, manipulating an object with the gripper is second nature. Release the grip and the jaws open. Squeeze the grip and the jaws close. More importantly, when the jaws stop moving — whether because of an object in the jaws or because the jaws are completely closed — so does the grip.

The toy gripper does add some complexity over direct manipulation. For example, there is slight tension in the grip because of the spring that keeps the jaws open, and pulling back the grip only a few millimeters causes a proportionally larger change in jaw opening. Despite these nuances, the interface is both intuitive and easy to learn. An inquisitive five-year-old can master the toy in seconds.

One reason the toy gripper is so easy to learn is the force feedback provided by the direct physical linkage between the grip and jaws. Pick up a Nerf ball with the toy and the grip feels squishy. Pick

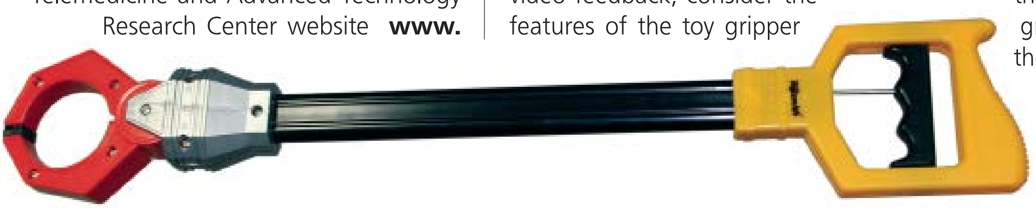


FIGURE 1. A toy gripper arm that provides force feedback to the operator through a direct mechanical linkage.

up a hard plastic ball and the sliding grip mechanism doesn't yield to additional pressure. With a few minutes of practice, you can probably learn to use the toy to discriminate between a range of rigid and soft objects by feel alone.

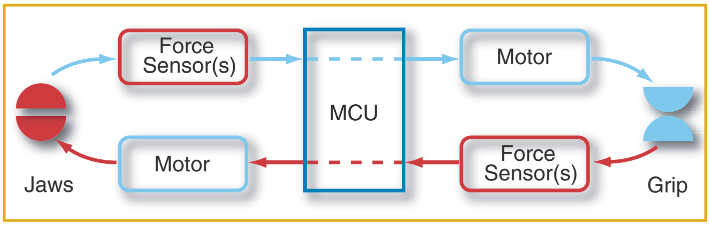
Imagine that we take the toy gripper, physically separate the grip from the jaws, and yet somehow maintain the force feedback. By replacing direct force feedback with synthetic force feedback, we have the haptic component of a telerobotic system. Although conceptually trivial, faithfully replicating the force feedback supplied by the mechanical linkage in the toy gripper can be challenging.

Figure 2 shows a high-level schematic of the key components of our new telerobotic gripper. Note the jaw and grip circuits are mirror images of each other. There are force sensors and effectors on either end of the loop, with a microcontroller unit (MCU) arbitrator in the middle. The MCU defines the properties of the feedback loop, providing filtering, amplification, attenuation, scaling, or range constraints for signals to the motors and from the force sensors. Although not shown in the figure, there may be bidirectional communications between the motors and MCU for position sense.

The MCU enables telerobotics to be applied to complex operations such as surgery or munitions handling. Consider the benefit of having the ability to scale down human input motions for microsurgical procedures, or of filtering small, sudden movements (i.e., tremors) to improve the precision of incisions. Amplified force feedback can enable a surgeon operator to detect small differences in tissue elasticity and in the tension on a suture while tying a knot. Furthermore, the ability to programmatically constrain the range of motion of remotely operated surgical instruments can minimize risk of injury through accidental operator movement.

Developing telerobotic surgical stations capable of enabling a surgeon to operate meters or miles away from the patient typically involve multi-million dollar investments. However, you can experiment with basic telerobotic concepts by building and working with the gripper described here.

FIGURE 2. Haptic gripper key components. Motor position sensors not shown for clarity.



Telerobotic Gripper

Creating a telerobotic gripper involves building two complete gripper circuits: one to provide the operator with force feedback and one to manipulate the jaws. The grip and jaw circuits both generate force and respond to stimuli as defined by the MCU.

Jaw Assembly

If you own a robot arm or gripper, then you're over half-way to a telerobotic gripper. However, if you're starting from scratch, then an inexpensive but workable gripper is the Big Gripper from Budget Robotics. I purchased the basic kit without servo and added an Airtronics 94358z, which provides more than the recommended torque, and a heavy-duty servo coupler.

This combination — shown in Figure 3 — enables me to work with the amplified power translation with enough jaw force to crush empty beverage cans. I mounted the gripper on a graphic tube using a Lynxmotion HUB-09 tubing connector. Graphite tubing, available from Kite Studio, is a strong, lightweight alternative to aluminum tubing for this and other robotics applications.

The next step is to add force sensing to the jaw assembly. One option is to use a pair of FlexiForce pressure sensors, which are accurate and provide good dynamic range, but are expensive. An economical alternative is to make your own sensors from the conductive foam used to protect semiconductors from ESD damage.

Start with five 15 mm x10 mm rectangles of foam (four for the jaws and one for the grip), four two-foot lengths of solid 28 gauge insulated hookup wire, and two 10" lengths of stranded 28 gauge wire. Strip 10 mm from one end of each wire and insert a pair of wires lengthwise into each of three foam rectangles, as shown in Figure 4. The two force sensors with solid wire will be used on the jaw assembly, along with two unwired pieces of foam to provide symmetry.

When inserting wires in the conductive foam, try to keep the wires parallel and about 3 mm apart. Use an ohmmeter to verify the wires inside the foam aren't touching and add a drop of thin CA cement (Super Glue) where the wires enter the foam to keep them in place. Nominal resistance should be about 5K ohms uncompressed to about 2K ohms when compressed to half of the foam's original thickness.

Glue one pad to the tip of one jaw and one pad to the middle of the same jaw using silicone glue. Because the jaws are hollowed out, you'll have to fill the center grove with glue so that the conductive foam compresses properly. Twist each pair of sensor wires together and secure them to the jaw assembly with tape or tie-wraps. Glue the two pieces of unwired foam to the other jaw at the same level (refer to Figure 3).

Operator Grip Assembly

A standard servo can be used to

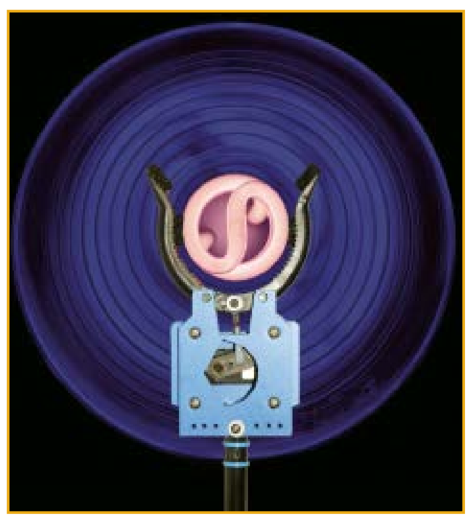


FIGURE 3. Telerobotic gripper jaw assembly grasping a fluorescent bulb.

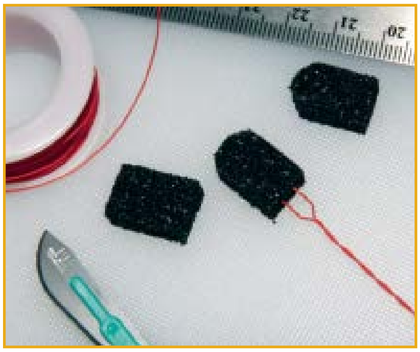


FIGURE 4. Construction of the conductive foam rubber force sensors.

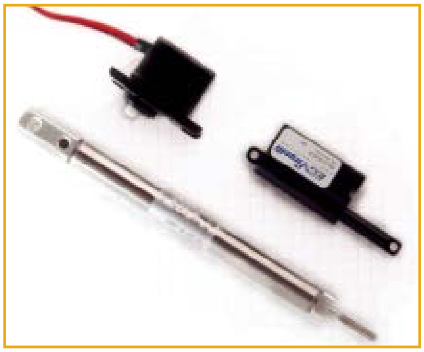


FIGURE 5. Firgelli PQ-12f linear actuator, GWS Naro Pro/Std servo, and SMC NCJ2D10-200S pneumatic linear actuator.

provide operator-side force feedback. However, I opted to use a Firgelli miniature linear actuator because it seemed like a good application to test the capabilities of the new actuator. The diminutive Firgelli PQ-12f linear actuator provides a force of 18N at 6 mm/s over a stroke of 20 mm when driven by 5 VDC at 250 mA. Position information is available from the PQ-12f through a 2K ohm linear potentiometer tied to the stroke position.

The 19 g linear actuator is comparable to the GWS Naro Pro/Std servo in size and weight and yet provides the same stroke as the much larger SMC NCJ2D10-200S pneumatic linear actuator. Although the pneumatic actuator provides 54 N of force, it weighs 50 g and doesn't provide the positioning capabilities of either the Naro or PQ-12f. Figure 5 shows the three actuators without their associated control circuitry or supply lines. You can probably envision a compact crawler robot made with the PQ-12f.

The linear format simplifies the design of the grip and saves space and weight. However, the PQ-12f requires a motor controller, such as the three-amp SMC03A motor controller with feedback from Pololu. The controller's analog voltage feedback mode proved a good match for the PQ-12f. Use the 2K potentiometer as a voltage divider and feed the wiper voltage to the feedback terminal on the motor controller. With the feedback jumper set to analog feedback mode, the speed commands listed in the manual act as position commands.

The handle from the toy gripper provides a reasonable hardware platform for the telerobotic gripper. Disassemble the plastic handle and attach a 3" x 4" PCB (printed circuit board) to one side of the handle. Mount the linear actuator and motor controller on the board. You can either use the odd five-pin 1 mm pitch connector supplied with the linear actuator or solder directly to the pads on the actuator's flex PCB cable.

Position the actuator so that the 20 mm stroke matches the normal grip range of motion. If you replace the existing metal connecting rod with an adjustable servo link, you'll have more flexibility in positioning the actuator. Wire the motor controller and linear actuator and add a screw terminal block for the serial communications and sensor lines to the MCU. Glue the remaining foam sensor — the one made with stranded wire — to the squeeze mechanism grip so that it will make contact with a finger as you squeeze the grip. Figure 6 shows the grip assembly with the linear actuator mounted on the toy grip handle.

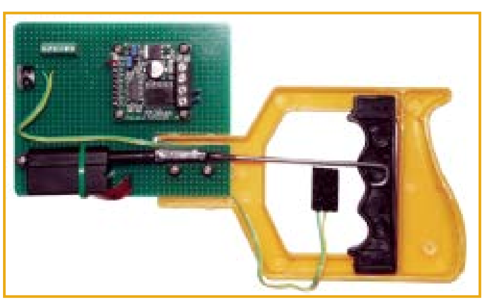


FIGURE 6. Exploded view of the operator grip assembly.

Microcontroller

Almost any microcontroller with analog inputs and a serial port to communicate with the motor controller will do. I used an ATMEL 128-based Mavric IIB from BDMicro. The board, shown with the complete telerobotic gripper circuit in Figure 7, provides terminal block access to 51 digital I/O and eight analog input pins. When combined with the BASCOM-AVR compiler, the Mavric IIB provides a rapid prototyping environment with ample processing power.

Connect the jaw assembly servo leads to the servo block on the Mavric IIB, using a servo extension cable. Calibrate the servo control code so that the servo stops before the jaws reach their mechanical range limits so that the servo isn't damaged. Using 5K potentiometers in series with each force sensor, create voltage dividers with the supply voltage. Feed the variable voltage leads to two analog input ports on the microcontroller board.

With the motor controller configured and the jaw and grip assemblies complete, the real fun begins — programming the microcontroller signal manipulation routines. Controlling the SMC03A with the Mavric IIB is straightforward. The motor controller is configured by sending four-byte commands to the serial port using the SEROUT command with BASCOM-AVR. The pseudocode for the microcontroller code is:

```
DO
IF Grip_Force = 0 THEN
     Open_Jaws
ESSE
     Close_Jaws
LOOP

Open_Jaws
     Jaw_Servo_Position = Jaw_Servo_Open_Limit
     Grip_Position = Grip_Open_Limit
Return

Close_Jaws
     FOR X = Jaw_Servo_Position to Jaw_Servo_Close_Limit
            IF Jaw_Tip_Force > Tip_Limit THEN Exit_Early
            IF Jaw_Mid_Force > Mid_Limit THEN Exit_Early
            IF Grip_Force < Grip_Limit THEN Exit_Early
            Jaw_Servo_Position = X + K1
            Grip_Position = X + K2
     NEXT X
     Early_Exit:
Return
```

The main loop continually checks the resistance of the grip force sensor. When there is no force on the sensor, the jaws open to their maximum position and the grip slides forward in the handle. When force is detected by the grip sensor, the jaws attempt to close from their current position. However, the position of the jaw servo and grip linear actuator aren't updated if the force levels detected by the jaw sensors is too great, or the pressure on the grip sensor is insufficient. Otherwise, the jaw servo position is incremented by a factor *K1* and the grip is advanced by a factor *K2*, where K1 and K2 can take on simple values or represent complex functions.

By adjusting *Tip_Limit, Mid_Limit*, *Grip_Limit*, *K1*, and *K2*, you can create a variety of force mapping functions for your microcontroller to provide varying degrees of physical feedback for a given resistance change in a jaw sensor. *K1* and *K2* vary the relative movement of the jaws and grip with each update. By dynamically varying *Grip_Limit*, you can create a stair step response that enables you to increase the force on the object in the jaws, up to the jaw force limits.

Operation

Squeeze the grip lightly and the jaws should begin to close from their open, resting position. The jaws should continue to close until one of the force sensors on the jaw is compressed by an object, the preset limit of servo travel has been reached, or you let up on your grip. When the jaws stop moving, so does the grip. In addition, if you squeeze the grip harder, the jaws will attempt to close more until the next force threshold on the jaw sensors is reached. If the jaw angle changes because the object is compressible, the grip should move, as well.

If you have a video camera, extend the grip-microcontroller cable so that you can place the jaw assembly in an adjacent room. You'll find that haptic feedback takes on added significance without direct visual feedback.

The maximum speed of the linear actuator limits the rate at which you can draw in the grip as you squeeze. The PQ-12f requires about 1.8 seconds for a 20 mm stroke. If you try to force the PQ-12f or squeeze the grip harder after the linear actuator has stopped, the jaws should continue to close, up to the range limit or the limit controlled by the jaw pressure sensor.

With force amplification, I can sense and pick up an empty aluminum beverage can without damaging it and, with added pressure on the grip, crush the can. To avoid stripping an expensive servo on a closed can or solid object, consider adding a current sensing circuit in the jaw servo supply line in case the object doesn't happen to align with one of the force sensors.

You'll notice a short lag between the time you apply pressure to the grip and when the jaws begin to close. Minimizing this delay is one of the most important challenges in telerobotics, especially in time-critical applications such as surgery. Try minimizing the lag in your system by using a higher performance microcontroller, adjusting sensor set points, and removing slack from the physical linkages.

Psychophysics

As you experiment with various mapping functions, you'll soon discover that mathematically nonlinear mappings feel linear after a minute, and that squeezing the grip twice as hard doesn't double the force applied to the grip sensor. This is because our organic sense organs and effectors are both nonlinear and time varying.

If you've spent any time in the gym, you've probably experienced the nonlinearity of muscular strength first-hand. The pressure that you can exert, say, curling a dumbbell, varies through the movement because of changes in the mechanical advantage of the elbow joint, elasticity and tone of the biceps and opposing triceps muscles, as well as short- and long-term neuromuscular adaptation. Fatigue, markedly decreased ambient temperature, drugs, sleep deprivation, and injury all affect the capacity for muscular work. Muscle efficiency drops with fatigue, extremes in ambient temperature, sleep deprivation, and injury.

Similarly, the human nervous system automatically adjusts the sensitivity of our sensors as a function of ambient noise and the nature of the signals. Our response to weight, sound, light, and many other stimuli can be approximated by the Weber-Fechner law or model, which is useful in identifying the *just noticeable difference* in a stimulus. The law states that the just noticeable difference of a stimulus is proportionally related to the magnitude of the stimulus [1]:

$$p = k \log s + C$$

where *p* is a measure of the perception

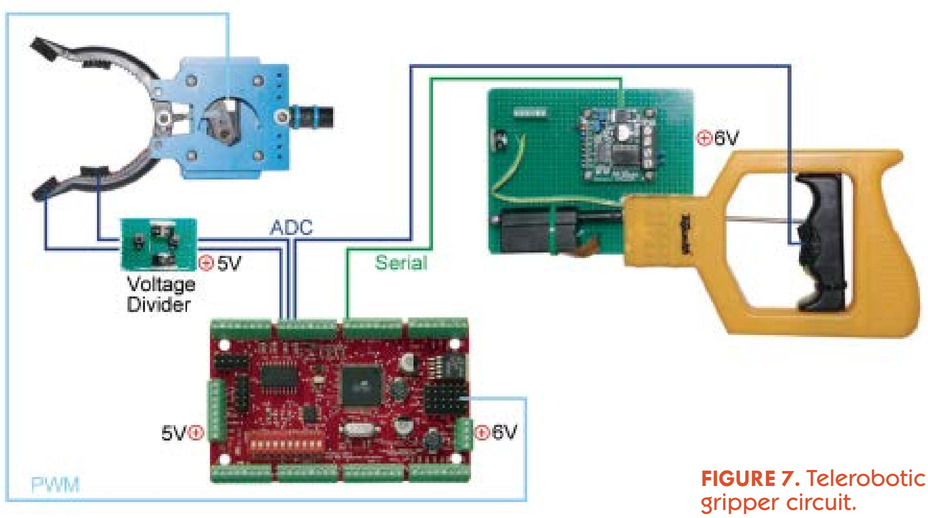


FIGURE 7. Telerobotic gripper circuit.

RESOURCES

Big Gripper
Budget Robotics
www.budgetrobotics.com

SMC Pneumatics
Available as individual components through Allied Electronics and Control Sources, Inc., or in complete kit form from IFI Robotics
www.alliedelec.com
www.controlresourcesinc.com
www.ifirobotics.com

PikStik Reacher
www.pikstik.com

FlexiForce Sensors
Available from Parallax
www.parallax.com

BASCOM-AVR
MCS Electronics
www.mselec.com

Pololu SMC03A Motor Controller
www.pololu.com

Graphite Tubing
Available from Kite Studio
www.kitebuilder.com

PQ-12f Miniature Linear Actuator
Firgelli Technologies, Inc.
www.Firgelli.com

of the stimulus, k is sense and level-dependent, s is the magnitude of the stimulus, and C is the constant of integration.

The Weber-Fechner law quantifies the common finding that humans are much better at sensing relative difference than absolutes. For this reason, photographers rely on light meters, sound technicians on sound level meters, and grocery shoppers on scales. If you're an audiophile, you know that doubling the power of your audio amp won't double the perceived volume (loudness) of your sound system. The relationship between sound power (stimulus strength) and loudness is logarithmic, and commonly described in terms of decibels.

If you have access to a force gauge, experiment with various force mappings. You'll find the force exerted by the Big Gripper is nonlinear, even with a linear control signal because of the varying mechanical advantage of the jaw mechanism and the variation in servo torque as the servo-jaw linkage moves through its full range of motion.

Variations

An inexpensive adult-sized gripper, sold as a reacher, provides a much better platform than the toy gripper, especially for operators with large hands. For example, the aluminum Pikstik Pro reacher has an ergonomic trigger grip and the jaws are good enough to be repurposed for a robot jaw.

The capabilities of the single-axis telerobotic gripper can be extended by adding additional axes and by replacing the grip-microcontroller wiring with a Bluetooth or WiFi connection. Try mounting the gripper and a wireless video camera on a mobile robot base and interacting with objects out of your direct field-of-vision. You can substitute a standard servo and servo saver for the linear actuator or use a pneumatic cylinder, such as the SMC NCJ2D10-200S or a less expensive LEGO Technics cylinder.

The NCJ2D10-200S is particularly well suited for the grip actuator because it has a built-in spring that returns the cylinder to the retracted position at rest. Connect a solenoid air valve to the air input port of the cylinder and activate the solenoid when the jaw sensors indicate contact. The partial vacuum will increase the travel resistance of the grip. The feeling is less crisp than that provided by the PQ-12f, but there is no limit on the squeeze rate or pressure, and no chance of stripping servo gears.

Consider using the remaining analog inputs on the Mavric IIB to read potentiometers to interactively set the amplification, tremor filter, and motion limits. If you have a CMUCam II, use it to identify skin tones and limit the closure force of the jaws, regardless of the pressure exerted on the grip sensor. When you're ready to build one of those autonomous telemedic robots, the TATRC site (**www.tatrc.org**) and the IEEE article by Rosen and Hannaford [2] are good places to start for more information. **SV**

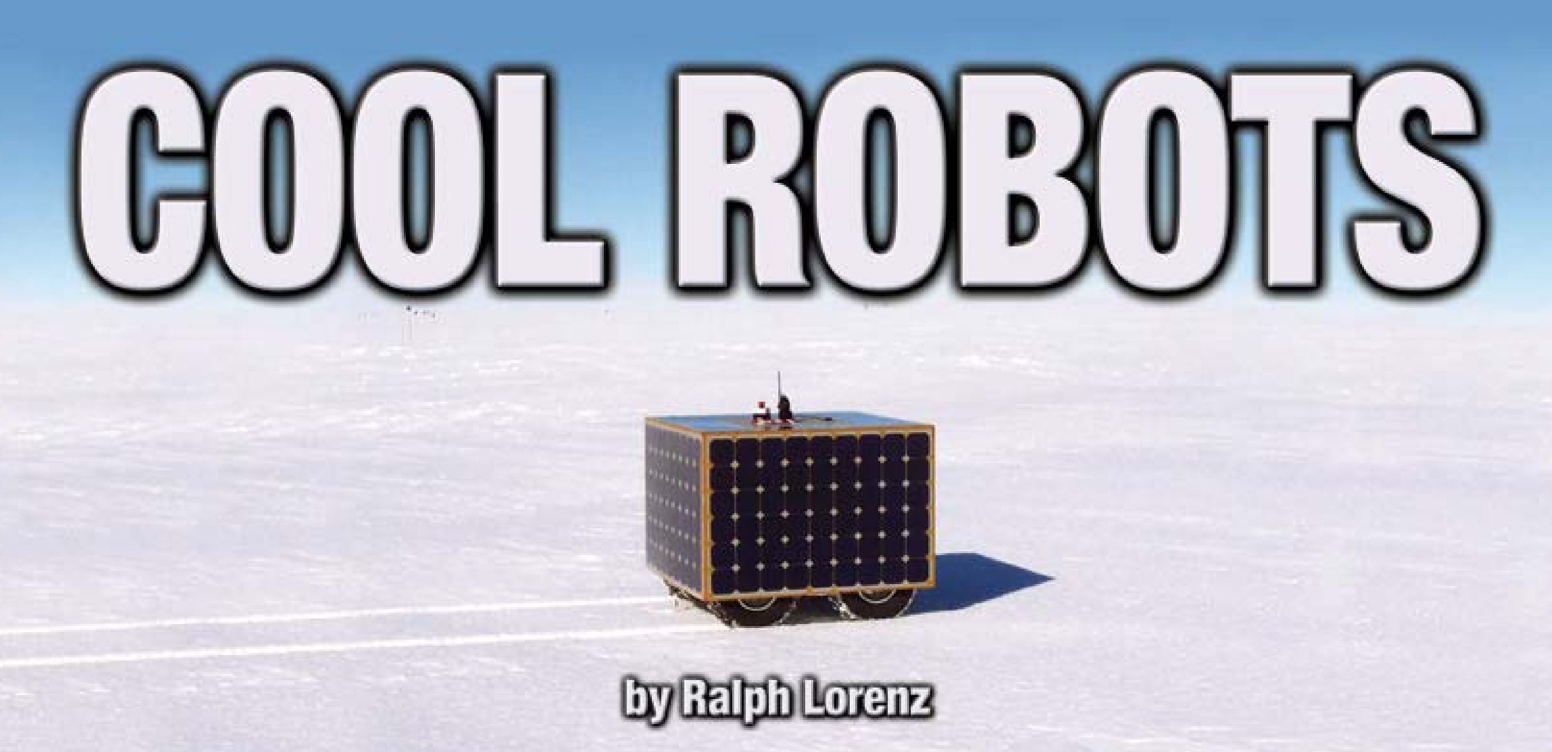
REFERENCES

[1] Dudley, B., *Basic Phenomena of Electronics, in Electrical Engineering Handbook*, D. Fink and D. Christiansen, Editors. 1986, McGraw-Hill: New York. p. 1-58.

[2] Rosen, J. and B. Hannaford, *Doc at a Distance*. IEEE Spectrum, 2006. 43(10): p. 34-9.

COOL ROBOTS

by Ralph Lorenz



Dull, Dirty, and Dangerous ...

These are the missions that are ideal for robots. Researchers at the US Army Cold Regions Research Lab and Dartmouth College are breaking new ground in developing a robot 'Husky,' to carry scientific instruments across polar ice sheets. Their 'cool robot' looks like a blue cube and can carry or drag equipment, or even a person, across the snow.

Although small and simple compared with the autonomous vehicles driving in the DARPA Grand Challenge, the Cool Robot tackles a broadly similar problem; 70-85% of the Antarctic and Arctic research budget pays for logistics. There can be huge savings, and great reduction in risk to personnel, if the relatively simple task of crossing large expanses of empty terrain to deploy equipment can be done with a simple solar-powered robot.

As well as acting as a delivery truck carrying or dragging supplies, the robot could be used to deploy networks of weather or seismic sensors, dropping packages off every few kilometers. Another application might be to conduct a geophysical traverse, dragging an ice-penetrating radar across the polar cap to measure the rock topography hidden underneath.

The concept that resulted looks more like a delivery truck than a sports car, comments Alex Streeter, a member of the team. It is a four-wheel drive solar powered box-shaped robot, measureing 1.2 x 1.2 x 1.0 m. It has a lightweight honeycomb structure with solar panels and a set of brushless DC motors with big tires. Away from the edges of the ice sheet, the terrain is almost featureless and flat, apart from a ribbed surface called sastrugi that wind action forms in the snow.

Obviously near the poles, a magnetic compass is not much use. The magnetic field lines dip very steeply towards the ground at high latitude, meaning only a small component of the field is useful for measuring direction. Also, since the magnetic pole is displaced from the geographic pole, the difference between true north and magnetic north becomes large and very sensitive to exact location.

The Cool Robot instead uses GPS guidance, comparing the reception times of precise signals from the set of Global Positioning System satellites some 20,000 km up. Because of the flat terrain, the signals are rarely blocked. The robot is programmed with a set of waypoints and simply navigates from one to the other. For tests, a short-range radio is used for control, although an Iridium satellite modem is used for long-distance communications.

Two factors are key to the success of the robot. First is power. It is expensive to deliver fuel to remote polar bases, and the environmental pollution associated with spills is a concern. Since most polar operations are carried out in summer, solar

A new winter sport? The Cool Robot towing more than its own weight. Note laptop for remote control via wireless link.

ADDITIONAL INFORMATION

More pictures and details on the Cool Robots project are available at: **http://engineering.dartmouth.edu/crobots/**

The paper available there — *Performance of a Solar-Powered Robot for Polar Instrument Networks* — gives lots of useful detail on the peak power tracker and on traction estimation.

What lies beneath. The robot chassis containing the motors, batteries, and control system; the solar cells are mounted on lightweight panels that are simply hung on the main structure.

power is a feasible option, and has the advantage that once the robot is delivered, it can keep operating year after year.

In polar regions, the sun will be always low in the sky, if it is above the horizon at all. This means that roof-mounted solar panels alone are not much use. In fact, in polar midsummer, the sun will be above the horizon in all directions — south at noon and due north for the 'midnight sun.' So the Cool Robot is equipped with solar panels on its four walls, as well as one on its roof.

Although it is bright, it is, of course, still cold. In fact, low temperatures improve the efficiency of solar panels, causing them to run at a voltage of 10-20% higher than usual.

One surprise was that with the sun low in the sky, a significant amount of sunlight was reflected off the bright snow onto the panels — in fact, as much as 30% of the power sometimes available was due to such reflection and appreciable power was generated by the panel facing away from the sun. After all, this reflected light is why people can get snow-blindness.

As the angle of the sunlight with the surface of the panels varies throughout the day due to the sun's changing position, the current-voltage curve of the panels changes. This can also happen due to temperature changes, and because of drops in illumination due to clouds, although the polar summers are usually fairly clear. If the electrical system was designed to work at a fixed current or voltage, it would generally not be able to extract the maximum power available. Instead, a dedicated circuit called a peak power tracker is used to continuously adjust the power being drawn from the panel — up to 40 times a second it draws more or less current to see if the new draw provides more power than the last. Starting from a low current state, the logic increases the current until each step starts to reduce the power, and so it steps back. This 'bang-bang' way of testing whether the panel is at maximum output is convenient because it requires no details about the panel being used.

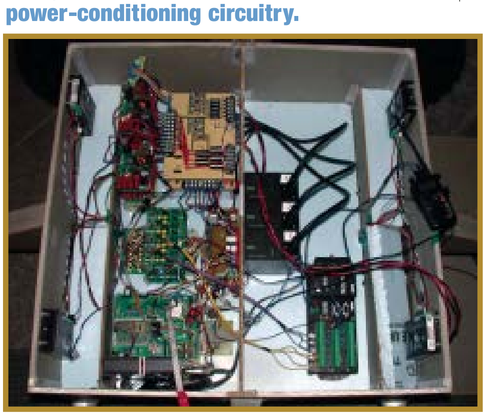
Because the sunlight falling on a panel can change rapidly as the robot crosses an ice ridge or sastrugi, the power-tracking logic (implemented on a microcontroller) is especially fast compared with the systems used, for example, on domestic rooftop solar panels. Another neat trick is that — apart from special situations like battery recharge stops or payload operations — the system can be programmed to match the motor current draw to the solar panel output, thereby minimizing the current going into or out of the batteries. This improves battery life, and avoids the need to have a shunt resistor into which excess power would have to be dissipated (which would, in turn, warm the panels, making them less efficient).

The second factor in the robot's performance is its net traction (T) compared to its weight (W). This depends on the rolling resistance of the tires and on the compaction of the snow. The goal was to achieve (T/W) of 0.5 or better, meaning that the vehicle could climb a 30 degree slope or tow a substantial load. The robot has large tires — 15 cm wide by 50 cm across. These have walls stiff enough to support the robot with zero inflation pressure. The researchers calculated a likely resistance-to-weight ratio of 0.25.

The Cool Robot was tested in summer 2005 at Summit Camp in Greenland. This site is at 72 degrees North and is at an altitude of some 3,200 m. Even though the robot weighs only 61 kg, it can tow a much heavier load on a sled behind it. It drove beautifully, observed Jim Lever of CRREL (US Army Cold Regions Research and Engineering Laboratory), and consumed even less power than predicted. It was able to climb 0.3 m high steep-faced berms, and in tests dragging up to three researchers on a sled on level snow, it could drag 3.5 times its own weight. Streeter notes, however, that in the field the toughest thing was diagnosing problems — it's easy in the lab to feel for a broken part or put a scope on a circuit, but at 15 degrees in wind chill, it's a different story!

The robot typically sank abut 5 cm into the snow, using

The guts of the robot — most of it is power-conditioning circuitry.



The key to good traction performance are the large wheels.

an ultrasonic ranger to measure the amount of sinkage. In fact, on the cold, hard snow of the Antarctic, the robot should sink less and therefore need less power to move than on the soft late July snow at Summit Camp.

In principle, the design should scale up well; larger robots generating more power and being able to carry more, faster, but the present design is a good compromise, being affordable and easy to transport. In fact, the ability to fit the disassembled robot into a tiny Twin Otter plane for delivery to the South Pole Station was a factor in sizing the robot.

ACKNOWLEDGEMENTS

The author thanks Alex Streeter, Laura Ray, and Jim Lever for sharing their experiences with the robots in the preparation of this article.


The robot climbs foot-high obstacles with ease.

The design goal for the Cool Robot is to cross 500 km of the Antarctic plateau with 15 kg of payload in less than two weeks, and the team's Greenland tests suggest that their 60 kg design should be able to achieve this. With further funding, the team hopes to build a fleet of five such robots. Of course, during polar summer, the robots can't drive off into the sunset, because it never happens! SV

Seeing With OpenCV

Implementing Eigenface

by Robin Hewitt

PART 5

Last month's article explained how the face recognition method called eigenface works. This month's article concludes both the OpenCV series and the eigenface topic with a detailed look at a complete program for implementing eigenface with OpenCV.

Here's a brief recap of last month's article explaining how eigenface works.

Eigenface consists of two phases: learning and recognition. In the learning phase, you give eigenface one or more face images for each person you want it to recognize. These images are called the training images. In the recognition phase, when you give eigenface a face image, it responds by telling you which training image is "closest" to the new face image.

Eigenface uses the training images to "learn" a face model. This face model is created by applying a method called Principal Components Analysis (PCA) to reduce the "dimensionality" of these images. Eigenface defines image dimensionality as the number of pixels in an image.

The lower dimensionality representation that eigenface finds during the learning phase is called a subspace. In the recognition phase, it reduces the dimensionality of the input image by "projecting" it onto the subspace it found during learning. "Projecting onto a subspace" means finding the closest point in that subspace. After the unknown face image has been projected, eigenface calculates the distance between it and each training image. Its output is a pointer to the closest training image. You can then look up which person eigenface identified.

Setting Up for Eigenface

In use, you'll probably want to combine eigenface with the face detection method presented in Part 2 of this series. To simplify the example code for this article, however, I'll be assuming you already have a set of training images and a set

FIGURE 1. The two input files for the eigenface program: train.txt and test.txt. The paths in these input files point to images in the ORL face database.

train.txt - Note...

File Edit Format View Help

```
1 s1/1.pgm
2 s2/1.pgm
4 s4/1.pgm
```

test.txt - Notepad

File Edit Format View Help

```
1 s1/1.pgm
2 s2/1.pgm
4 s4/1.pgm
1 s1/2.pgm
2 s2/2.pgm
4 s4/2.pgm
1 s1/3.pgm
2 s2/3.pgm
4 s4/3.pgm
1 s1/4.pgm
2 s2/4.pgm
4 s4/4.pgm
1 s1/5.pgm
2 s2/5.pgm
4 s4/5.pgm
1 s1/6.pgm
2 s2/6.pgm
4 s4/6.pgm
```

main()

```
#include <stdio.h>
#include <string.h>
#include "cv.h"
#include "cvaux.h"
#include "highgui.h"

//// Global variables
int nTrainFaces                = 0;   // number of training images
int nEigens                    = 0;   // number of eigenvalues
IplImage ** faceImgArr         = 0;   // array of face images
CvMat    *  personNumTruthMat  = 0;   // array of person numbers
IplImage *  pAvgTrainImg       = 0;   // the average image
IplImage ** eigenVectArr       = 0;   // eigenvectors
CvMat * eigenValMat            = 0;   // eigenvalues
CvMat * projectedTrainFaceMat  = 0;   // projected training faces

//// Function prototypes
void learn();
void recognize();
void doPCA();
void storeTrainingData();
int  loadTrainingData(CvMat ** pTrainPersonNumMat);
int  findNearestNeighbor(float * projectedTestFace);
int  loadFaceImgArray(char * filename);
void printUsage();

void main( int argc, char** argv )
{
   // validate that an input was specified
   if( argc != 2 )
   {
      printUsage();
      return;
   }

   if( !strcmp(argv[1], "train") ) learn();
   else if( !strcmp(argv[1], "test") ) recognize();
   else
   {
      printf("Unknown command: %s\n", argv[1]);
      printUsage();
   }
}
```

FIGURE 2. The top-level source listing for the eigenface program.

of test images. These face images must all be *exactly the same size*.

For the examples in this article, I've used a free, publicly available face database — the Olivetti Research Lab's (ORL) Face Database. The URL is listed in the References and Resources sidebar.

To set up for using eigenface, unzip the ORL database in the same directory you'll use to run eigenface. This database contains 10 face images for each of 40 subjects. These are organized into 40 directories, named s1-s40. Each directory contains 10 images, named 1.pgm-10.pgm. All the ORL images are already the same size — 92 x 112 pixels.

You'll also need two input files: train.txt and test.txt. Figure 1 shows an example of these input files. Both files use the same format: person number, whitespace, path to image file.

You may have noticed that the first three test images are the same as the training images. These are useful test cases, because eigenface should always give the right answer when you ask it to recognize one of its training images. If it doesn't, you know you have some debugging to do!

To run the learning phase of this eigenface program, enter

```
eigenface train
```

at the command prompt. To run the recognition phase, enter

```
eigenface test
```

Source Code

Figures 2-10 contain the complete source listing for a basic eigenface program. To keep the presentation simple, I've omitted most of the error checking.

The Top-Level Listing

Figure 2 shows the top-level source listing for eigenface — a program to learn and recognize faces with OpenCV's eigenface methods. At line 4, it includes cvaux.h. Until now, we've only included cv.h and highgui.h. But we'll use two specialized functions for face recognition, `cvCalcEigenObjects()` and `cvEigenDecomposite()`, that are defined in cvaux.h.

The primary variables for eigenface are defined at lines 8-15. One of the datatypes here, `CvMat`, is one we haven't used before. This is OpenCV's matrix datatype. A matrix contains a table of data, arranged as rows and columns.

If you only need to hold data temporarily within your program, an ordinary C-style array is usually a little easier to use than `CvMat`. But the `CvMat` datatype can be nice when you want to take advantage of OpenCV functions for working with matrix data. The ones we'll use are OpenCV's persistence functions. With these, you can store matrix data with a single line of code. Reading it back into your programs later is just as easy. Here, I've used a C array for two variables (`faceImgArr` and `eigenVectArr`) and `CvMat` for several others.

The `main()` function simply reads the input string, then calls either the `learn()` method or the `recognize()` method. Figure 3 shows the `printUsage()` helper function.

FIGURE 3. The `printUsage()` helper function.

printUsage()

```
void printUsage()
{
  printf("Usage: eigenface <command>\n",
         "  Valid commands are\n"
         "    train\n"
         "    test\n");
}
```

The Learning Phase

Figure 4 shows the `learn()` function, which implements the learning phase as four steps:

1) Load the training data (line 6).

2) Do PCA on it to find a subspace (line 16).

3) Project the training faces onto the PCA subspace (lines 20-29).

4) Save all the training information (line 32).
   a) Eigenvalues
   b) Eigenvectors
   c) The average training face image
   d) Projected faces
   e) Person ID numbers

The next four subsections analyze each

learn()

```
1  void learn()
2  {
3    int i;
4
5    // load training data
6    nTrainFaces = loadFaceImgArray("train.txt");
7    if( nTrainFaces < 2 )
8    {
9      fprintf(stderr,
10       "Need 2 or more training faces\n"
11       "Input file contains only %d\n", nTrainFaces);
12     return;
13   }
14
15   // do PCA on the training faces
16   doPCA();
17
18   // project the training images onto the PCA subspace
19   projectedTrainFaceMat = cvCreateMat(nTrainFaces, nEigens, CV_32FC1);
20   for(i=0; i<nTrainFaces; i++)
21   {
22     cvEigenDecomposite(
23       faceImgArr[i],
24       nEigens,
25       eigenVectArr,
26       0, 0,
27       pAvgTrainImg,
28       projectedTrainFaceMat->data.fl + i*nEigens);
29   }
30
31   // store the recognition data as an xml file
32   storeTrainingData();
33 }
```

FIGURE 4. The `learn()` function. This function implements the learning phase of eigenface.

```
loadFaceImgArray()

1  int loadFaceImgArray(char * filename)
2  {
3     FILE * imgListFile = 0;
4     char imgFilename[512];
5     int iFace, nFaces=0;
6
7     // open the input file
8     imgListFile = fopen(filename, "r");
9
10    // count the number of faces
11    while( fgets(imgFilename, 512, imgListFile) ) ++nFaces;
12    rewind(imgListFile);
13
14    // allocate the face-image array and person number matrix
15    faceImgArr        = (IplImage **)cvAlloc( nFaces*sizeof(IplImage *) );
16    personNumTruthMat = cvCreateMat( 1, nFaces, CV_32SC1 );
17
18    // store the face images in an array
19    for(iFace=0; iFace<nFaces; iFace++)
20    {
21       // read person number and name of image file
22       fscanf(imgListFile,
23          %d %s, personNumTruthMat->data.i+iFace, imgFilename);
24
25       // load the face image
26       faceImgArr[iFace] = cvLoadImage(imgFilename, CV_LOAD_IMAGE_GRAYSCALE);
27    }
28
29    fclose(imgListFile);
30
31    return nFaces;
32 }
```

FIGURE 5. The `loadFaceImgArray()` function loads face images and person ID numbers for both the learning and recognition phases.

of these steps in detail.

Loading Face Images for Training or Test

The `loadFaceImgArray()` function (Figure 5) loads face images and person ID numbers for both the learning and recognition phases.

The face images — assumed here to be all the same size — are stored in the global variable `faceImgArr`; `loadFaceImgArray()` returns the number of face images loaded.

The person ID numbers are stored in a `CvMat` variable, `personNumTruthMat`. "Truth" here refers to the AI term, "ground truth." It means the values in this variable are the true (correct) values for each face image. During the learning phase, those are the only type of person ID numbers we have. But during the recognition phase, the program will have both ground truth values (specified in the file test.txt) and the output from eigenface. Having both allows us to evaluate how well eigenface does under varying conditions.

The `cvCreateMat()` function — called at line 16 — creates the `personNumTruthMat` variable. This function takes three parameters: the number of rows, the number of columns, and the datatype for the matrix. On a 32-bit operating system, the datatype for a matrix of int values is `CV_32SC1`. The `S` here stands for "signed," and `C1` indicates that the matrix has one channel. (A matrix can have up to four channels. Multiple channels allow you to add a third dimension to a matrix variable.)

The `CvMat` datatype is a `struct`, with the raw data stored in the `struct` element `data`; `data` is defined as a `union` (the definition is given in the CXCORE documentation), with `int` data accessed as `data.i`. Lines 22-23 show one way to access `CvMat` values — as offsets from the start of the data buffer.

Matrix rows are aligned to start on four-byte intervals. The in-memory row width, in bytes, is stored in `CvMat.step`. Since we're using a four-byte datatype with this matrix (and also, since it has only one row), we can ignore `CvMat.step`. But, if you create a matrix of, for example, char data, you may need to take the step size into account when you access the data elements.

Finding the PCA Subspace

The code to find the PCA subspace is in Figure 6. It calls the built-in OpenCV function for doing PCA, `cvCalcEigenObjects()`, at lines 27-36. The remainder of `doPCA()` creates the output variables that will hold the PCA results when `cvCalcEigenObjects()` returns.

At line 8, the number of eigenvalues is set to one less than the number of training images. (As explained last month, this is the maximum number of eigenvalues we can find.)

Lines 11-15 create the global image array `eigenVectArr`. When `cvCalcEigenObjects()` returns, each image in this array will hold one eigenvector, one "eigenface," in other words. Note that these are floating-point images, with data depth = `IPL_DEPTH_32F`.

At line 18, another matrix is created — `eigenValMat`. This matrix will hold the eigenvalues. The eigenvalues are floating-point numbers, and we only need one channel for this, so the matrix type is `CV_32FC1`. That gives us a one-channel matrix, with 32-bit, floating-point data values.

To do PCA, the dataset must first be "centered." For our face images, this means finding the average image — an image in which each pixel contains the average value for that pixel across all face images in the training set. The dataset is centered by subtracting the average face's pixel values from each training image.

You don't have to do that yourself. It happens inside `cvCalcEigenObjects()`. But you do need to hold onto the average image, because you'll need it later to project the data. So you'll need to allocate memory for the average image. The code for doing that is at line 21. Note that — like the eigenvectors — this is a floating-point image.

The last step before calling `cvCalcEigenObjects()` is to prepare a data structure called `CvTermCriteria`. The fields in this structure specify termination criteria for iterative algorithms such as PCA. You can read more about `CvTermCriteria` options in the CXCORE documentation. Here, we can simply tell it to compute each eigenvalue, then stop, since that's all we need.

FIGURE 6. `doPCA()` finds the PCA subspace using OpenCV's `cvCalcEigenObjects()` function.

doPCA()

```
1  void doPCA()
2  {
3     int i;
4     CvTermCriteria calcLimit;
5     CvSize faceImgSize;
6
7     // set the number of eigenvalues to use
8     nEigens = nTrainFaces-1;
9
10    // allocate the eigenvector images
11    faceImgSize.width  = faceImgArr[0]->width;
12    faceImgSize.height = faceImgArr[0]->height;
13    eigenVectArr = (IplImage**)cvAlloc(sizeof(IplImage*) * nEigens);
14    for(i=0; i<nEigens; i++)
15       eigenVectArr[i] = cvCreateImage(faceImgSize, IPL_DEPTH_32F, 1);
16
17    // allocate the eigenvalue array
18    eigenValMat = cvCreateMat( 1, nEigens, CV_32FC1 );
19
20    // allocate the averaged image
21    pAvgTrainImg = cvCreateImage(faceImgSize, IPL_DEPTH_32F, 1);
22
23    // set the PCA termination criterion
24    calcLimit = cvTermCriteria( CV_TERMCRIT_ITER, nEigens, 1);
25
26    // compute average image, eigenvalues, and eigenvectors
27    cvCalcEigenObjects(
28       nTrainFaces,
29       (void*)faceImgArr,
30       (void*)eigenVectArr,
31       CV_EIGOBJ_NO_CALLBACK,
32       0,
33       0,
34       &calcLimit,
35       pAvgTrainImg,
36       eigenValMat->data.fl);
37 }
```

The code for that is at line 24.

Now that all the output variables are ready, we call `cvCalcEigenObjects()` to compute the PCA subspace for the training faces. The last parameter, `eigenValMat->data.fl`, is the pointer to the data values in eigenValMat. Here, we use the `data.fl` field, not `data.i`, since this matrix variable holds floating-point data.

Projecting the Training Faces

Now that you've found a subspace using PCA, you can convert the training images to points in this subspace. As explained last month, this step is called "projecting" the training image. The OpenCV function for this step is called `cvEigenDecomposite()` (Figure 4, line 22).

The OpenCV function names are, unfortunately, confusing. Not only is the projection function oddly named, but there's also a function named "EigenProjection" that doesn't project image data onto the subspace. In fact, it does the opposite. It restores (uncompresses) projected data, turning it back into the original image. The correct name for doing that is Reconstruction, not Projection!

You'll need a place to put the projected training images. Line 19 in Figure 4 creates a matrix for that purpose. The for loop, at lines 20-29, calls `cvEigenDecomposite()` once for each training image.

Saving the Learned Face Model

The small bit of extra effort to use OpenCV's `CvMat` datatype really pays off when it comes time to save the training data! Figure 7 shows the complete code for saving all the data for your learned face representation as an XML file using OpenCV's built-in persistence functions.

At line 7, the call to `cvOpenFileStorage()` opens an XML file named facedata.xml. The last parameter to this function controls the access mode. Here, it's `CV_STORAGE_WRITE`, which means to create (or overwrite) that file and open it for writing.

To write basic C-language data — integers, floating-point values, and strings — in XML format, you can use the function `cvWrite<datatype>()`. For example, the call to `cvWriteInt()` at line 10 writes the number of eigenvalues as <nEigens>2</nEigens>.

The really nice thing about using OpenCV's persistence functions is that it's just as easy to save complex datatypes, such as an image or matrix.

storeTrainingData()

```
1  void storeTrainingData()
2  {
3     CvFileStorage * fileStorage;
4     int i;
5
6     // create a file-storage interface
7     fileStorage = cvOpenFileStorage( "facedata.xml", 0, CV_STORAGE_WRITE );
8
9     // store all the data
10    cvWriteInt( fileStorage, "nEigens", nEigens );
11    cvWriteInt( fileStorage, "nTrainFaces", nTrainFaces );
12    cvWrite(fileStorage, "trainPersonNumMat", personNumTruthMat, cvAttrList(0,0));
13    cvWrite(fileStorage, "eigenValMat", eigenValMat, cvAttrList(0,0));
14    cvWrite(fileStorage, "projectedTrainFaceMat", projectedTrainFaceMat, cvAttrList(0,0));
15    cvWrite(fileStorage, "avgTrainImg", pAvgTrainImg, cvAttrList(0,0));
16    for(i=0; i<nEigens; i++)
17    {
18       char varname[200];
19       sprintf( varname, "eigenVect_%d", i );
20       cvWrite(fileStorage, varname, eigenVectArr[i], cvAttrList(0,0));
21    }
22
23    // release the file-storage interface
24    cvReleaseFileStorage( &fileStorage );
25 }
```

FIGURE 7. `storeTrainingData()` saves all the data for the learned face representation as an XML file using OpenCV's built-in persistence functions.

```
recognize()
1  void recognize()
2  {
3     int i, nTestFaces  = 0;         // the number of test images
4     CvMat * trainPersonNumMat = 0;  // the person numbers during training
5     float * projectedTestFace = 0;
6
7     // load test images and ground truth for person number
8     nTestFaces = loadFaceImgArray("test.txt");
9     printf("%d test faces loaded\n", nTestFaces);
10
11    // load the saved training data
12    if( !loadTrainingData( &trainPersonNumMat ) ) return;
13
14    // project the test images onto the PCA subspace
15    projectedTestFace = (float *)cvAlloc( nEigens*sizeof(float) );
16    for(i=0; i<nTestFaces; i++)
17    {
18       int iNearest, nearest, truth;
19
20       // project the test image onto the PCA subspace
21       cvEigenDecomposite(
22          faceImgArr[i],
23          nEigens,
24          eigenVectArr,
25          0, 0,
26          pAvgTrainImg,
27          projectedTestFace);
28
29       iNearest = findNearestNeighbor(projectedTestFace);
30       truth    = personNumTruthMat->data.i[i];
31       nearest  = trainPersonNumMat->data.i[iNearest];
32
33       printf("nearest = %d, Truth = %d\n", nearest, truth);
34    }
35 }
```

FIGURE 8. The `recognize()` function implements the recognition phase of the eigenface program.

Lines 12-15 add three matrices and an image to the same XML file. The built-in persistence functions save not only the row and column data, but all the header information, as well. Here's the XML that line 13 generates:

```
<eigenValMat type_id="opencv-matrix">
 <rows>1</rows>
 <cols>2</cols>
 <dt>f</dt>
 <data>
 14279064. 9614034.</data></eigenValMat>.
```

The second parameter to the `Write()` functions is a string. The string can be anything you like, but to ensure uniqueness — and for clarity's sake — it's usually a good idea to make it the same as your variable name.

When you've finished writing data, close the file and release the file storage as in line 24.

The Recognition Phase

Figure 8 shows the `recognize()` function, which implements the recognition phase of the eigenface program. It has just three steps. Two of them – loading the face images and projecting them onto the subspace — are already familiar.

As described above, the face images for recognition testing should be listed in a file named test.txt, using the same format as in train.txt. At line 8, the call to `loadFaceImgArray()` loads these into the `faceImgArr` and stores the ground truth for the person ID number in `personNumTruthMat`. This step is similar to line 6 of the `learn()` function in Figure 4. Here, the number of face images is stored in the local variable, `nTestFaces`.

We also need to load the global variable `nTrainFaces`, as well as most of the other training data — `nEigens`, `EigenVectArr`, `pAvgTrainImg`, and so on. The function `loadTrainingData()` in Figure 9 does that for us. Again, OpenCV's persistence functions make this step easy. To open file storage for reading, use the `CV_STORAGE_READ` flag. Then, simply call the appropriate `Read()` function for each variable. OpenCV locates and loads each data value in the XML file by name. When the variable is a `CvMat` type, OpenCV creates a new matrix for you automatically, then sets its data values.

The last parameter in the `Read()` function's interface is a default value. If a named variable is missing from the XML file, it will be set to the default. For pointer types — such as the matrices — it's a good idea to set the default to 0. You can then add a

```
loadTrainingData()
int loadTrainingData(CvMat ** pTrainPersonNumMat)
{
   CvFileStorage * fileStorage;
   int i;

   // create a file-storage interface
   fileStorage = cvOpenFileStorage( "facedata.xml", 0, CV_STORAGE_READ );
   if( !fileStorage )
   {
      fprintf(stderr, "Can't open facedata.xml\n");
      return 0;
   }

   nEigens = cvReadIntByName(fileStorage, 0, "nEigens", 0);
   nTrainFaces = cvReadIntByName(fileStorage, 0, "nTrainFaces", 0);
   *pTrainPersonNumMat = (CvMat *)cvReadByName(fileStorage, 0, "trainPersonNumMat", 0);
   eigenValMat  = (CvMat *)cvReadByName(fileStorage, 0, "eigenValMat", 0);
   projectedTrainFaceMat =
      (CvMat *)cvReadByName(fileStorage, 0, "projectedTrainFaceMat", 0);
   pAvgTrainImg = (IplImage *)cvReadByName(fileStorage, 0, "avgTrainImg", 0);
   eigenVectArr = (IplImage **)cvAlloc(nTrainFaces*sizeof(IplImage *));
   for(i=0; i<nEigens; i++)
   {
      char varname[200];
      sprintf( varname, "eigenVect_%d", i );
      eigenVectArr[i] = (IplImage *)cvReadByName(fileStorage, 0, varname, 0);
   }

   // release the file-storage interface
   cvReleaseFileStorage( &fileStorage );

   return 1;
}
```

FIGURE 9. OpenCV's persistence functions make it easy to load the saved training data from the XML file.

FIGURE 10. The findNearestNeighbor() function computes the distance from the projected test image to each projected training example to find the closest training image.

findNearestNeighbor()

```
1  int findNearestNeighbor(float * projectedTestFace)
2  {
3     double leastDistSq = DBL_MAX;
4     int i, iTrain, iNearest = 0;
5
6     for(iTrain=0; iTrain<nTrainFaces; iTrain++)
7     {
8        double distSq=0;
9
10       for(i=0; i<nEigens; i++)
11       {
12          float d_i =
13             projectedTestFace[i] -
14             projectedTrainFaceMat->data.fl[iTrain*nEigens + i];
15          distSq += d_i*d_i;
16       }
17
18       if(distSq < leastDistSq)
         {
19          leastDistSq = distSq;
20          iNearest = iTrain;
21       }
22    }
23
24    return iNearest;
25 }
```

validation check to make sure these pointers have a non-zero value before you use them. To simplify the example code, I've omitted these (and similar) validation steps from the loadTrainingData() function.

After all the data are loaded, the final step in the recognition phase is to project each test image onto the PCA subspace and locate the closest projected training image. The for loop at lines 16-34 of the recognize() function (Figure 8) implements this final step. The call to cvEigenDecomposite(), which projects the test image, is similar to the face-projection code in the learn() function.

As before, we pass it the number of eigenvalues (nEigens), and the array of eigenvectors (eigenVectArr). This time, however, we pass a test image, instead of a training image, as the first parameter. The output from cvEigenDecomposite() is stored in a local variable — projectedTestFace. Because there's no need to store the projected test image, I've used a C array for projectedTestFace, rather than an OpenCV matrix.

Finding the Nearest Neighbor

As last month's article explained, eigenface "recognizes" a face image by looking for the training image that's closest to it in the PCA subspace. Finding the closest training example in a learned subspace is a very common AI technique. It's called Nearest Neighbor matching.

Figure 10 shows the code for the findNearestNeighbor() function. It computes distance from the projected test image to each projected training example. The distance basis here is "Squared Euclidean Distance." As last month's column explained, to calculate Euclidean distance between two points, you'd add up the squared distance in each dimension, then take the square root of that sum. Here, we take the sum, but skip the square root step. The final result is the same, because the neighbor with the smallest distance also has the smallest squared distance, so we can save some computation time by comparing squared values.

The for loop at lines 6-22 computes the squared distance to each projected training image, and keeps track (at lines 18-21) of which training image is closest.

The return value is the index of the closest training image. In the recognize() function (Figure 8), this return value is used, at line 31, to look up the person ID number associated with the nearest training image.

Here is the print output from the recognize() function:

```
nearest = 1, Truth = 1
nearest = 2, Truth = 2
nearest = 4, Truth = 4
nearest = 1, Truth = 1
nearest = 2, Truth = 2
nearest = 4, Truth = 4
nearest = 1, Truth = 1
nearest = 2, Truth = 2
nearest = 4, Truth = 4
nearest = 1, Truth = 1
nearest = 2, Truth = 2
nearest = 4, Truth = 4
nearest = 1, Truth = 1
nearest = 2, Truth = 2
nearest = 4, Truth = 4
nearest = 1, Truth = 1
nearest = 2, Truth = 2
nearest = 1, Truth = 4
```

Not bad! We only have one mismatch: the last test image was misrecognized as Subject 1 instead of 4.

Improving Eigenface

Having a framework like this for training and testing will make it easier for you to add improvements to eigenface and to test their effects.

One of the first improvements you might want to add is to change the way distance is measured. The original eigenface paper used Euclidean distances between points, and that's the distance basis I've used in findNearestNeighbors(). But a different basis, called Mahalanobis distance (after its inventor), usually gives better results.

One of the things that happens when you project a face image onto the PCA subspace is that each dimension receives a certain amount of stretch. The amount of stretch isn't the same, though, in every direction. The directions that correspond to the largest eigenvalues get stretched far more than the directions associated with smaller eigenvalues. Because Euclidean distance ignores this stretching, using it to measure distance is approximately the same as using only one eigenvector and ignoring the rest!

It's easy to switch from Euclidean to Mahalanobis distance. Just change

References and Resources

- *OpenCV on Sourceforge*
  http://sourceforge.net/projects/opencvlibrary

- *Official OpenCV Usergroup*
  http://tech.groups.yahoo.com/group/OpenCV

- Turk, M., Pentland, A., *Face recognition using eigenfaces*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1991.

- *ORL Database*
  www.cl.cam.ac.uk/research/dtg/attarchive/facedata base.html

- Source code in this article can be downloaded from **www.cognotics.com/opencv/servo**.

line 15 in `findNearestNeighbors()` from

```
distSq += d_i*d_i;
```

to

```
distSq += d_i*d_i/eigenValMat->data.fl[i];
```

Switching to Mahalanobis distance eliminates the mismatch error mentioned above, bringing recognition accuracy up to 100% for these three subjects.

Where to Go From Here?

This article introduced several new OpenCV concepts. You can gain a deeper understanding of these from the OpenCV documentation. The persistence functions, the `CvTermCriteria struct`, and the `CvMat` datatype are described in detail in the CXCORE documentation. The eigenface functions are described in the CVAUX documentation. The CVAUX documentation isn't linked from the documentation index page, but you can find it in the documentation subdirectory named ref.

If you want to incorporate eigenface into a system that detects faces in live video, you'll first need to detect the face, then extract it into a separate image. Since each face image must be exactly the same size, the easiest way to do that is to define a standard size, say 50 x 50 pixels, ahead of time. Then, when you detect a face, you can use code like this to extract and resize it:

```
CvRect * pFaceRect = (CvRect*)cvGetSeqElem(pRectSeq, 0);
cvSetImageROI(pImg, *pFaceRect);
IplImage * pFaceImg =
   cvCreateImage( STD_SIZE, IPL_DEPTH_8U, 1 );
cvResize(pImg, pFaceImg, CV_INTER_AREA );
```

There are more capabilities built into OpenCV, and many, many more computer vision programs one can create using this library. I hope this short series of articles has given you a taste of what's possible with OpenCV, and perhaps motivated you to explore more of its capabilities.

Be seeing you! **SV**

Build the Ultimate Remote Control

by Michael Simpson

Last month, we created a walking robot called the BioCrab. The problem with such a bot is that there are a great many control points and a remote control does not exist that will allow us to control all its aspects. I am going to show you how to build a custom control system that has four dual axis joysticks, as well as six buttons and a keypad. There is also a 4 x 16 LCD display for configuration and robot feedback. We will add a serial interface that can be connected directly to a robot via a wireless Zigbee module.

FIGURE 1. The Ultimate Setup.

In order to build this remote, you will need some controls like those shown in Figure 2. These will range from buttons to full action joysticks. The buttons weren't an issue, but finding a joystick at a reasonable price was a real problem. A company called P3 America sells them, but even the simplest dual axis joystick was over $100 each. Since I wanted at least four, my remote would probably cost me over $400 for the joysticks alone and over $500 for the completed remote. There just had to be another option.

Vex Radio Interface

What I really wanted was a joystick like you find in those four- or six-channel R/C radios. They seem to be very modular and rugged enough for the kind of remote I wanted to build. Many of you may have already seen the Vex R/C radio shown in Figure 3. When first released, they sold for over $100. A while back, RadioShack dropped the Vex line and many of the radios have made their way to some of the mail order houses at a very reasonable price. At the time of this article, All Electronics was selling them for $29.95 each.

FIGURE 2

With a small screwdriver and a soldering iron, you can retrieve the two joysticks shown in Figure 4 with about 10 minutes of effort. One really nice feature about these joysticks is that you may remove a small spring and the stick will remain where set and not return to the center position. You can do this with either or both axis. This makes them perfect for our remote.

Before I get into the details on building the remote, I thought I would show you how to use the Vex radio with the BioCrab. The Vex radio is a six channel

FIGURE 3

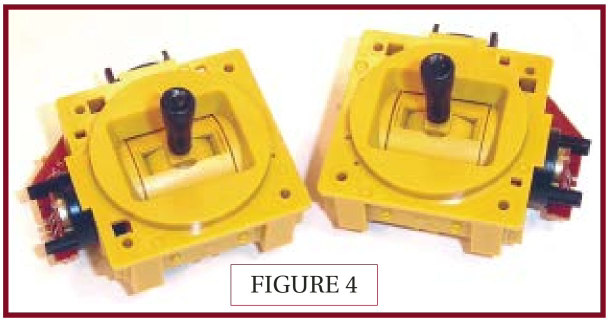


FIGURE 4

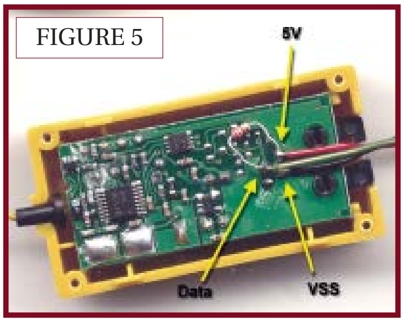
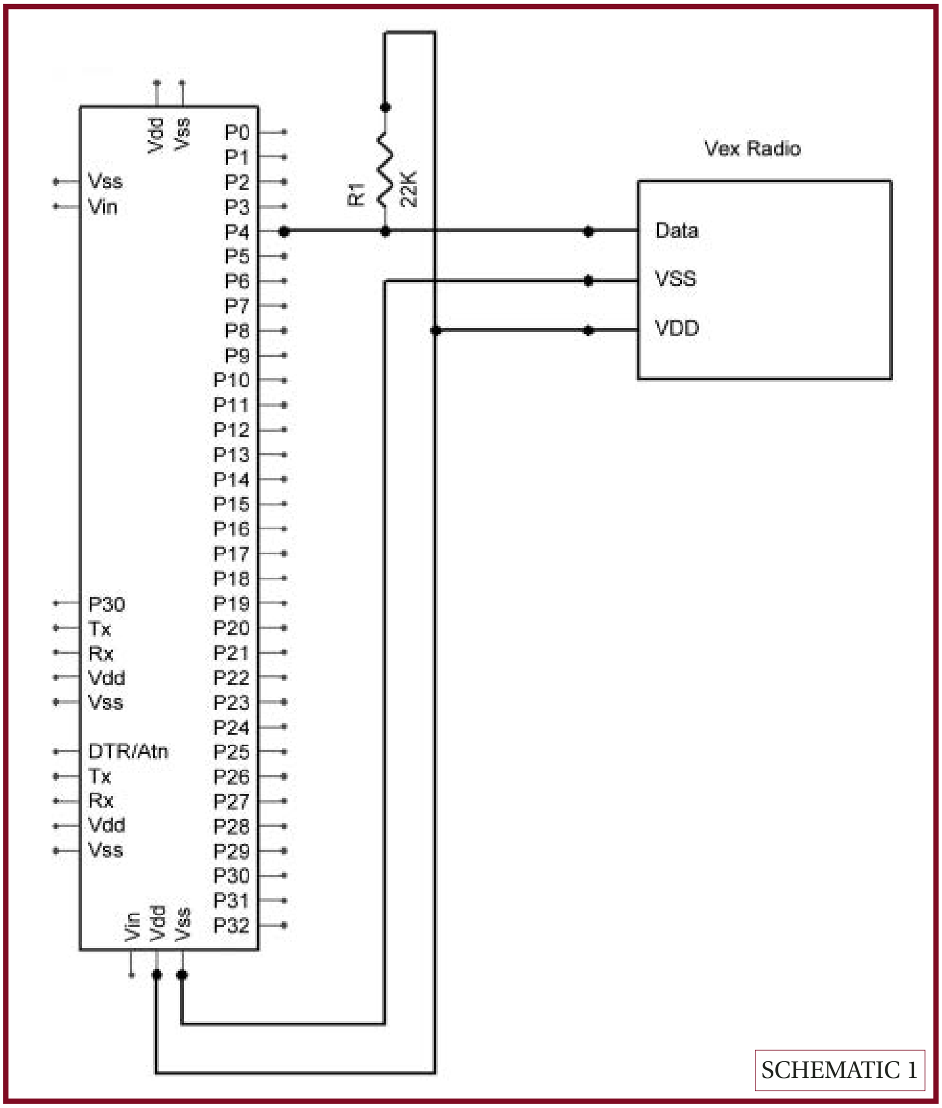


FIGURE 5

FM system. In addition to the two dual axis joysticks, there are four buttons on the back of the remote. While the radio does not have the capacity to control every aspect of the BioCrab, it may be worth experimenting with before you take the radio apart for our project.

The Vex radio kit comes with a receiver that is unlike any other R/C radio receiver you may have seen. The big difference is that the receiver does not contain the channel decoder circuitry for the individual servo channels. While this is not a good thing for R/C enthusiasts, it is perfect for those who want to interface the radio to a microcontroller. The receiver outputs all the pulses for the six channels one right after another. To handle this pulse train, there is a library built into the DiosPro.



SCHEMATIC 1

The radio does require a little bit of an interface to be able to plug it into one of the I/O ports of the DiosPro. You need to supply power to the radio and pull the data pin high with a 22K resistor. To add the resistor and connections for this interface, remove the four screws from the back of the receiver and add three wires as shown in Figure 5. You may also add the 22K resistor as shown or connect it between the data port and 5V at the microcontroller.

To use the radio, connect the radio to the DiosPro as shown in Schematic 1. If you hard-wired the 22K resistor inside the radio, you may omit R1. As an option, the radio kit also comes with a small yellow cable with small modular connectors on each end. You can cut one of the connectors off and attach the small wires to a header. The red wire is Vss, the yellow is Vdd, and green is the data lead.

The heart of the interface is the single readvex function. Simply call this function passing the port number, and six global variables will be populated based on the joystick positions and the four buttons on the back of the receiver. The function will return 0 if the radio is not connected or off. To test the interface, run Program 1.

I have included a program called BioCrabVex.txt that will allow you to use the Vex radio to control the crab. Feel free to experiment.

Custom Remote Base Construction

I will be referencing various parts that I purchased for this project. A complete source list for those items is provided at the end of the article.

Joystick Removal

Before we start, you will need to obtain the four joystick assemblies. On the back of the Vex radio, there are six screws that must be removed. Under the battery holder are an additional three. On the front of the radio, there are four screws holding each joystick in place. Once these are removed, lift the cover off the back of the receiver and

PROGRAM 1

```
DiosPro
'Vex Radio Interface Demo
func main()
  dim stat as integer
loop:
  stat=readvex(4)
  print stat,": ",VCH1," ",VCH2," ",VCH3," ",VCH4," ",VCH5," ",VCH6
  goto loop


endfunc

include \lib\vexradio.lib
```

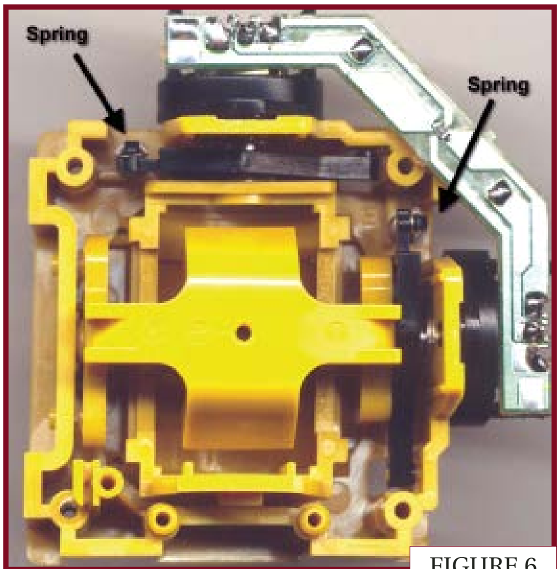


FIGURE 6

expose the joysticks. There is a circuit board in the middle of the radio. Simply lift it off of its posts to gain access to the joysticks. You will have to use your soldering iron to remove the wires that are attached to the joysticks, but once done, they can be removed by simply lifting them out of the radio.

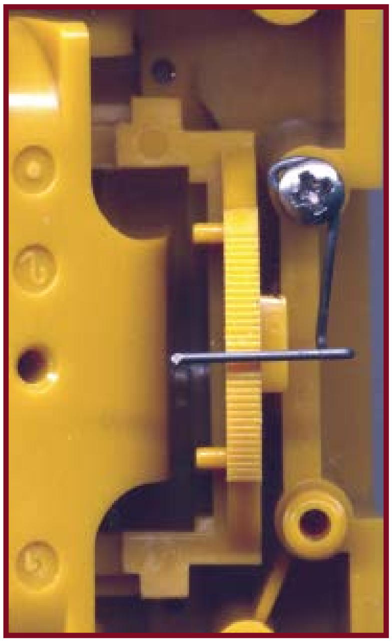


FIGURE 7

The right and left joysticks are the same except for the layout of the small circuit board that is attached to the stick shown in Figure 6. As I mentioned earlier, you may remove the small springs on one of the axes. There is also a small detent area on the vertical axis, and by adding a small bent wire, you can add a cool click detent to that axis as shown in Figure 7. On my control, I removed the spring on all the axis of the bottom two joysticks and added the detent wire to the vertical axis on them, as well.

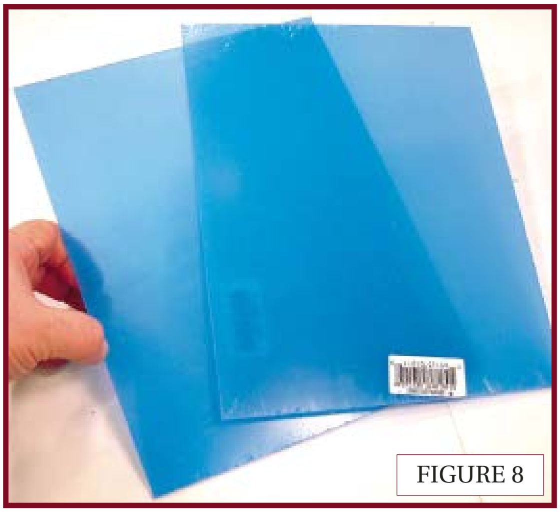


FIGURE 8

Base Construction

You will need to select some sort of material to use for your base. In my case, I used a piece of 1/8" x 8" x 10" clear acrylic shown in Figure 8. I like the acrylic because it comes with a protective coating and its transparency allows you to easily mark the holes for the various boards and controls that you will need to attach. You can purchase this material from your local home center in various sizes.

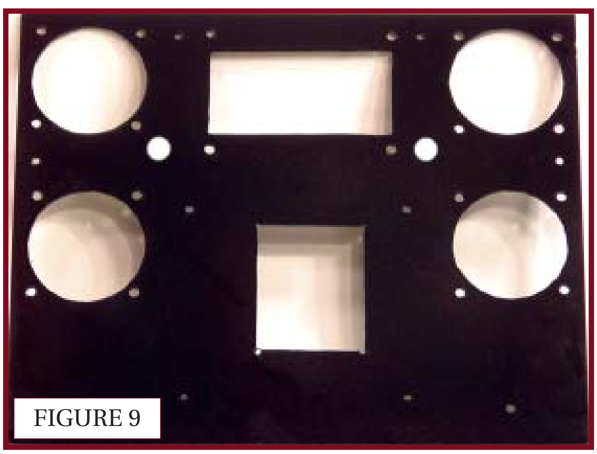


FIGURE 9

The joysticks need a 2" hole cut out in order to mount them. I used a scroll saw, but a 2" hole saw could also do the trick. Use Figure 9 as a rough layout guide. I recommend you lay out your components to get a feel for placement. The LCD needs to be mounted between the top two joysticks. Keep this in mind if you use a different LCD or set of joysticks. Once the holes are cut, you will need to insert the joysticks, then mark the four

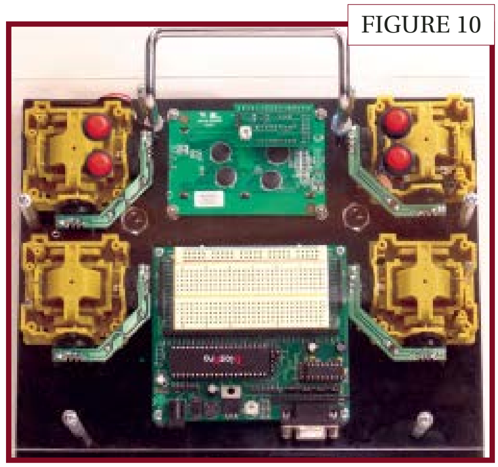


FIGURE 10

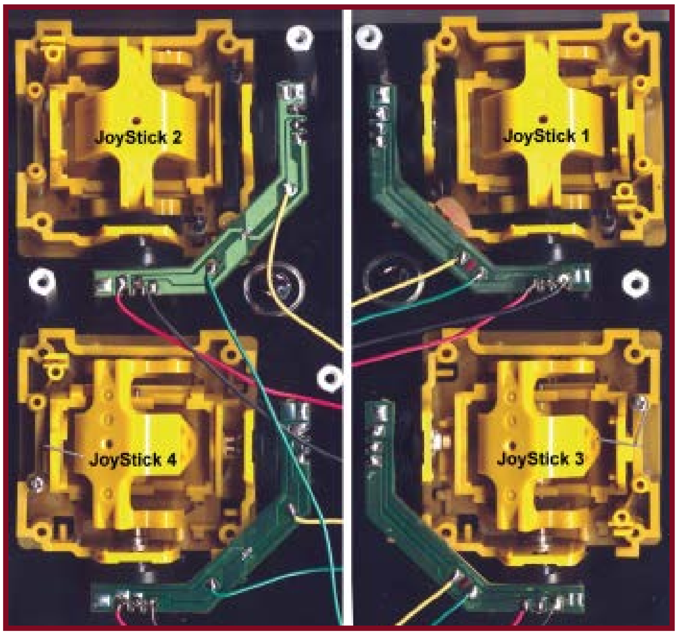


FIGURE 12

mounting holes for each. Attach the joysticks using some 1/2" machine screws. The original screws may be used, but must be cut short as they are too long. This is just a dry fit to aid in assembly; you will have to remove the sticks in order to add the additional cutouts as needed.

We are going to use a Kronos Robotics Serial LCD. This is a four line by 16 character LCD with a LED backlight. To mount the LCD, simply place the LCD in place and mark the four mounting holes and the outline of the LCD to create a cutout. If you plan on keeping the base clear, you don't need to add a cutout for the LCD, simply mark the four mounting holes.

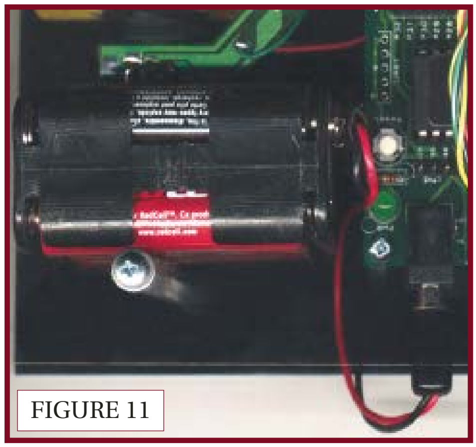


FIGURE 11

The key pad can be mounted under the base with a cutout or on top with a slot cut for the header. The keypad was purchased from All Electronics and is a row/column matrix of keys. Other types will work as well, but you may need to make changes to the software.

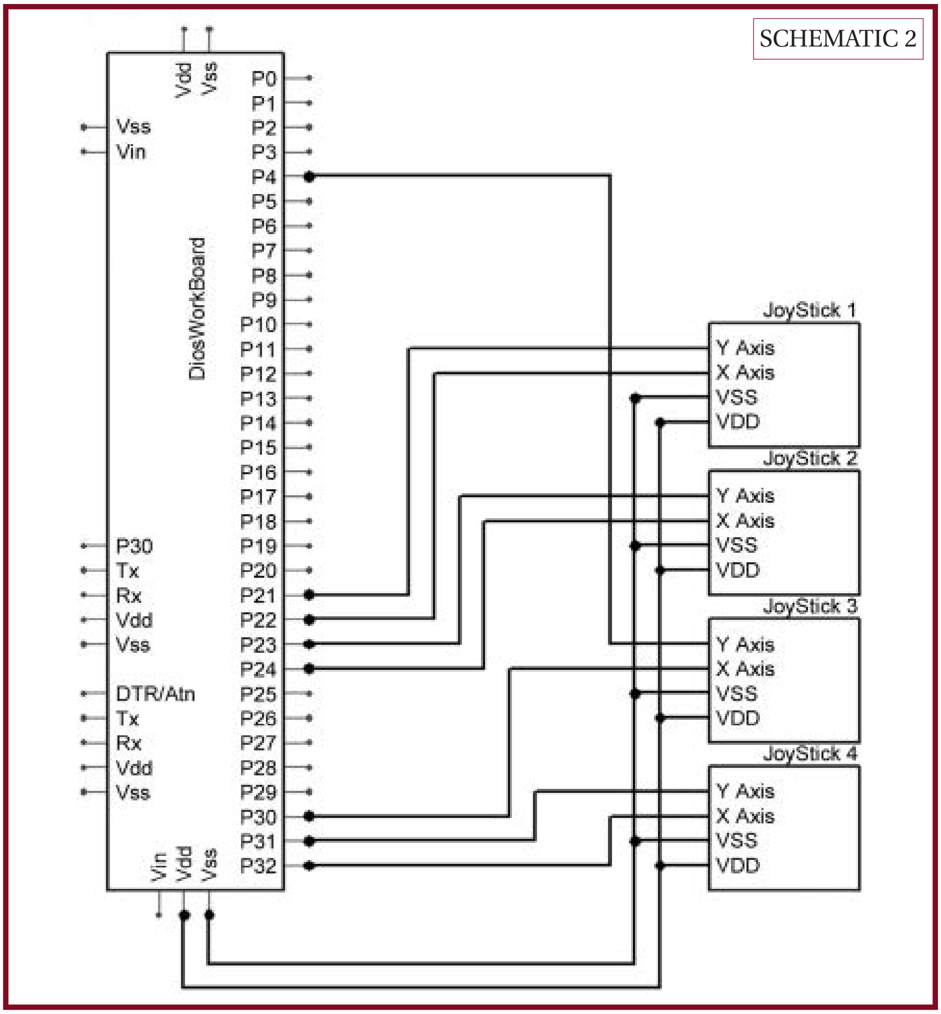
Figure 9 shows all the cutouts and holes in my base. Your layout may be slightly different. Note the two small 1/2" holes. These are for two buttons. You may want to place them in different positions. There will also be four buttons mounted on the bottom base.

Figure 10 shows all the components mounted on the base. It also shows the lower base attached, as well. I used a 2" standoff that I created by connecting a 1" M/F standoff to a 1" F/F standoff. These are #4 standoffs purchased from Jameco. You may have also noticed that I attached the metal handle from the Vex radio to the lower base.

The top two buttons and the lower four buttons were purchased from All Electronics. You will need to drill a 1/2" hole for these. Place the lower button in a position that your fingers can reach while you are manipulating the joysticks.

The Dios WorkBoard Deluxe is mounted to the upper base using four 1" standoffs. These standoffs keep it above the header connected to the keypad. Notice that the breadboard is being used for this project. I recommend its use until you have your remote configured to your satisfaction. It will be very simple to make hardware changes, and once you get the remote operating the way you want, you can

SCHEMATIC 2



remove the breadboard and solder the wires directly to the prototyping area under the breadboard.

At this point, it is important to note that the two joysticks that you removed from the Vex radio are slightly different. The difference is in the thin PCB (printed circuit board) attached to the two potentiometers. One has a small crisscross design. You should mount them as shown in Figures 10 and 12.

The lower standoffs used to attach the upper base to the lower base are positioned in such a way that they hold a six-cell battery pack snugly in place as shown in Figure 11. Here, I used a 9V battery clip attached to a 2.1 coax to plug into the board, but you could also use the onboard power header, as well.

Hookup

Let's take a look at each subsection in detail. This will allow us to test each section as we add it to the system.

Joystick Hookup

Each joystick has a small PCB; you will make the connections as shown in Schematic 1. Figure 12 shows the joystick placement when viewing from the bottom of the control. The yellow wire is connected to the Y axis of each joystick. The green is connected to the X axis. Red is Vdd (5V) and black is Vss.

Load and run the program called Joy1.txt. This program will display the X and Y axis from each of the four joysticks.

LCD Hookup

The serial LCD comes with a small four-pin header that you may mount as needed, but we won't be using it. Simply attach four wires as shown in Figure 13.

The last pin on the serial LCD interface is optional and can be used to set the baud rate to 19200 when tied to Vss. In this application, we are going to use the 19200 baud interface as shown in Schematic 2.

Load and run the program called Joy2.txt. It will display the X and Y axis from each of the four joysticks on the LCD. Note that you may need to adjust the contrast trimmer in order for the characters to be visible.

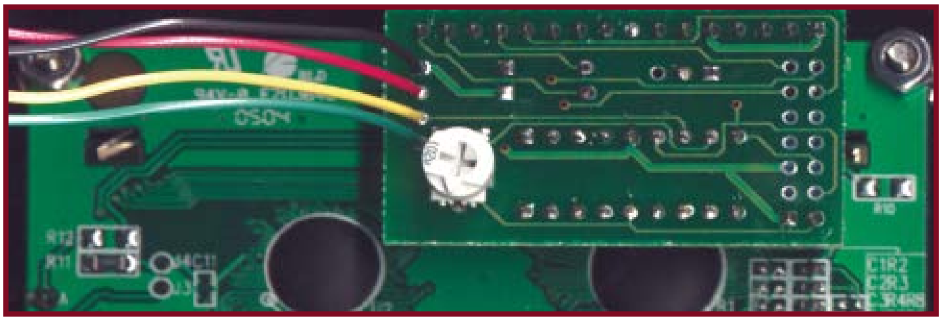


FIGURE 13

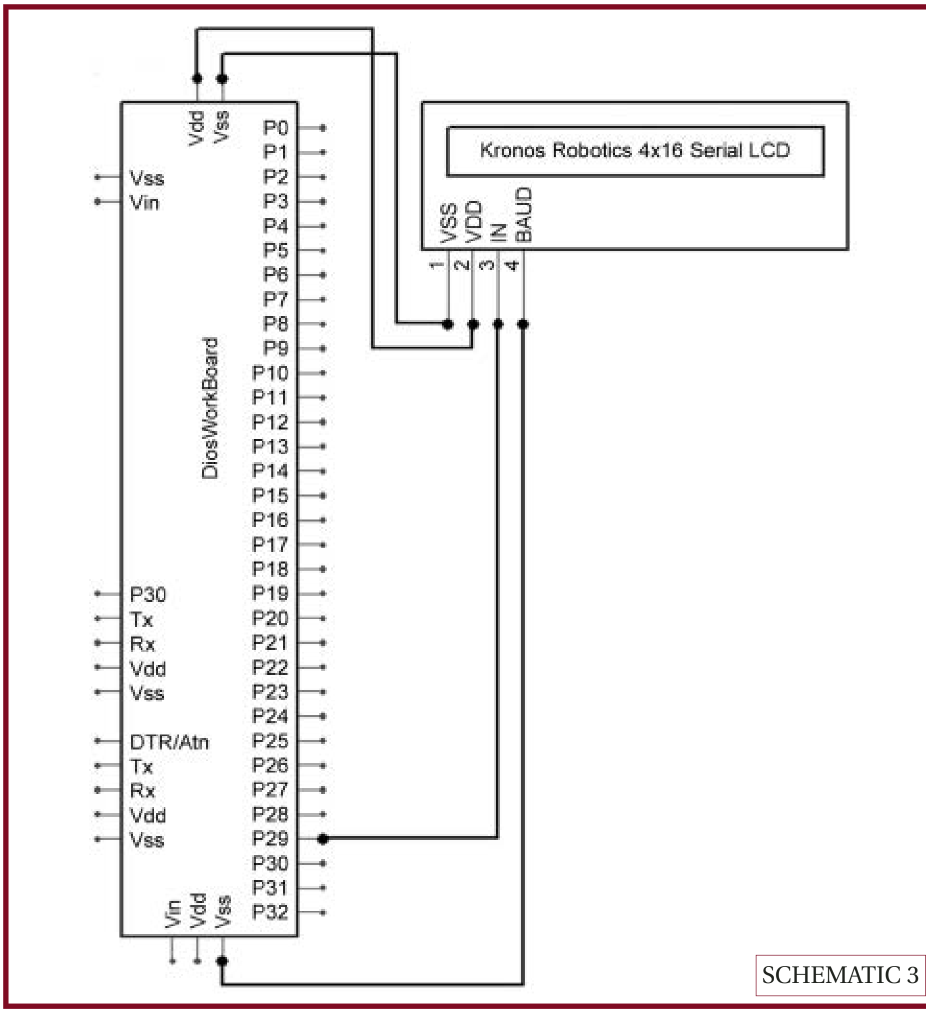
Keypad Hookup

The keypad is very easy to hook up. You only need to connect seven wires to the header on the keypad as shown. Note that the schematic shows the rear view of the keypad. This is the view of the header that you see on the underside of the base. I cut off a seven-pin section from a female header and soldered seven wires. This allowed me to slip the header onto the keypad even while the Dios Workboard is installed. Once the header is attached to the keypad, plug the wires into the ports shown in Schematic 4.

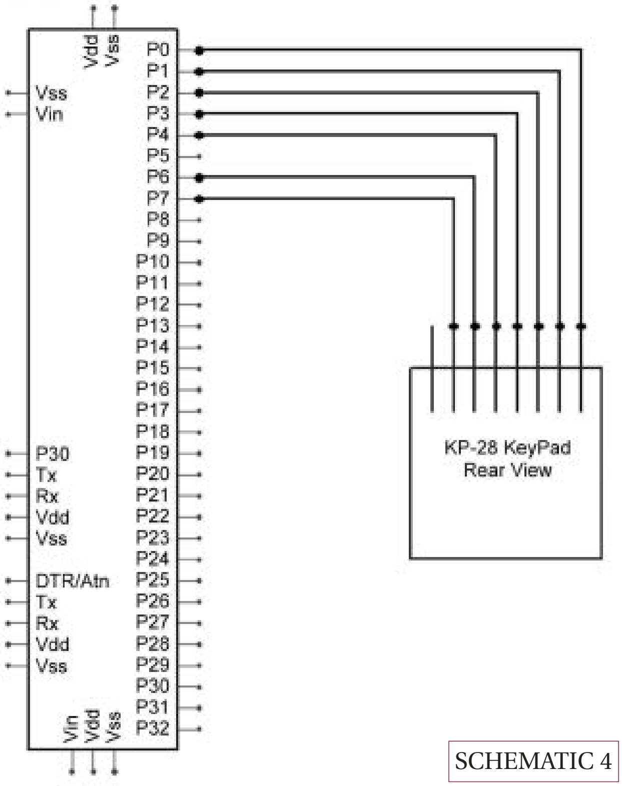
Load and run the program called Joy3.txt. This program will display the joystick information, as well as the keypad key numbers 1-12.

Button Hookup

One pin of all the buttons is tied to Vss as shown in Schematic 5. The other pin is connected to an I/O port on the



SCHEMATIC 3



SCHEMATIC 4

PARTS LIST

AVAILABLE FROM KRONOS ROBOTICS
- DiosPro chip (#16148)
- Dios Workboard Deluxe (#16452)
- DiosCompiler (a free download from the Kronos website)
- 4x16 Serial LCD (#16531)
- 36-pin Snapable Female Header #16291 (used to make header sockets)
- Nine-pin serial cable #16259 (needed to program the DiosPro)
- Six cell battery pack (#16321)
- 9V battery clip (#16264)

AVAILABLE FROM JAMECO ELECTRONICS
- Eight 3/8" #4 Machine Screws (#40969)
- Four #4 Washers (#106826)
- Eight #4, 1" Standoffs F/F (#139206)
- Six #4, 1" Standoffs M/F (#139231)
- 50 #4, 1/2" machine screws (#106810)
- 50 #4, washers (#211131)

AVAILABLE FROM ALL ELECTRONICS
- Two six-channel transmitter rReceivers (#JS-6)
- Six push buttons (#PB-138)
- Keypad (#KP-28)

OTHER
- Two 8x10 1/8" Plexiglass (or similar) (Available at most home centers)

WEBSITES
Jameco Electronics
www.jameco.com

All Electronics
www.allelectronics.com

Kronos Robotics
www.kronosrobotics.com

DiosPro. The ports are held high with a 10K resistor. When the button is pressed, the corresponding I/O port will go from a high state to a low state.

First, solder and connect the two

buttons mounted on the top of the base. Then connect wires to the four buttons on the lower base. I used 10" wires held together with small tie wraps. This will give you a good deal of room to wire the lower base to the mounted components on the upper base.

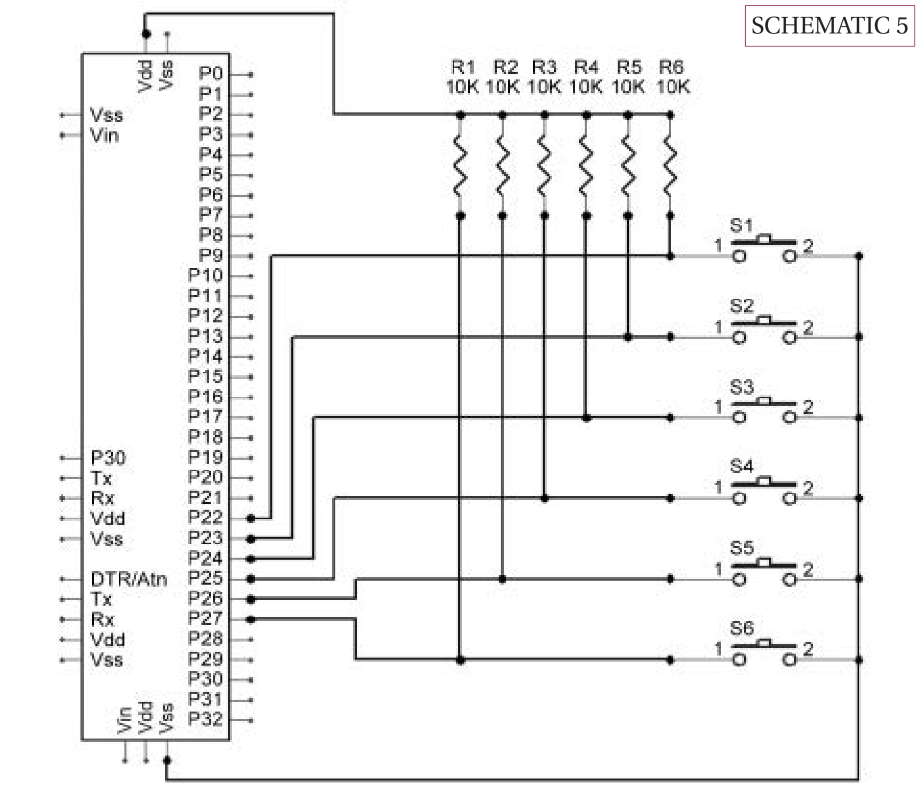
Load and run the program called Joy4.txt. This program will display the joystick, keypad, and button information on the LCD. Push each button to make sure the display goes from 1 when idle to 0 when pressed.

What's Next?

I had to cram quite a bit into this article. Next month, I will add a bit of software to create a tethered serial interface that can be used to talk to robots. Then we will add a ZigBee interface to complete our remote.

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

SCHEMATIC 5



Ultra-Sonic Microsensor Comparison Project

by Kyle Haston, University of Arizona, ECE Undergraduate

Microsensors have proven their usability in many ways including object detection applications such as automatic doors, ranging applications such as the rear bumper of modern vehicles, and educational applications such as microrobotics.

There are many different microsensors on the market today featuring a spectrum of possibly confusing characteristics. The recipients of these products are often aspiring microrobotics hobbyists and may be unaware of the differences between the many sensors. Also, manufacturers usually boast solely on the positive features of their sensor without explaining the resulting detriments. For example, a sensor with high gain may be able to see a long distance, but this is usually accompanied by a unique beam pattern. Whether the sensor is actually better than any others depends also on the user's application.

The many tests performed to evaluate sensors are usually conducted differently. It is easy to conduct a test or represent data slightly differently to give one sensor an apparent advantage over others.

The purpose of my project was to conduct a series of tests on the major microsensors available today and find out how they actually compare. The candidates for the test are as follows: SRF02, SRF04, SRF08, SRF10 from Devantech, Maxsonar-EZ1 from MaxBotix, and PING))) from Parallax. The SRF04, SRF08, and SRF10 sensors from Daventech and the PING)))™ sensor from Parallax are most commonly used in microrobotics applications such as data acquisition/logging or robot Sumo. The MaxBotix sensor is much smaller in size and is one of the newest sensors on the market. The SRF02 was recently released by Devantech and is similar in size and shape to the MaxBotix sensor. The websites for these three manufacturers are **www.robot-electronics.co.uk**, **www.maxbotix.com**, and **www.parallax.com**, respectively.

The Tests

Most microsensors require a microcontroller to switch them on

FIGURE 1. Beam patterns for the PING))) sensor.

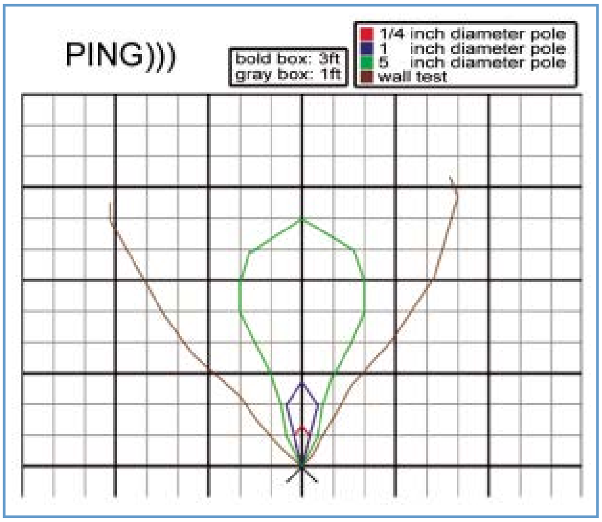
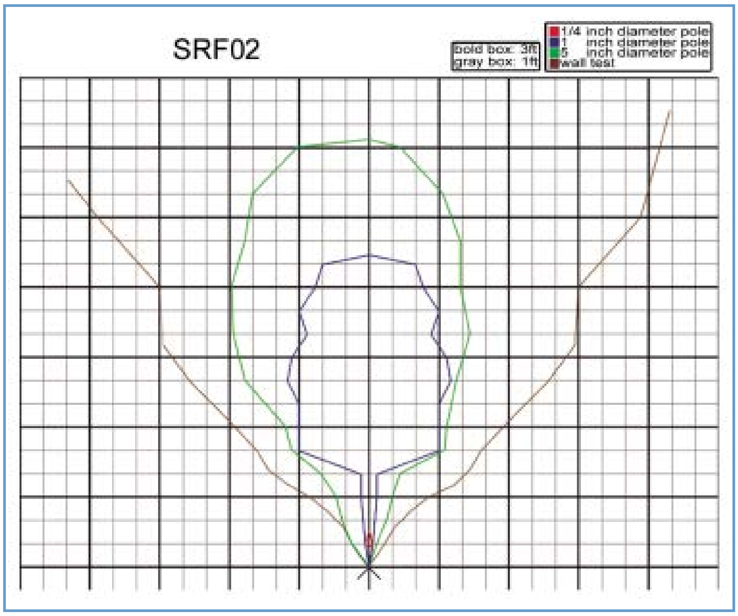


FIGURE 2. Beam patterns for the SRF02 sensor.



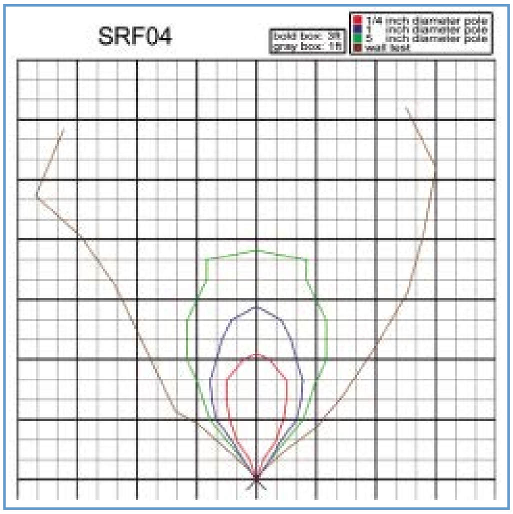


FIGURE 3. Beam patterns for the SRF04 sensor.

and off and interpret the readings that they output. The test setups — including which microcontrollers were used with which microsensors — and code examples that were used can be found at my website at **www.ece.arizona.edu/~khaston/**

The tests used to compare the sensors determine the beam pattern, accuracy of readout for objects in motion, and current draw for each sensor. The results for each sensor are described below.

Beam Patterns

The beam pattern of a sensor is usually very important in determining whether or not a certain sensor will be useful in the intended application. Symmetry, smoothness, size, shape, etc., should all be considered. It should also be noted that the beam pattern of a sensor changes depending on characteristics of the object being sensed. These characteristics include size, shape, and texture. For this test, we will use wooden dowels of two diameters: 1/4 inch, one inch, and a plastic PVC pipe with a five inch diameter. The beam pattern from a sensor actually exists in three dimensions. While most representations are shown on a two-dimensional plane, it is generally understood that such a pattern is lathed 180 degrees about the approximate axis of symmetry.

Another way to generate a beam pattern for a sensor is to place the sensor a fixed distance from a wall and test the maximum and minimum degrees that one can rotate the sensor while still having the sensor 'see' the wall. In this instance, textural characteristics of the wall also affect the results.

The previously stated

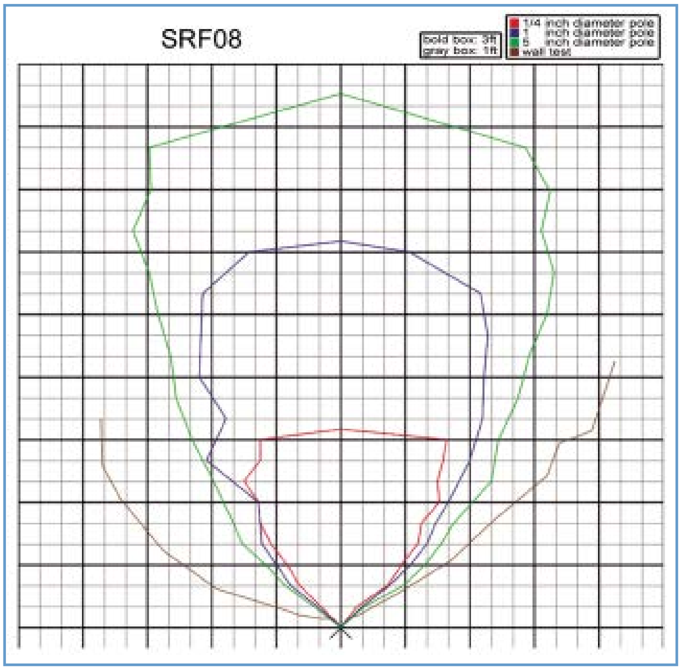


FIGURE 4. Beam patterns for the SRF08 sensor.

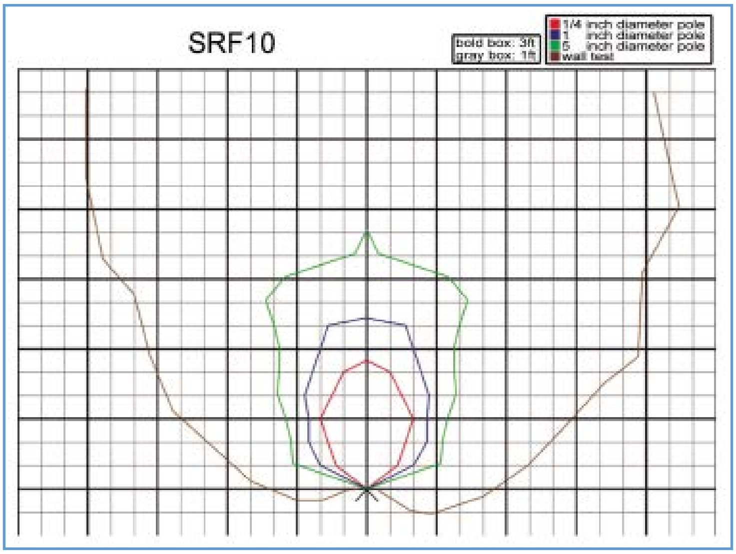


FIGURE 5. Beam patterns for the SRF10 sensor.

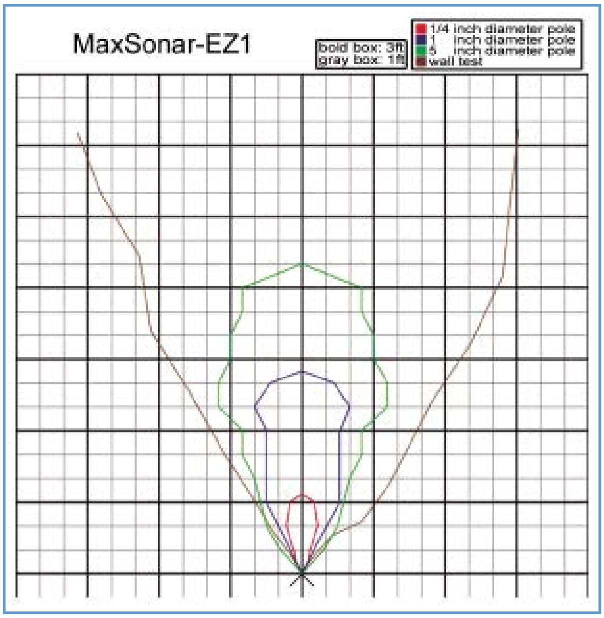


FIGURE 6. Beam patterns for the MaxSonar-EZ1 sensor.

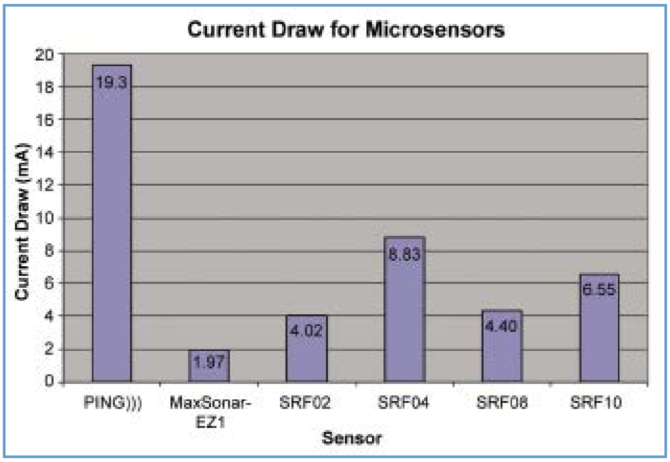


FIGURE 7. Current draw comparison chart for microsensors.

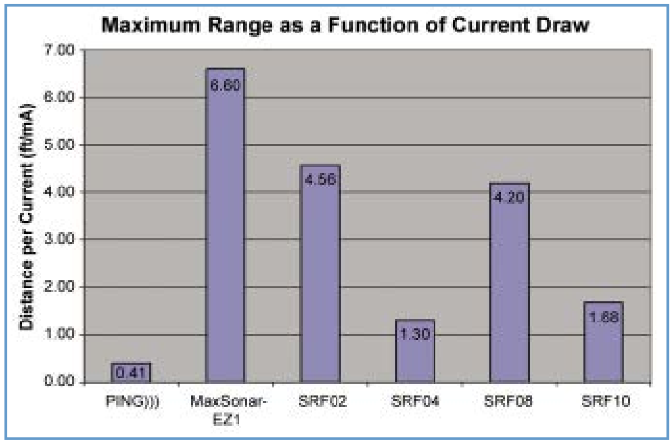
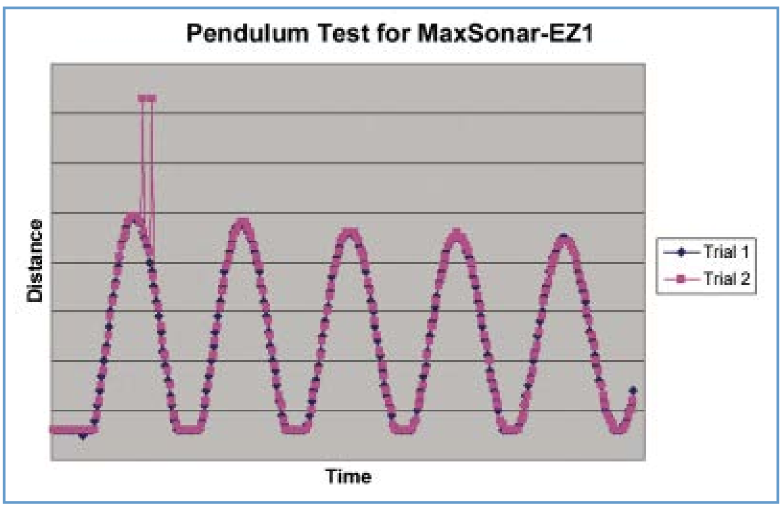


FIGURE 8. Comparison chart of Distance per Current Consumption for microsensors.

tests were performed for all six microsensors. The results were then graphed in overlay format to produce a single beam pattern sheet for each sensor as seen in Figures 1-6.

FIGURE 9. Pendulum test results for the MaxSonar-EZ1 microsensor.



Current Draw

Current draw is another feature of microsensors that is usually considered important. To perform this test, one must simply supply power to a microsensor in series with an ammeter. The results are shown in Figure 7.

One feature that is often tweaked by the manufacturer to achieve a larger beam pattern is called the gain. By increasing gain, oftentimes sensors can see further. Notorious side effects of this include non-smooth beam patterns, reading instability, and higher power consumption. Figure 8 shows a chart comparing the sensors with both power consumption and distance considered. The distance used for each sensor is the maximum reading achieved for the five inch diameter PVC pipe. This value was chosen because it represents the farthest the sensors could see.

Pendulum Test

Some sensors give accurate distance readings for objects moving to and from the sensor. Others will suddenly jump to a reading of the sensor's maximum range or zero. Stable and accurate readings in this setup are necessary for some user applications, so included among the tests is one of this nature.

An easy way to test whether a sensor performs well in this scenario is to mount the sensor somewhere and swing a pendulum of some sort in front of the sensor. The pendulum needs to swing to and from the sensor to produce an object with a distance that varies over time. If the results are captured via serial port and plotted, the result should be a sinusoidal wave of gradually decreasing amplitude. Sudden spikes to obviously inaccurate

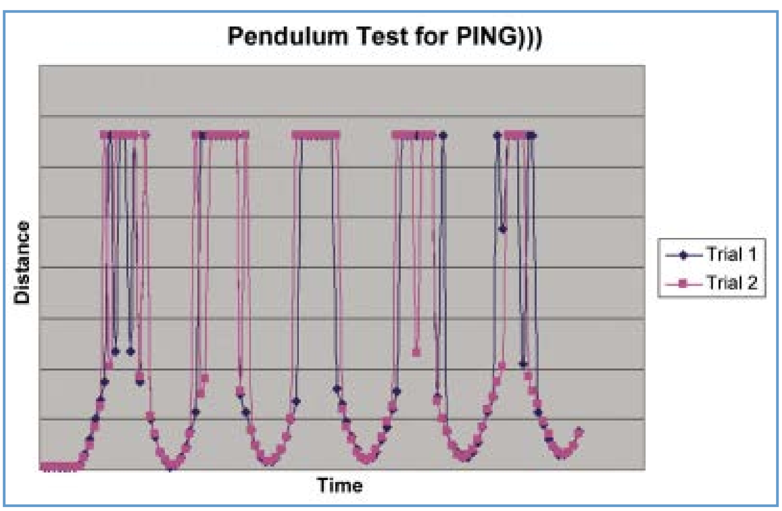


FIGURE 10. Pendulum test results for the PING))) microsensor.

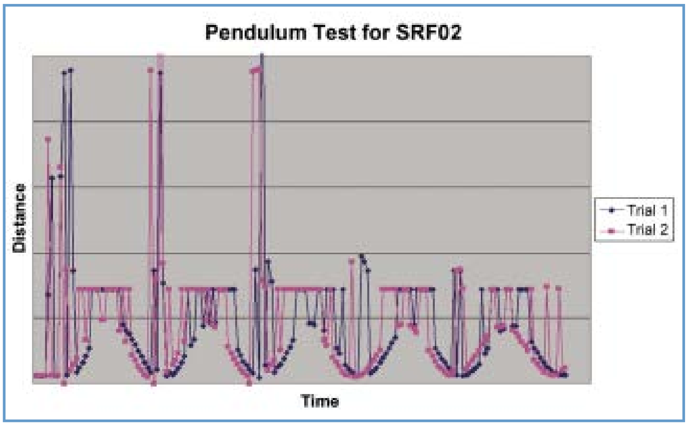


FIGURE 11. Pendulum test results for the SRF02 microsensor.

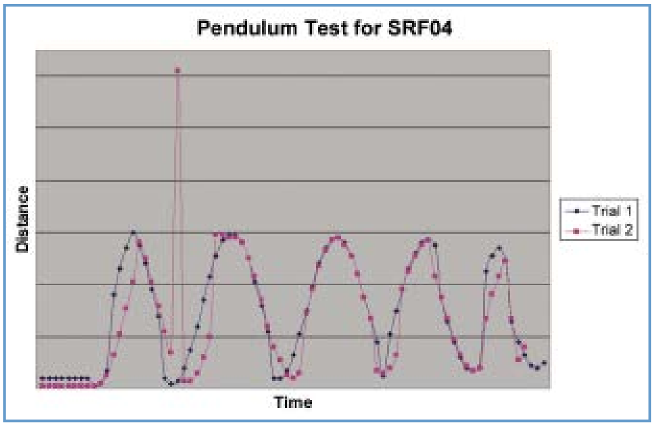


FIGURE 12. Pendulum test results for the SRF04 microsensor.

distance values indicate potential inadequacies depending on the intended application.

The units of the time axis has been omitted and the number of points plotted varies between sensors, because various microcontrollers take different amounts of time to output a distance reading. The number of readings per second is also influenced by the code used to program the microcontroller that controls the sensor. The units for the distance axis have also been omitted because some sensors output data that requires calibration, and it is the sinusoidal resemblance of the data that reveals performance more than actual values. The results of two trials for each sensor are shown in overlay format in Figures 9-14.

Understanding the Data

There are many legitimate conclusions that can be drawn from the results of this data, but it is almost impossible to determine which sensor is best. This is due to the fact that certain sensors will be more advantageous in some applications and less so in others. The purpose of this project was to compare all chosen sensors equally. The results of the comparisons and intended use of the microsensor will determine the rest. **SV**

FIGURE 13. Pendulum test results for the SRF08 microsensor.

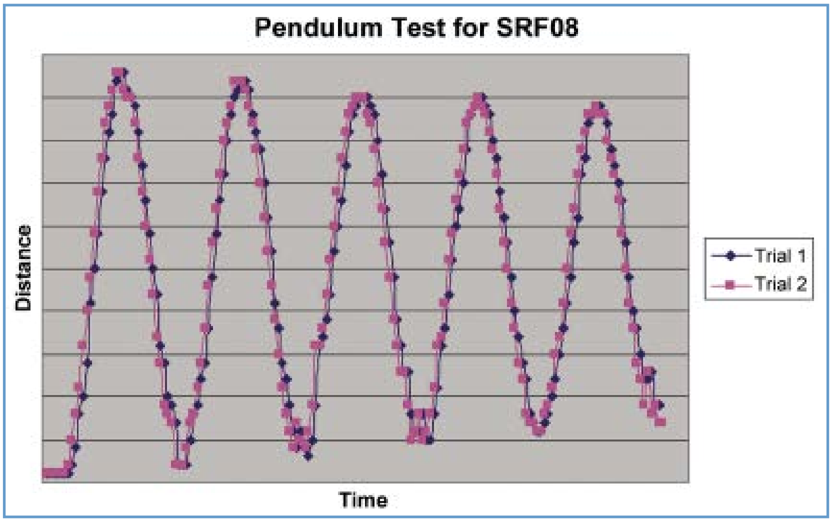
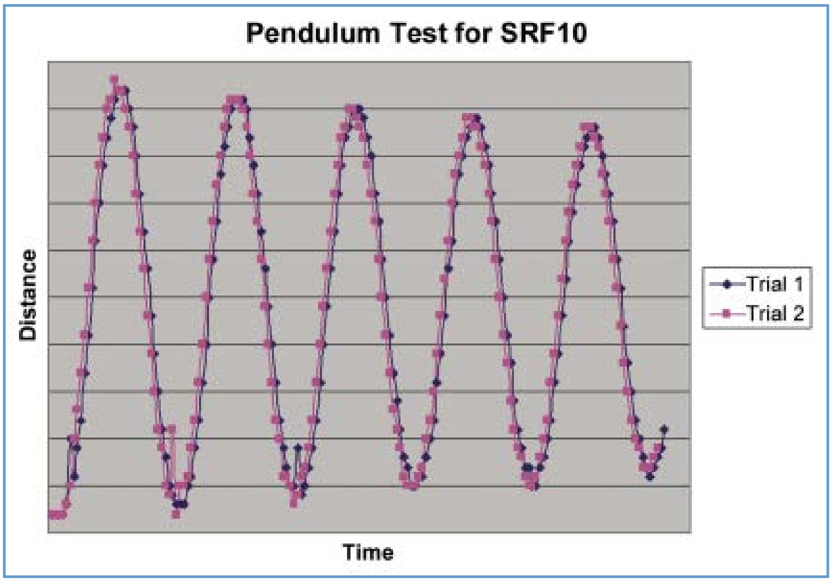


FIGURE 14. Pendulum test results for the SRF10 microsensor.



FOR YOUR INFO

More on this project can be found at my website at **www.ece.arizona.edu/~khaston/**.

There is a considerable amount of extra and useful information currently available on the website with more to come soon.

Mind Candy For Today's Roboticist

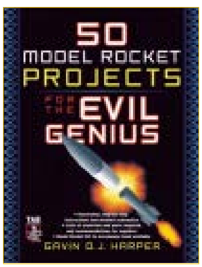
The *SERVO* Store

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

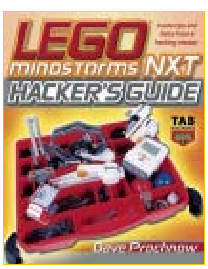
by Gavin D. J. Harper



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



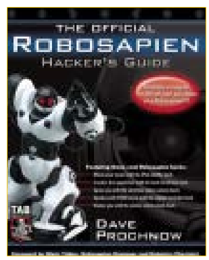
Here is an awesome next-generation collection of LEGO MINDSTORMS projects that enables you to build and program a real working robot in just 30 minutes! New technologies and expanded sensor capabilities make it easier than ever to add a level of sophistication to robotic and architectural creations. 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**

Mechanisms and Mechanical Devices Sourcebook

by Neil Sclater / Nicholas Chironis

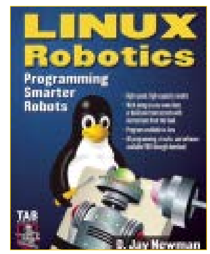
The fourth edition of this invention-inspiring engineering resource covers the past, present, and future of mechanisms and mechanical devices. You'll find drawings and descriptions of more than 2,000 components that have proven themselves over time and can be incorporated into the very latest mechanical, electromechanical, and mechatronic products and systems. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology, along with tutorial chapters on the basics of mechanisms and motion control, will bring you up-to-speed quickly on these cutting-edge topics. **$19.95**

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

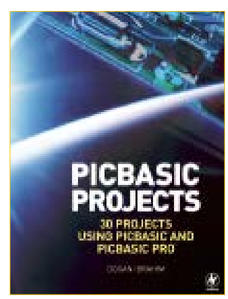
by D. Jay Newman



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**

PIC Basic Projects

by Dogan Ibrahim



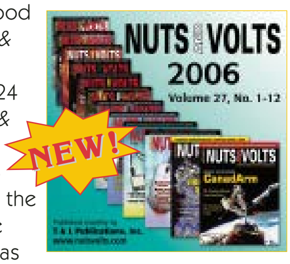
Covering the PIC BASIC and PIC BASIC PRO compilers, *PIC Basic Projects* provides an easy-to-use toolkit for developing applications with PIC BASIC. Numerous simple projects give clear and concrete examples of how PIC BASIC can be used to develop electronics applications, while larger and more advanced 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.

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

Mind Candy For Today's Roboticist

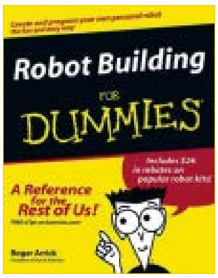
Nuts & Volts CD-ROM



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

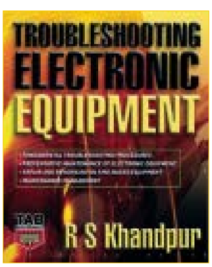
by Roger Arrick / Nancy Stevenson



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**

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**

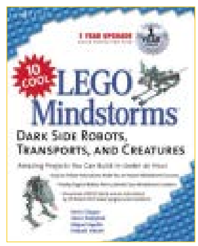
Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

Build Your Own All-Terrain Robot

by Brad Graham / Kathy McGowan

Remotely operated robots are becoming increasingly popular because they allow the operators to explore areas that may not normally be easily accessible. The use of video-controlled technology has sparked a growing public interest not just in hobbyists, but also in the areas of research, space, archeology, deep-sea exploration, and even the military. This book has the ideal mix of "brains and brawn," making it appealing to hobbyists and interested professionals alike. **$29.95**

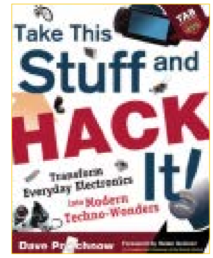
10 Cool LEGO Mindstorm Dark Side Robots, Transports, and Creatures



The Dark Side Developer's Kit is targeted towards the young or novice LEGO MINDSTORMS designer, age 9 and up, although experienced MINDSTORMS fans will appreciate the unique possibilities offered by this kit, as well. *The Dark Side Developer's Kit* includes special MINDSTORMS pieces that allow the user to create a host of Star Wars themed robots, creatures, and vehicles. It also comes with the Micro Scout, a mini-computer with seven built-in programs, a motor, and a light sensor that brings the MINDSTORMS creations to life with a minimum of effort. **$24.95**

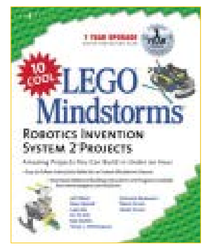
Take This Stuff and Hack It!

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**

10 Cool LEGO Mindstorm Robotics Invention System 2 Projects



The LEGO MINDSTORMS RIS 2.0 is the core set for all MINDSTORMS users, that lets users design and program working robots — limited only by their imagination. Initially designed for users 12 and up, LEGO MINDSTORMS has taken off with LEGO enthusiasts of all ages. *10 Cool LEGO MINDSTORMS RIS 2.0 Projects: Amazing Projects You Can Build in Under an Hour* provides step-by-step instructions and detailed illustrations for users of all skill levels and proficiencies. **$24.95**

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.

by Dave Calkins

ROBOGames Prep:

Sumo Robots — The Easiest and Hardest Event There Is

So you want to build a competition robot? Well, the easiest kind of robot — and the event in which most competition builders get started — is Sumo robots. Sumo robotics is like chess. The basics are very simple, yet players vary wildly in how good they do when competing — even though both players know the same rules.

What is a Sumo robot? Well, there are six weight classes: 25 g, 100 g, 500 g, 1 kg LEGO, 3 kg, and 3 kg humanoid. The 3 kg "cake-boxes" and 3 kg humanoids can be either remote controlled or autonomous (though the R/C bots don't compete against the autonomous robots — they have separate divisions). The other classes are all autonomous only — the robot has to think for itself.

Sumo robot competitions are the oldest robot competitions in the world, having started in the 1970s (!) in Japan. The concept is the same as human Sumo wrestling — two opponents face off on a "Dohyo" (a circular competition ring with two starting lines called shikiri lines). When the match starts, the contestants try to push each other out of the ring. When one contestant goes out of the ring or touches the floor with any part of his body other than the feet, he loses. Even if he went out of the ring by himself.

Simplicity in Itself

Except when it's not. Because while the rules are simple, the various strategies (such as circling around the other guy and push from behind), tricks (use better servos), and tactics (replaceable blades on the front scoop to make sure you get under the other guy) make the matches exciting and unpredictable. Just like combat robots, the designs of Sumo robots vary wildly. Unlike combat robots, most matches are over in about 10 seconds, robots have no weapons, and almost all the robots are autonomous.

So, Let's Get Building a Sumo Robot!

If you want to compete in the 3 kg remote control category, basically all you have to do is build a powerful, 20 cm (7.87") long by 20 cm wide robot that can push anything else about three feet. No weapons. Just raw pushing power. The closest thing to a weapon you can have is a perfectly flat wedge along the front of your robot so you can get under the other robot. Of course, the other driver will have his own ideas about avoiding you.

But the real fun of building a Sumo robot is in building an autonomous robot. Most people start with the 500 g class. The robots need to be no more than 10 cm (3.94") long by 10 cm wide.

The 500 g and 3 kg Sumo robots can be any height however, which is a huge bonus if you work it right. Because once the match starts, the robot can fall over, and thus be longer that its initial size. I've also seen some really cool robots that start out as a cube, and then expand along a diagonal axis, so they end up as a long wedge.

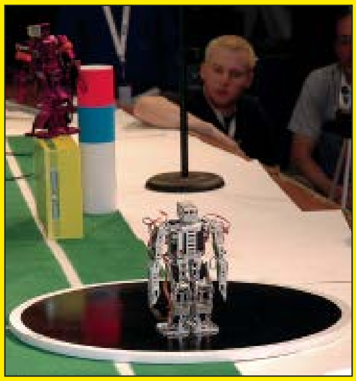
The challenge in a good autonomous robot isn't just being able to push the other robot, but also being able to find the other robot, while not driving itself out of the ring. All Sumo robots have some method of detecting the edge of the ring. The dohyo itself is all black, except for a thin white lining along the edge. Most robots use IR sensors or photo-resistors to detect the white line along the edge, so that the bot knows to turn around once it sees the edge of the ring (you'll have to program it to react once it sees the white line).

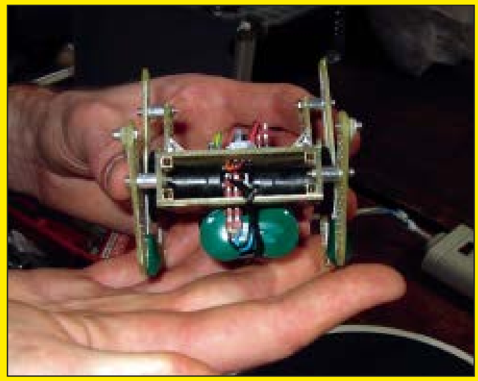
But that's only half the battle. While there are some Sumo robots out there that wander around the arena turning when they find the edge, they only win by chance — the robot happens to bump into the other robot. What we need is some sensors!

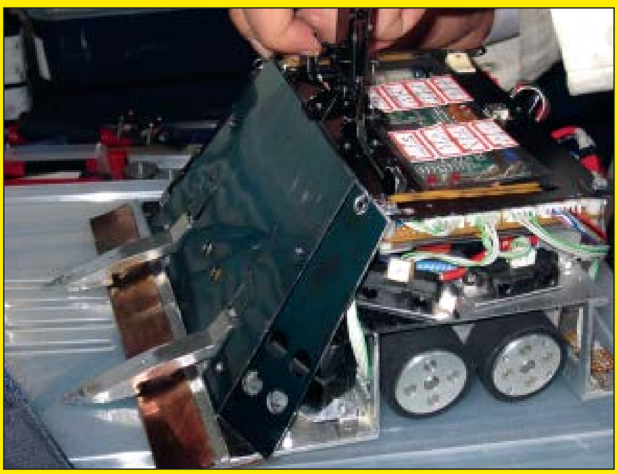
You can use infrared, sonar, laser, photo-resistors, video cameras — just about anything! And that's the biggest challenge — being able to see the other guy, and maneuver around the other bot before pushing it out of the dohyo and winning!

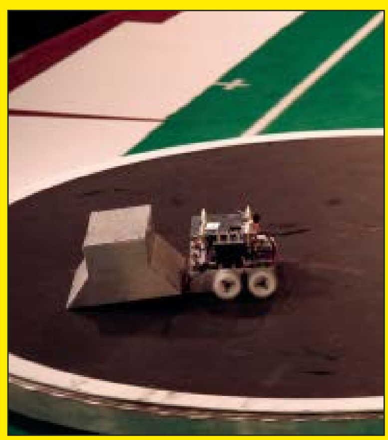
One of the easiest and most effective ways to win at 500 g Sumo is to have the most powerful servos you can afford. These will have to be continuous rotation servos, which you can modify from regular servos, or buy (ServoCity will modify any servo to continuous rotation for a fee.) I once spent weeks building the "perfect" sumo,

> “Sumo robotics is like chess. The basics are very simple, yet players vary wildly in how good they do when competing — even though both players know the same rules.”









and got slaughtered by a 12-year-old who simply used better servos than I did. You can also "boost" a servo by overvolting it. I've overvolted servos up to 12 VDC. This absolutely will destroy the servo after a few matches, but you quadruple the power every time you double the voltage. So while you'll end up throwing out the servos, it's still a pretty easy way to boost your power.

Modifying a servo to continuous rotation would take a whole article. But the basics are that you replace the potentiometer with a few 10K ohm resisters, and snip off the mechanical stop so that the gears can rotate 360 degrees. It's more complex than this, but there are lots of articles on the net that will give you a step-by-step guide in servo modification.

If you're building your first Sumo robot, start with either the 500 g or 3 kg class. The 3 kg class gives you a LOT of flexibility in terms of power and the kinds of sensors you use. You can stick a lot of batteries and much bigger gearmotors on a 3 kg, although the cost will be higher than on a 500 g Sumo. Most first-timers build 500 g robots to save money while they learn the basics. If you understand basic microcontrollers (such as a BASIC Stamp, PIC, or ATMega) you can build a pretty good 500 g Sumo for under $100.

You can also buy a kit and run it as-is, or modify it. There are lots of kits you can buy that are competitive out of the box — Parallax, Lynxmotion, JCM inVentures, OWI, Solarbotics, and several other companies all make very good Sumo robot kits. Of course, if everyone has the same kit, it's just chance as to who wins. So you should upgrade!

One of the biggest disappointments in international Sumo robot competitions is the dearth of US competitors. At RoboGames, Singapore and Japan have swept the 3 kg classes and taken 1st in 500 g every year since we started. Yet, it's really not that hard to build a competitive robot. A few Dewalt 24V drills could easily get you a gold medal if you spent the time!

Carlo Bertocchini — famous for his heavyweight combat robot "Biohazard" — is also the reigning US champion of 3 kg sumo robots with his bot, "The Beast." Many of the new 3 kg robots coming out of Asia aren't much better than The Beast. You could be the person to take the gold medal at RoboGames 2007 — all you have to do is try!

Beyond RoboGames, Sumo robots are the most popular autonomous robot class in the world. You'll find events and competitions for Sumo bots all around the country — Seattle, Portland, Dallas, Chicago, and many other cities all have Sumo events.

Autonomous Sumo robots are absolutely the single best way to get involved in robots. No matter where you live or what your goals are, you should build a Sumo. It will teach you everything you need to know about the basics of robotics — sensors, pulse width modulation and motor control, gearing, voltage-to-torque ratios, microprocessor control, simple mechanical engineering, and robot construction and assembly.

Sumo robots are how I got involved in robots. If it weren't for Sumos, there would be no RoboGames /ROBOlympics. I wouldn't be teaching robotics at SFSU. I probably wouldn't be involved in robots at all.

Go build a Sumo. You'll never regret it. The first hit's free ... **SV**

SUMO ROBOT RULES

- All matches are "best-of-three." Robots fight three times and the winner of two matches goes on to the next round.

- 500 g, 1 kg LEGO, and 3 kg robots can have any starting height, or they can "grow" by flipping open or expanding once the match starts. Before the match, they must fit into the size cube relative to their class. 25 g and 100 g class robots cannot expand.

- Mini (500 g), Micro (100 g), and Nano (25 g) robots must be autonomous. Any method of control may be used, as long as it is fully contained within the robot. Robot operation must begin automatically no less than five seconds after being started by the user. 3 kg Sumo and Humanoid class robots can be remotely controlled.

- Autonomous robots must not start operating for a minimum of five seconds after initiation by the user.

- Dohyo rings for 3 kg robots are 154 cm in diameter. 500 g and 1 kg LEGO doyhos are 77 cm in diameter; 100 g doyhos are 38.5 cm in diameter; and 25 g doyhos are 19.25 cm in diameter.

- Jamming devices — such as IR LEDs intended to saturate the opponents IR sensors — are not allowed.

- Parts that could break or damage the ring are not allowed. Do not use parts that are intended to damage the opponent's robot or its operator. Normal pushes and bangs are not considered intent to damage.

- Devices that can store liquid, powder, gas, flame, or other substances for throwing at the opponent are not allowed.

- Devices that throw things at your opponent are not allowed.

- Sticky substances to improve traction are not allowed. Tires and other components of the robot in contact with the ring must not be able to pick up and hold a standard 3" x 5" index card for more than two seconds.

- Devices to increase down force, such as a vacuum pump or magnets, are only allowed in the 3 kg class. They are not allowed in all other classes.

- All edges — including, but not limited to the front scoop — must not be sharp enough to scratch or damage the ring, other robots, or players.

- One match will be fought for a total of three minutes, starting and ending upon the judge's command. The clock shall start ticking five seconds after the start is announced.

ROBOLYMPICS

www.robolympics.net

San Francisco, California - June 15-17 2007

Robots from around the world invade San Francisco for the Third Annual International ROBOlympics!

Join us in 2007 for the only international combined-event robot competition. Compete in any of 60 different competitions, including no-cost junior league events (for kids 17 and under.) At the 2006 event, over 650 engineers from 13 different countries competed with more than 450 robots! No matter which event you compete in, you'll still meet hundreds of robot builders from around the world, see new robots you never knew existed, and learn more about robots than you imagined possible!

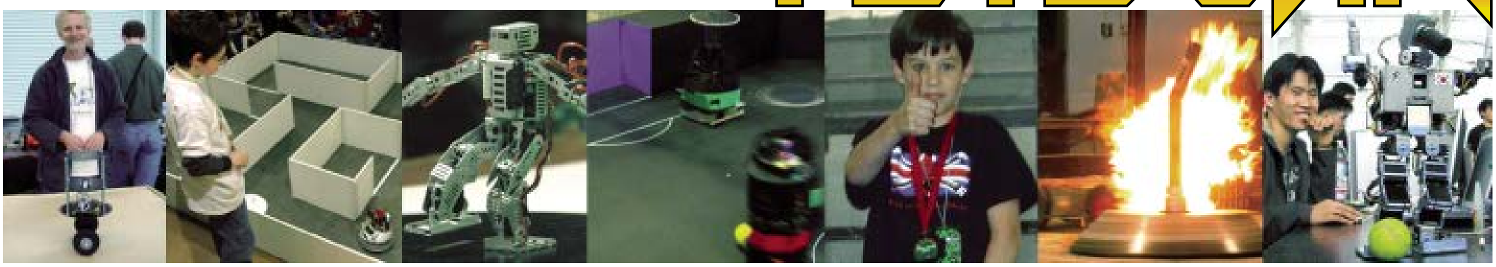
Don't miss this show!

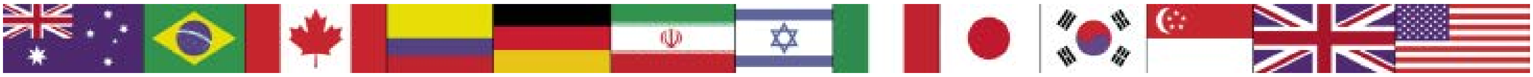
Sponsorship opportunities are still available for the 3nd Int'l ROBOlympics! Take advantage of the tremendous publicity potential! Robotics is becoming the new lifestyle sport of the thinking age. The Renaissance had music, we have DC motors. ROBOlympics is the channel to a growing audience with both a voice and a high disposable income. Contact us for info on this exciting opportunity:
sponsors@robogames.net

Events at ROBOlympics:

Combat: 150g-340lbs and 2 autonomus classes;
Robot Soccer: Humanoid, MiroSot, NanoSot, RoboSot, Aibo, Small League, Middle League;
Sumo: 25g - 3kg;
Open: Robot Triathlon, Ribbon Climber, Line Slalom, Mindstorms Challenge, Best of Show, Maze Solving, Aibo Performer, Balancer Race, Fire Fighting, Mindstorms Open, Table Top Navigation, Vex Open, Vex Challenge, Biped Race, Walker Challenge, Robomagellan
BEAM: Speeder, Photovore, Robosapien Hacker
Junior League: Lego - Woots and Snarks, 120 lb Combat (Jr), Mindstorms Challenge, Best of Show, 500 g Sumo, Handy Board Ball, Mindstorms Open, BasketBall Challenge, Vex Open, Vex Challenge
Robo-One: Wrestling, Agility, Stair Climbing, Door Opening, The Eagle, Toss
Art Bots: Musical, Static, Kinetic, Painting, Bartending
Bot Hockey: 3kg & 12 lb
and of course:

TETSUJIN





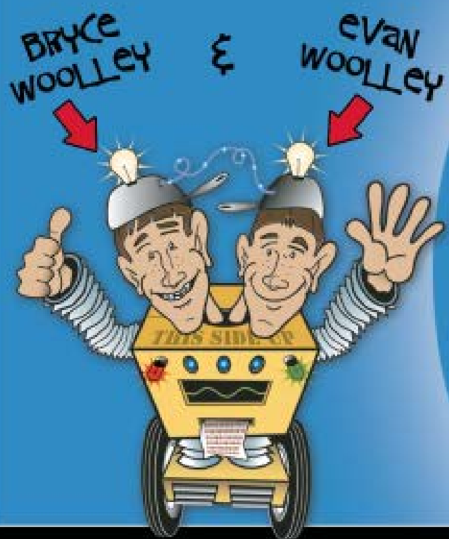
The Third International ROBOlympics is sponsored by

Robotics Society of America

COMBOTS

SERVO MAGAZINE

Twin Tweaks



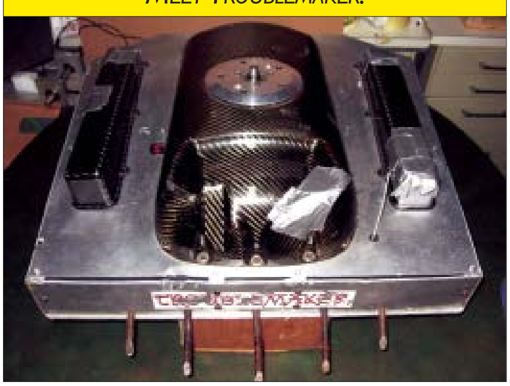
THIS MONTH:

Rummaging in the Robot Reliquary

Usually we're given a kit to work with for our column, but such is not always the case. So, left to our own devices, we had to figure out a good project for our next column. One of our initial ideas was to make a simple robot out of scrap we had in our garage. A quick survey of Robot Central reveals that it is not your typical garage. A cursory look will reveal solar powered boats hanging from the ceiling, shells of FIRST robots past, and copious amounts of aluminum strewn about, courtesy of our dad's job at Cosworth Racing. There was certainly enough there with which to make a robot — Maxon motors, Colson wheels, carbon fiber plenum tops (for cool factor, of course), and charred Vantec speed controllers. Upon pawing through some other possible robot bits, we uncovered a project that would be much more useful.

Troublemaker was our first major robotics project — a 60 lb combat robot. In its present condition in the garage, Troublemaker was inoperable. It wasn't because of a fatal skirmish in competition as one might expect from a retired combat bot, but rather an upgrade gone awry. But that is at the end of the story, and the beginning is a much better place to start.

MEET TROUBLEMAKER.



Once Upon a Time

It is a difficult thing to nail down exactly when our interest in robotics began. Maybe it was playing with Z-Bots as small children. Maybe it was experimenting with LEGOs and eventually LEGO Mindstorms. But what we generally label as our first major foray into the world of robotics was in the arena of combat robots. Combat robotics might seem like an odd choice for an introduction to robotics, but after watching shows like Battlebots and pitting the Z-Bots and Voids against each other in numerous epic battles, it was certainly an area of robotics that had captured our interest.

When we were just wee middle schoolers, we had the opportunity to apply for a $500 grant from CAG (California Association for the Gifted) to fund an educational project. Robots looked like a lot of fun that would incidentally be hugely educational, so we applied for the grant to build a combat robot. We received the grant, and we were on our way to becoming real roboticists.

How do middle schoolers build a 60 pound combat robot anyway? Thankfully, we had two great resources: our dad, who works at Cosworth, and that wonderful tool called the Internet. Robotics resources abound on the web, and our dad was able to salvage scrapped aluminum and motors from his work to give us something to work with. One of the most inspiring pieces we received courtesy of Cosworth was an obsolete plenum top; an air intake manifold from one of Cosworth's racecar engines. The plenum top was made from super cool carbon fiber, and the compelling shape galvanized our imaginations.

With Background Music, This Would Make a Good Robot Building Montage

The plenum top had a circular hole cut out from its roof, and we were immediately inspired to fill that gap. Our minds leapt to the most logical conclusion — when you have a combat robot with a hole in the top, what do you put in it? A spinning kinetic weapon, of course! Many ideas were tossed around and contemplated. Everything from Active Threat Detection to Tesla coils was addressed, but we had to fit our plan to the constraints of the rules of the competition and of practicality. Ideas were jotted down and refined on napkins from a local pizza place, and schemes developed over the pizza were refined in the garage.

Designs were refined from napkin to computer. We wanted an interesting robot, but a spinning weapon on a box was not quite up to our expectations of cool factor. Instead of a simple square, we angled off the back corners to come up with something more unique. Drawings were made on AutoCAD with the eventual goal of getting our creation machined.

We decided the ground floor of the robot was a good place to start, and we wanted to make the physical construction of the robot as easy as possible. Our goal was to make a floor that we could simply bolt everything onto, but to do that we basically had to design everything first. We had to know exactly what we wanted in the robot, and where to put it. Back to the pizza place and the napkins.

A pleasant discovery it was when we found out that robots didn't really require everything from RadioShack and a kitchen sink. All we needed were batteries, a radio transmitter, speed controllers, an antenna, a kill switch, motors, wheels, a solenoid, a junction block, and mounting brackets. It may seem like a long list, but those really are the only necessities we needed for our robot. Things like solenoids aren't required for robot construction, but we thought it was a handy way to turn on and off our spinning weapon. We used big, heavy sealed lead acid batteries to power our bot, with the unpleasantness of the weight and size offset by the fact that they, too, came courtesy of Cosworth. We spent our grant money in purchasing our Vantec speed controller and Futaba radio, and with that, we had all of our essentials. It was time to get down to real robot building.

A philosophy that we have alternately embraced and abandoned (for cool factor) has been the classic KISS — Keep It Simple Stupid. This being our first robot, we really did want to keep it simple, so we settled on a simple direct drive train with four motors and four wheels. After balancing simplicity with practicality, we powered our spinning weapon through a pulley and belt system, in anticipation of some heavy shocks. To give our robot a little extra bite, we outfitted it with a set of hardened chromoly spikes on the front, and with that we were ready for action.

Open the Door, You're There

Our original plan when we took the plunge into combat robotics was to actually compete in Battlebots. But the more we researched the competition, the more unmanageable of a commitment it seemed. After building an expensive robot, a costly trip to San Francisco was out of the question, so we had to look for other options. What we found was Botbash — a smaller competition located in Phoenix, AZ that still drew a lot of big names from the combat robotics establishment. It seemed like a perfect fit, so we packed the robot in the car and headed off to the Copper State.

Botbash 2001 was held in a abandoned warehouse affectionately and inscrutably known as "Automatic." The competition was a double elimination tournament, where each match would feature two robots going head-to-head in a fight to the death. Even before the matches started, we had an exciting opportunity to survey the other competitors, and everyone was more than happy to answer any questions about their robots. Botbash 2001 was also where we first met Dan Danknick, the esteemed former technical editor of *SERVO Magazine*.

Overall, Botbash 2001 gave us a very good first impression of the combat robotics community and of robotics in general. We were afforded the same respect as everyone else there even as young rookies, and we were amazed by the variety of designs at the competition. Baseball bats, spinning shells, and saw blades were all fair game, and we left the event with a galvanized imagination and a few battle scars.

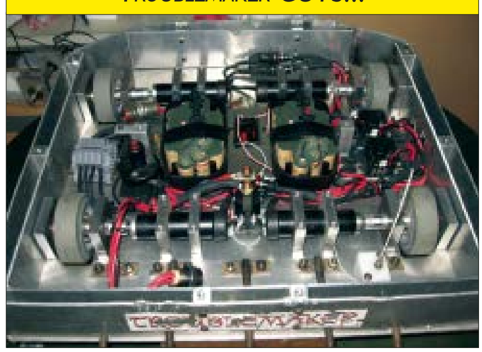
Causing More Trouble

Botbash 2001 had left Troublemaker with a few battle scars. Namely, the starter motor we used to power our kinetic spinner was completely smoked, filling the plenum top with the lingering odor of electrical death. We replaced the motor and we also replaced the heavy sealed lead acid batteries in Troublemaker with a lighter pair of cordless drill batteries. The extra weight the batteries afforded us went towards a second inner layer of 7075 T-6 aluminum armor. Troublemaker was ready to play again.

Almost. We found the thrill of competition so inspiring that we simply had to build another robot. The plenum top once again served as the nucleus of our design, and we created a 30 lb robot — Twibill Trouble — which we fashioned to look like the head of a double bladed battle axe. Twibill Trouble was ready just in time for our second combat robotics endeavor: Botbash 2002 in Tempe, AZ.

Botbash 2002 introduced some new aspects of competition. The matches were no longer one on one — two teams of two robots squared off against one

TROUBLEMAKER GUTS!!!



TWIBILL TROUBLE.

another in each match. Also, the objective was no longer simply to destroy the opposition. Botbash 2002 introduced Killball, where teams would win the match by scoring the most goals with an unevenly weighted ball. The second event was Elimination, where teams would participate in the traditional fight to the death. The arena was also much more interactive. Instead of a barren steel floor, this arena featured a swinging pendulum in the middle and two pits on either end of the field. Goals for Killball were also placed at either end of the field.

Troublemaker had an interesting time at Botbash 2002. Our new motor for our kinetic spinner was certainly up to the job, but it turned out that other parts of the design were unexpected weak points. In one of our first matches, we went up against a formidable opponent with a large, two toothed, spinning disk as a weapon. Our opponent actually cut the steel cable that fastened a weight to our spinner, and the weight went flying. A particularly interesting note is that seconds later, the Killball match was over because the Killball had somehow made it into our team's goal. Upon further inspection, it was revealed that the Killball had become seriously dented. We have no proof, but one theory we have is that the disembodied weight actually hit the Killball into our own goal. Apparently Troublemaker was quite the pool shark, though it was unfortunate that we had to sacrifice our spinner to find that out.

Fortunately, we still had our chromoly spikes, and Troublemaker was still ready to brawl. In a later match, however, Troublemaker inexplicably died in the middle of the melee. The autopsy revealed that we had smoked our Vantec speed controller. Troublemaker was down for the count, but the matches that we did play were certainly a great time.

Twibill Trouble remained operational throughout the event, and even though it often exited matches with bent blades or scraped up carbon fiber, it certainly proved itself to be one scrappy little bot. Twibill Trouble, however, was eventually eliminated. But even though we didn't leave Botbash 2002 with awards, we left with the memories of another great and inspiring experience.

What We Have Here is a Failure to Communicate

On the drive home from Botbash 2002, we were already thinking of upgrades we wanted to make for Botbash 2003. Number one on the list for Troublemaker was a new speed controller to replace our trashed Vantec, and we were also dreaming of some more protection for Twibill Trouble's battered plenum top.

Rather than replacing our charred Vantec with another Vantec, we decided to equip Troublemaker with two Victor speed controllers instead. We had heard great things about the robustness of Victors at the Botbash events, so they seemed like a logical choice. We also outfitted Troublemaker with new steel cables for the weights on the spinner, and we shielded the cables with protective tubes. Twibill Trouble was equipped with a titanium guard that circled the lower perimeter of a new plenum top. Of course, these upgrades required extensive planning on more pizza place napkins, and the robots were finished on the usual timetable — a few late nights before the competition.

BOTBASH 2002.

Our usual timetable allotted the usual time for testing — the first hours of the competition. Twibill Trouble fired up smoothly, but when we went to turn on Troublemaker, the sound it made was like crickets in a cartoon. We were stunned that Troublemaker was totally non-operational. We were able to determine that the problem was rooted in our apparent upgrade; moving from our charred Vantec speed controller to two robust Victors.

The problem was that our Futaba radio was on PCM (pulse code modulation) frequency, and our new Victors apparently didn't speak that language. What we needed was an electronic interpreter to mediate between the Victors and the radio controller, and that interpreter takes the form of a mixer. We didn't anticipate such a problem and we had no mixer with us at the competition, but thanks to the generosity of Ted Shimoda (our neighbor in the pits), we were able to outfit Troublemaker with a mixer for the duration of the competition.

The Pit and the Flaming Pendulum

Botbash 2003 — also affectionately known as Botbash by the Sea — was a thrilling competition. As part of the Del Mar Fair, the event brought many more spectators than previous years, and the combat arena itself had some exciting upgrades. The pits and pendulum were carried over from the previous year's event, but the pendulum was now on fire. The events from the previous competition were also carried over, so we had the opportunity to play Killball and Elimination once again.

We enlisted some of the Club CREATE members from our high school to help us in the pits, but for much of the competition, we fortunately did not have too heavy of a workload beyond popping the tops off of the robots to extract batteries for charging. That, however, does not mean that Troublemaker and Twibill Trouble avoided battle scars.

Troublemaker's kinetic spinner met with more trouble — another weight went for an unscheduled flight after hitting the floor while TM was on uneven footing partially in a pit. Twibill Trouble's titanium guards did help in averting some damage, but the plenum top still met with the business end of some weapons. Both robots fared pretty well, and Troublemaker even secured second place after a tense one-on-one deathmatch.

THE FLAMING PENDULUM.

THE PIT.

Back to the Future

We had to return the loaned mixer at the end of Botbash 2003, so Troublemaker was again rendered non-operational. We thought that Troublemaker's catatonic state was not fitting for a robot that had just brawled its way to second place, so we ordered our own mixer. Alas, the frenzy of FIRST (For Inspiration and Recognition of Science and Technology) and other aspects of life's general business stole our attention from the deserving robot, and Troublemaker sat waiting in the garage, slowly being buried, for almost four years.

Now we return to the beginning of the article, where we've unearthed Troublemaker and gathered all of the requisite parts to bring it back to life. We bought our mixer from Team Delta, and the product came with comprehensive instructions that made installation a snap. We were able to scrounge up a few over-long PWM cables, allowing us to make the requisite connections to the speed controllers and the receiver module. After four years, Troublemaker was finally ready to play again.

More Than Meets the Eye

Combat robotics is often cast in an antagonistic light — the thought is that since the object of the competition is to destroy the other person's robot, there would naturally be an atmosphere of cutthroat adversarialism. Such is not the case. That might seem counterintuitive, given the apparent object of the competition. But is that really the object of the competition?

A closer examination of combat robotics events like Botbash reveals them to very much be kindred spirits with competitions like FIRST. Combat events, like FIRST, are all about creative ideas being born out in competition, with Botbash and FIRST simply being different types of competitions.

While the FIRST competition has the stated philosophy of Gracious Professionalism, participants in all different types of robotics competitions espouse an ideology of sportsmanship. Even though combat robots are pitted against each other in a fight to the death, competitors will not go overboard, and they will often help former opponents with repairs to their creation.

In competitions like Botbash, every competitor knows that it is an accomplishment simply to be there. It is an incredible challenge to take your idea from being just a dream to a concrete creation, and everyone at these competitions can relate to that grueling process. The sportsmanship and professionalism at these competitions also stems from respect. Every competitor respects their competition because everyone there is an innovator. Everyone from the intrepid garage tinkerer to the professional engineer have created something unique, and progress is simply made by a series of innovations.

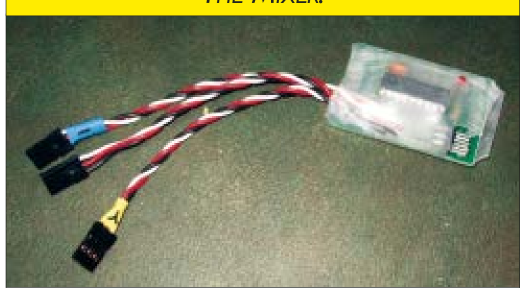
That really gets to the heart of these competitions. Why do these roboteers spend lots of time and money on remote controlled gizmos that simply beat the crap out of each other? Because it's not just about the end result (in this case, FUN!), but rather in the creative process. Competitions like Botbash are stimulating the exploration of new technology and new uses for existing technology. Perhaps it sounds overreaching, but competitions like these are helping to democratize technology.

Take the Plunge

When we dove into robotics, the driving force was really just the idea that we "wanted to build a robot." That really is all it takes; anyone can get involved in robotics if that is what they want to do. Of course, we greatly benefited from the mentorship we received from our dad, but that doesn't make robotics an exclusive pursuit. Let the wealth of information on the Internet be your guide, or volunteer at a local FIRST regional, or just get involved in some way, shape, or form.

The bottom line is that there are plenty of resources for anyone to get involved, no matter what their background is. And even though robotics may seem like an overly academic pursuit, competitions like Botbash are what make it a fun and cultivating thing to do. Even though Botbash events are no longer being held, maybe you'll see Troublemaker and Twibill Trouble at some other RFL melee. SV

THE MIXER.



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

BY GORDON MCCOMB

Fabulous Robots With Pre-Fab Parts!

Twenty five years ago, if you wanted a robot you built it from scratch.

Out to the garage you'd go, with sheet metal, wood, and plastic bits in hand. You'd cut, drill, sweat, and in the end you'd have a functional but perhaps amateur-looking robot to show to your friends and family. And that would be just for the mechanical body. The electronics and brain would be a project for the next year — if you made the effort at all.

Starting about 10-15 years ago, a few enterprising mail order businesses — such as Solarbotics and Lynxmotion — began offering robot construction kits. You could bypass the mechanical construction phase, and go straight to the electronics, programming, or even play phase! The kits were special-purpose, designed to construct a single version of the robot. To this day, there are still plenty of pre-made kits that make it easy to build a robot, letting you concentrate on the other aspects that you may find more enjoyable.

Now, the latest trend in amateur robot construction is prefabricated components that allow you to build a variety of designs, all from a single kit of parts. The idea is hardly new — for decades, kids have been building things with Erector and LEGO parts, even Lincoln Logs. While the idea of mixing and matching components to build custom designs is old-hat, only recently has the concept come to the field of educational and amateur robotics.

In this month's column, we'll take a closer look at some of the robot pre-fab construction kits that are available, and why such a kit may be just what you're looking for if you're wanting to experiment with robotics.

Vex Robotics Design System

The Vex system is based around the Erector set style of pre-drilled stamped metal girders and connector pieces. To build a robot, you fasten the girders and other parts together using machine screws and nuts. What makes Vex different than the Erector set you can buy at the toy store is the collection of parts specially designed for small robotics. It comes with two types of motors — servo and continuous rotation gear head — made to fit the girder construction of the system. To these motors you can attach a variety of mechanical parts, such as wheels, gears, and even tank-type treads.

Once upon a time, Vex was sold exclusively by RadioShack, so it was widely and easily available. Alas, that company had rolled back its inventory in hobbyist electronics parts, and no longer distributes the sets. You can still get Vex via mail order from the manufacturer's website at **vexlabs.com**. They sell a starter kit, as well as separate components. Need an extra set of wheels? No problem! They are available separately, along with various packages of extra girders, nuts, bolts, and other hardware.

The $300 Vex starter kit also includes a programmable electronics module, sensors, and rechargeable batteries. The Vex is the de facto standard kit of parts for the US FIRST Vex Challenge robotics competition, open to high school students. Participants form teams who compete with one another in building and programming a robot to perform a specific task.

LEGO Mindstorms

Now in its second incarnation as the NXT, LEGO continues as a favorite robot construction set in the home and school. NXT sets combine specially-made electronic components with traditional (and some not-so-traditional) LEGO building blocks.

At the center of the NXT system is the NXT controller — a second-generation microcontroller that supports up to three motors and up to four sensors and other inputs. Like the RIS controller in the original Mindstorms set, the NXT provides visual feedback using an LCD display, and receives its programming from a PC (in the case of the NXT, the connect is via USB; early RIS controllers used infrared communications). Once it has received its programming, the NXT is self-contained and operates by itself.

NXT connects to a wide variety of motors and sensors, which either come with the basic Mindstorms NXT kit, or are available separately. Basic touch, light, and sound modules provide basic sensory feedback, and ultrasonic and accelerometer sensors give the NXT the ability to judge distances and even tilt.

The motors on the NXT are enhanced and provide built-in encoders, which count the revolutions of the drive shaft as the robot rolls along the carpet. While NXT comes with graphical programming software, LEGO also provides an open source firmware API (application programming interface), so you can lift up the hood and work directly with the electronics in the NXT module.

Lynxmotion Servo Erector Set

Lynxmotion was one of the first robotics-specific companies to realize the benefits of producing prefabricated construction parts, allowing builders the freedom to create custom designs. Their Servo Erector Set is composed of a variety of stamped aluminum metal brackets that are specially designed to work with common R/C servo motors.

By connecting the brackets, channels, tubing, and other construction hardware in various ways, it's possible to build two-, four-, six-, and even eight-legged walking robots. In similar fashion, you can build arms, grippers, pan/tilt mechanisms, wheeled robots, and just about anything else that uses standard R/C servos. A page on the Lynxmotion site provides a long list of illustrated examples of what you can build using the Servo Erector Set.

The component parts of the company's Servo Erector Set are wide and varied. Lynxmotion sells some small kits with parts designed for specific tasks, like a two- or three-degree-of-freedom leg. Buy four, six, or eight sets to construct a complete walking robot. Or, you can purchase the individual parts separately. These parts aren't inexpensive, but they're far easier than making them yourself, and cheaper than if you go to a local metal shop for custom-made parts.

Rounding out the Servo Erector Set offerings is a complete collection of robot "torsos" or bodies, some made of stamped and pre-drilled aluminum, and some from laser-cut polycarbonate plastic. These bodies are designed to attach to the various legs and other mechanics that you've built using Servo Erector Set parts. Among the available torso parts is the framework for a bipedal robot, the body for a four- and six-legged walking robot, and parts to make a sci-fi looking robot hand.

Robix Rascal

The Rascal Classroom Robot Set from Robix is, as its name implies, designed for classroom study of robotics and mechanics. Yes, you can build several different types of robots with the Rascal kit, but the emphasis is on learning about mechanical design and construction.

Rascal parts are relatively simple and straightforward, and that's the secret to their ingenuity. Kits are composed of standard R/C servos and pre-cut and pre-drilled aluminum angle stock called Rascal Links. The links come in different lengths.

The servos in the Rascal kit are standard R/C unique, and can be commanded using most any serial servo controller or microcontroller. A controller is included in the Rascal kit, which is designed to operate up to 32 servos via a USB link to a PC or Mac. Software supplied with the kit provides a programming environment.

Robotis Bioloid

An exciting new entrant into the prefab robot kit arena is the Bioloid, from Korea-based Robotis. The full Bioloid kit (about $900) comes with enough parts to — according to the sales literature — build 26 different robots. No doubt with a bit of creativity you could build more, especially if you combine parts from additional kits.

The full Bioloid kit — called the Comprehensive Kit — contains a main programmable controller, 18 special programmable servo motors, and various frame construction pieces. A less expensive entry-level kit (about $350) contains the controller, but only four servo motors and a selection of construction parts. Both kits also include a multifunction sensor module, which includes an infrared receiver, infrared distance sensor, light sensor, and buzzer.

Robotis sells their kits through a worldwide network of dealers. See the list of dealers on their home page (**www.robotis.com**). I've included several North American dealers in the Sources section that follows.

As pre-fab kits go, the Bioloid is relatively inexpensive, especially when you consider it comes with everything you need to start building, including the battery pack and the motors. The servos used in the Bioloid are not off-the-shelf; in fact, they are proprietary designs that boast upwards of 220 oz-in torque (at 9.6 volts). They are unlike traditional servos in that they are operated using serial commands. You don't need or use separate pins on a microcontroller for each servo; instead, all the Bioloid servos are connected to the same serial communications input. The Bioloid servos provide torque and angular feedback.

When you consider that a high-torque standard R/C servo motor costs about $40, 18 such motors would set you back $720. At $899 for the full Bioloid kit, you get the equivalent value in servos, plus a microcontroller, battery pack, and construction pieces. Actually, this comparison is not accurate, because traditional analog R/C servos do not provide the feedback possible with the Bioloid servos, and few are capable of torque much beyond 100-110 oz-in. (The 200 oz-in range is mostly occupied by digital hobby servos, which retail for $75-$125 each.)

The Robotis kit is a great deal now, and promises plenty of fun. It's likely only the first one in a soon-to-be long line of all-in-one imported kits from Korea, Japan, and China that are sure to follow. This is a market that is sure to catch on.

Ye Olde Standbys

Not to be forgotten, these construction toys are useful as the framework for DIY robotics. The best part: You probably already have at least one set in the back of your closet. Dig it out, and start using it for your next robotics project!

• *Erector* — One of the older construction toy brands. The original kits were made of all stamped metal, but the modern versions contain a number of plastic pieces. The sets come in various sizes, and are generally designed to build a number of different projects. Many kits are engineered for a specific design with provisions for moderate variations.

• *Plastic Construction Bricks* — You can literally build anything with plastic construction bricks, typified by the LEGO and LEGO Technic sets. A few companies make "LEGO compatible" sets (without actually saying so) with similar functionality. Check out the sets by MEGA BLOKS and others. The parts snap together, but for more permanent creations, you can use a dollop of solvent cement. Use ABS cement for LEGO bricks; plastic model cement for most of the other brands.

• *Tamiya* — Besides plastic models, Tamiya makes a line of educational kits that include motors, wheels, tank tracks, and a (wired) remote control robot construction set.

• *Capsela* — This popular snap-together motorized parts kit uses unusual tubular and spherical shapes. The kits contain unique parts that other put-together toys don't, such as plastic chain, sprockets and gears.

• *Fischertechnik* — These kits are less toys and more learning labs. They aren't designed for use by small children. Pre-made kits offer a snap-together approach to making working electromagnetic, hydraulic, pneumatic, static, and robotic mechanisms.

• *K'Nex* — Using a series of half-round plastic spokes and connector rods, K'Nex encourages construction of larger models. A number of K'Nex kits are available, from simple starter sets to rather massive special-purpose collections (many of which are designed to build robots, dinosaurs, or robot-dinosaurs). Several of the kits come with small gear motors, so you can motorize your creation.

Sources for Prefabricated Robot Construction Sets and Related Parts

80/20, Inc.
www.8020inc.net

Aluminum extrusions and connection parts for industrial-strength constructions. Useful for larger robots. Check the site for local retailers.

Construction Toys
www.constructiontoys.com

Online and local retailer of construction toys. These toys are available both online and in the retail store: Capsela; Eitech; Erector; Fischertechnik; Geofix; Geomag; K'NEX; LEGO Dacta; Roger's Connection; Rhomblocks; Rokenbok; and Zome System.

CrustCrawler
www.crustcrawler.com

US-based reseller of the Bioloid kits, as well as their own custom-made metal robot kits. Extensive site with plenty of how-to and background information on using the Bioloid kits.

e-Hobbyland
www.e-hobbyland.com

Well-established retail and online seller of all types of toys.

Fischertechnik
www.fischertechnik.com

Company website for Fischertechnik in North America. Check out the Retail Outlets links.

Hobby Engineering
www.hobbyengineering.com

General source for robot parts, as well as a lengthy list of Erector set kits of all shapes and sizes.

Jameco Robot Store
www.robotstore.com

Reseller of a number of robot kits, including the Tamiya remote-controlled robot construction set.

KBtoys.com
www.kbtoys.com *or* **www.etoys.com**

Online mail order. Check often for deep discounts on LEGO, K'NEX, and other brands.

LEGO Shop-at-Home
shop.lego.com

Online outlet for LEGO products, including spare parts (when available).

LEGO Mindstorms
mindstorms.lego.com

Informational page for the LEGO Mindstorms sets. Be sure to check out the user-to-user forums to see what other LEGO builders are up to.

Lynxmotion
www.lynxmotion.com

Lynxmotion offers complete robot kits, as well as a unique Servo Erector Set, a collection of brackets, and other parts for building custom robots using standard-size R/C servos.

Only Toys
www.onlytoys.com

Only Toys carries metal Erector sets; most are for building vehicles, and some (like the Steam Engine) are quite elaborate. The company also sells Rokenbok radio-controlled toys.

Robotis
www.robotis.com

Manufacturer's site for the Robotis line of robot construction sets and parts. Check out the dealer pages to find a reseller near you.

Robotshop
www.robotshop.ca

Resellers of Fischertechnik and other robot construction sets.

Tamiya
www.tamiya.com

Home page for Tamiya. Check out their Educational Construction Series line of kits, such as the remote-controlled robot construction set.

Target
www.target.com

Retail stores and online site. Both offer great deals in clearance items. Make it a habit of regularly checking the website for clearance items.

Timberdoodle
www.timberdoodle.com

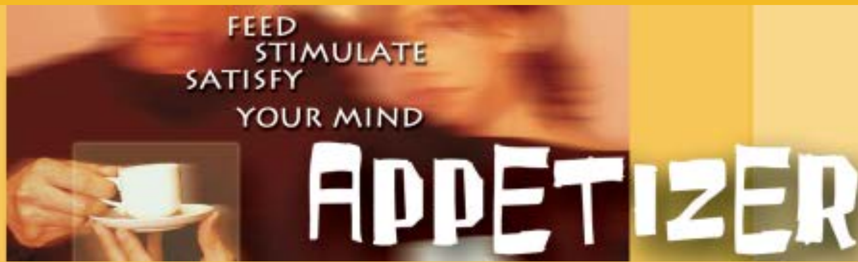
Timberdoodle specializes in home education products. They offer a good selection of Fischertechnik kits at good prices. They also sell K'NEX and electronics learning labs. Be sure to check their "swan gong" closeout deals.

US First
www.usfirst.org/community/fvc/

Main page for the FIRST Vex Challenge. (If you are a high school student, check with your school to see if they participate in the FIRST competition. If they don't, talk to your teacher or administrator about getting involved.)

VexLabs Vex Robotics Kits
www.vexlabs.com

Makers of the VEX Robotics Design System. Use their online ordering system. **SV**



Environmental Sensing in a Robotics Curriculum

by Chris D. Odom

The Allure and Repulsion of Computer Science

The readers of this magazine certainly understand the attraction of robotics. After all, designing, building, and programming robots to do what you tell them to do is exciting, challenging, and mentally stimulating. Recognizing that microcontrollers and robot kits had become both robust and affordable, four years ago I decided to expand my high school computer science curriculum to include the study of robotics. My students have certainly appreciated the upgrade. With their line-following, fire-fighting, soccer-playing, and land-navigating robots, they are proof-positive that robotics applications excite students to the point of increasing both their enjoyment and their understanding of math, electronics, and computer programming. (The effect was so positive and immediate that I even wrote a textbook on using robots in the classroom.)

In the four years since I began teaching robotics at my school, enrollment in my computer science courses has increased nearly 300%. However, this is certainly not the trend experienced by colleges and universities in the United States. Despite the increased demand for a highly skilled workforce, the number of computer science majors in this country has dropped significantly in the past five years. The causes for this decline are numerous, but the two reasons most commonly sited by students and computer scientists is the media's attention to offshoring and the fear of another dot-com bust.

A third reason cited by a growing number of experts is the fact that computer science programs are not seeking solutions to the big questions and the important problems. After all, very few of my students will go on to careers in industrial robotics even though it was robotics which lured them to my courses in the first place. Instead, most of my students will pursue majors in the pure research fields of chemistry, biology, physics, and environmental science; for these are the disciplines that attempt to answer mankind's deepest and most fundamental questions.

Robotics and Pure Science

As I see it, computer science programs in colleges, high schools, and even in middle schools should place more emphasis on pure scientific research. Not only is this healthy for computer science students, but it will also give our science-minded students a deeper understanding and appreciation of technology. I will demonstrate that even in a robotics course it is easy and straightforward to conduct meaningful scientific research, and that the research can be a natural extension of the robotics curriculum.

Today's modern educational science labs use sophisticated data acquisition interfaces (e.g., Vernier's LabPro or PASCO's Xplorer) and a wide array of probes and sensors that measure everything from temperature and voltage to conductivity and turbidity. These acquisition packages are expensive, and while I wouldn't dream of teaching my physics courses without them, I am saddened that my science students treat them as little black boxes that magically record, display, and plot relevant data on cute, multicolored graphs.

My robotics students, too, incorporate a number of sensors to help their autonomous robots monitor their surroundings. But because robotics students are building and programming their own probes and sensors, the data collection process is much more creative, transparent, and rewarding. Even beginning students learn how to use photoresistors, thermistors, infrared transducers, and IR range finders in their robotic applications. Another advantage to designing and programming your own sensors is the cost: a 50¢ thermistor does the job of a $22 temperature probe; a 25¢ photoresistor replaces a $59 light sensor; and two pennies and a piece of conductive foam can be used in place of a $119 force sensor.[1]

I should explain that the school in which I teach is not set up with a research-grade electronics laboratory or an industrial machine shop that some schools enjoy. Instead, I buy off-the-shelf equipment when I can and emphasize software solutions to our robotics problems. To this end, I have found that NetMedia's BX-24 microcontroller and BasicX software fit the needs of my classroom perfectly. The BX-24 is a fast and powerful microcontroller with 16 standard I/O pins (eight of which function as 10-bit A-to-D converters), 20 MHz 32KB EEPROM, and a processor speed of 82,000 BIPS, which is able to perform floating-point math and is capable of multitasking. The BasicX software, which is compatible with Microsoft's Visual Basic™ language, makes programming the BX-24 easy and user-friendly. In my classroom, I use Robodyssey System's educational robots, which are equipped with their RAMB motherboard and the BX-24 microcontroller. The

RAMB allows for direct connection of off-the-shelf sensors (such as IR rangers and GPS modules). Furthermore, with a simple voltage divider, fabricating our own variable-resistance sensors such as thermistors and photoresistors is a trivial matter.[2] My point is this: Collecting, recording, and analyzing data is a breeze with the BX-24 and RAMB.

The Environment Sensing Station (ESS)

It seemed to me that a few lessons in environmental science, with its heavy dependence on remote sensing and data collecting, would be a natural extension of my robotics curriculum. To test this idea, I designed an Environment Sensing Station (ESS) made of three-inch (I.D.) PVC pipe and fitted it with a BX-24 and RAMB to make simultaneous readings of up to seven sensors. (The BX-24 has eight analog ports, but I used one of them to set the apparatus to data-taking or printing mode.) The ESS, which only cost a few dollars to build, is capable of collecting data for several days — or longer if a solar panel was used to recharge the batteries. The station is weather- and water-proof, so I decided to test it underwater using five sensors to measure and record internal and external temperature, ambient light, turbidity, and conductivity.

All the sensors were inserted into Robodyssey voltage divider boards (VDB) equipped with variable resistors.[3] These VDBs were then connected to the BX-24 A-to-D pins via the RAMB. While I generally prefer the fixed resistor VDBs, the variable ones are nice during the development stages of an experiment. Measuring temperature and light intensity is a simple matter of screwing a thermistor and photoresistor into their respective VDBs. Conductivity was measured with a gold IC wire-wrapping socket. The pins along one side were soldered together and connected to a wire, as were the pins along the other side. The two wire leads were then connected to a VDB. In air, the resistance between the two rows of pins is very high so the signal read by the BX-24 is nearly equal to the input voltage of 5V. In water, the resistance varies according to its mineral content, which can be quite variable. I measured the turbidity of the water by shining a bright LED directly into a photoresistor.[4] During this test run, none of the sensors had been calibrated, so only raw data is displayed.

It is not necessary for the ESS to be tethered to a PC, for data can be stored with the BX-24's EEPROM. The amount of data one can collect is dependent upon the amount of available space. Because my computer program consumed only about 6,000 bytes of the 32,768 total, 26,768 bytes were set aside for data storage. Each sensor was connected to an A-to-D channel on the BX-24 and each data point was stored as a two-byte integer. I also decided to store the local time using the BX-24's built-in real time clock (RTC), which requires 12 bytes per reading. With five sensor readings and the time, 22 bytes of EEPROM was consumed with each data sample. This means, for example, that our application could collect one sample each second for 20.3 minutes; or it could collect a sample, say, every 15 minutes for a total of 12.6 days. Not bad, huh?

For my first run, I elected to record data every two minutes for 18 hours. Little did I realize this test run would yield some amazing data. At 18:10 on August 7, 2006, I began acquiring data. At 18:20, I left my office and carried the instrument to a creek running through campus. At 18:37, the ESS was placed in the creek and submerged under about 25 cm of water.

Data was later imported into Excel and graphed back in the lab. If you'd like to study the graphs (which are available on the *SERVO* website; **www.servomagazine.com**), note that light intensity, temperature, and conductivity increase along the vertical axis. You should know that on August 7 there was a brief but powerful thunderstorm that occurred between 21:53 and 22:15.

If you look at the *Light/Turbidity graph*, note that the ambient light probe was sensitive enough to detect twilight and dawn conditions, the timing of which agreed with meteorological forecasts. From that graph also note that the turbidity sensor detected an influx of soil runoff immediately after the storm. I assume the noise recorded by these two sensors after sunrise was due to debris that became trapped by the ESS.

Data shown on the *Temperature graph* illustrate many interesting thermodynamics processes. Note that the internal temperature lagged behind the external temperature as expected. This is most pronounced when the ESS was first taken outside into the hot weather and again when placed into the cool stream. Note, too, that the water temperature rose substantially after the storm runoff from the warm earth made its way into the creek. Finally, notice the lag time in the diurnal heating and cooling of a body of water associated with sunrise and sunset.

Data from the *Conductivity graph* also shows the effect the thunderstorm had on the creek's electrical properties. While more difficult to analyze, the effect of the storm is unmistakable.

I believe there are many things to glean from this experiment. First, there is room in a robotics curriculum for pure scientific research. Second, it doesn't take a lot of equipment or programming skill to perform solid scientific experiments. Third, even the simplest of experiments can offer a wealth of data that could keep a classroom occupied for days.

So, will this and other similar traditional science experiments find their way into robotics curriculum? Absolutely! Certainly my future robotics students will continue to create their fire-fighting and soccer-playing robots, but they will also see how technology and science mesh together. Hopefully, my students will get a taste of real scientific inquiry that may even be more rewarding than they would get in a traditional science class.

For more information on the this experiment including computer code, raw data, graphs, construction plans, classroom curriculum materials, and detailed local weather reports for August 7-8, 2006, see **www.basicxandrobotics.com/apps/** and click on Environment Sensing Station. **SV**

Footnotes

1) All comparison prices are from PASCO's 2007 catalog.

2) In 2004 we launched the BX-24, the RAMB, and Robodyssey's voltage divider boards (VDB) into space onboard NASA's Terrier-Orion rocket. The acceleration at lift-off was nearly 20Gs and all components worked flawlessly throughout the 25-minute mission.

3) With the addition of a Robodyssey Multiflexer, we would be able to monitor seven additional sensors.

4) You can fabricate your own divider board, but the VDB is inexpensive and easy to use.

ROBOBusiness
CONFERENCE & EXPO 2007

The International Business Development Event for the Mobile Robotics & Intelligent Systems Industry

May 15-16, 2007
Hynes Convention Center
Boston, MA

Conference Tracks

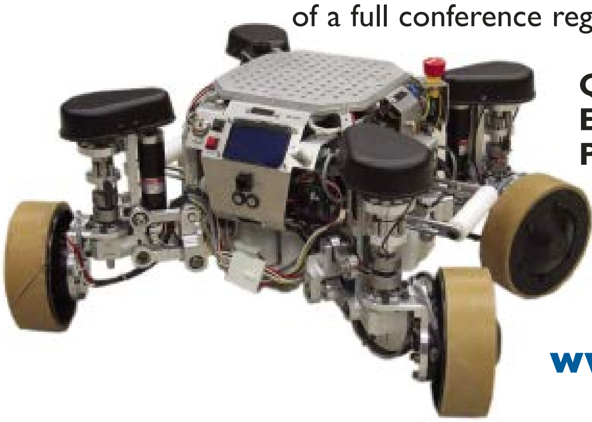
- Business Development and Investment
- Technology and Standards
- Applications and Products
- New Markets for Industrial Robotics
- Application and Products
  - First Responder Robotics
  - Security and Defense
  - Intelligent Transportation & Field Robotics
  - Healthcare Robotics
  - Consumer Robotics

Now in its fourth year, the RoboBusiness Conference and Exposition is the 'must attend' event for the mobile robotics and intelligent systems industry.

- Two full days of sessions and keynotes in 5 tracks featuring 50+ world class speakers from business, government and academia
- Expo floor featuring the latest robots and robotic technology from around the world – many never seen in North America
- Special events, networking receptions, Robot Hall of Fame luncheon, breakfast keynotes hosted by Evolution Robotics and Microsoft and more!

REGISTER EARLY AND SAVE

Register online at www.roboevent.com and use Priority Code RBCD200 to save $200 off the price of a full conference registration.



Questions?
E: info@roboticstrends.com
P: 508-663-1500 x329

www.roboevent.com

Founding Sponsor

iRobot™

Premier Sponsors

The Technology Collaborative

Mass Technology Leadership Council

Gold Sponsors

CoroWare

evolution robotics

freescale semiconductor

Holland+Knight

Microsoft

Corporate Sponsors & Exhibitors

Advanced Digital Logic
Atair Aerospace
ATI Industrial Automation
Autonomous Solutions
Axis New England
Boston Engineering
Braintech
Carnegie-Mellon University
City of Osaka
Energid
Foster-Miller
Galileo Mobility Instruments
Gears Educational Systems
Global Emerging Trade Organization
GOSTAI
Hagisonic
Hobby Manufacturers Association
Hokuyo Automatic
Honeywell
Hydroid
InterSense
InTouch Health
IZI Robotics
Japan External Trade Association
Kumotek
MachineBus
Magmotor
Merlin Systems
MicroInfinity
NeuroRobotics
NEXSTAR
Object Management Group
OC Robotics
OceanServer Technology
OLogic
Perrone Robotics
Robosoft
Robot Hall of Fame
RoboTeX
RoboTech Center
Robotics Society of America
SCHUNK
Segway
SolidWorks
Strategy Analytics
Valde Systems
Vecna Robotics
Virtus

Media Sponsors

SERVO Magazine

Nuts & Volts

Then and NOW

ROBOT NAVIGATION

by Tom Carroll

Navigation — the definition from Webster's *New Collegiate Dictionary*:

"The act or practice of navigating (duh). The science of getting ships, aircraft, or spacecraft (and, I'll add robots) from place to place. The method of determining position, course, and distance traveled."

Basically, it means the process of planning to get from where you are to where you want to go. You do that when you get into your car and drive somewhere or even when you walk across a room. Every robot has to do some sort of navigation, even industrial robots that are fastened to a single location. Ever since the first robot designers connected a computer to some sort of mechanical arm or mobile base, that person also had to know the location and orientation of the arm's end effector or the location of the base. This is where the design and operation of a robot becomes a bit more complex. We find that we have to jump from simple motor control to *intelligent* motor control.

Robot Navigation Compared to Boat Navigation

There are numerous factors to be considered in true navigation, but I'll just list a few for lack of space and time. These main elements are position, course, and distance. Notice that I have left out *obstacle detection* — a factor that alters a course and will be discussed later. Referring to the above list of elements, the first is *position*. To have navigational autonomy, the robot must know where it *is*. (We'll discuss methods of determining position a bit later, as well.)

The above definition also mentioned *course*, but, to determine a *course*, the robot must know where it is and the position of another location where the robot is supposed to go. For example, a boater can look at a chart and see that he has to travel on a course of 190° to go from his present location to a harbor on a nearby island. Of course, he has to take into consideration *variation* of the Earth's magnetic field where he's at. The magnetic north pole is quite a ways south of the true North Pole, so variation at Los Angeles is 13.5° to the *east*, whereas it is 18.3° east to the north at Seattle. What this means is the needle of the compass is pointing further clockwise to the right than it should. On the East Coast, it varies to the *west*. There is a wavy line in the central US where the compass points directly north, to both the magnetic north pole *and* the true North Pole.

The boater also has to consider *deviation* caused by magnetic and electrical things on his boat, so his *compass* course may have to be 15° to 25° from the 190° *true* course he measured on his chart to have an accurate course to the other harbor. Robots have the same problem with magnetic compasses, whether the flux gate type or any magnetic compass installed on a robot with ferromagnetic materials and electrical systems on board. One neat robot that I saw years ago used a magnetic compass for the general direction reference and had a gyro compass to determine degrees of turning *immediately* as a magnetic compass has a bit of a lag.

It is also helpful to know the *distance* that you have to travel to get to the next location. In a boat, after reading the chart and measuring the distance with a compass, you can estimate your distance traveled by knowing your boat's speed and the elapsed time traveled (D = S x T). Of course, in boats, there are currents and winds and the angles they present to the boat's course that you have to take into consideration, but usually not with robots.

Robots and cars can use what is known as *odometry* — or measuring the distance by wheel revolutions and shaft encoders on the wheels to create pulses for the microcontroller to count, or mechanical odometers in cars. This method of determining speed and distance along multiple points in a course is know as *dead reckoning* and is the basic navigation method used with most mobile robots for course planning.

Early Robot Navigation

The ability for an experimenter's robot to navigate has not always had today's processing power and sensors available for the machine's designer. One of my earliest robots used a bunch of switches and six volt DPDT relays that my brother had given me as the 'logic' for my robot. It had two motorized wheels arranged in a differential (tank style) configuration and when a microswitch 'whisker' touched a wall or obstacle, it would kill the power to the opposite side's motor for a moment and

Figure 1. Elsie — crude but effective.

allowed the robot to turn away from the wall. If it hit a wall head on, a wooden stick that ran from front to back with bumpers on each end toggled a DPDT switch to change the polarity of the battery power supply to the motors. It worked well for a kid's robot as it 'navigated' itself around my house like a crazed sheep randomly grazing grass.

I later added the relays to provide a bit of logic (i.e., if *this* switch and *that* do *this*, then cause *this* action to occur) and sometimes wired them wrong to occasionally short out the batteries in certain situations. The hard part came when I used a couple of cadmium sulfide 'photo resistive cells' to trigger relays in the presence of light. I had one mixed up robot for the longest time until I finally figured out a relay logic circuit to have the robot track light. Is that navigating? No way!

It sure amazed my friends and my science teacher, though. I managed to learn logic the hard way and ate up a bunch of flashlight batteries in the process. It sure would have been nice to have some simple logic ICs or a microcontroller back then. My 'programming' and 'software' was anything but soft; it was a hard, hot soldering iron.

Grey Walter's Tortoise Navigates to a Light

In my *SERVO* column of October '05, I wrote about Grey Walter — the English neurophysiologist who built the first autonomous mobile robot in 1949 that he called an *imitation of life*. His robot named Elsie only had a two-tube 'brain with a single photoelectric tube 'eye,' yet was able to navigate to its 'hutch' much like a bacterium. Figure 1 shows Elsie without her plastic shell. The photoelectric tube stands above the front, steered wheel and has blinders on each side to focus only on a light source. You can see the worm gear drive steering system and the two tubes straddling the shaft in the middle of the robot.

Figure 2 shows a diagram representation of the robot's path to a seven-watt light source. Notice the opaque objects that Elsie must search around in order to locate the light and the wrong turns that caused her to erratically search until the light was rediscovered. When it came close to the light and its hutch, it would dance about wildly, much like a moth near a porch light, until its battery was low, and then enter its hutch until the batteries were replaced.

Figure 2. Elsie's path to light.

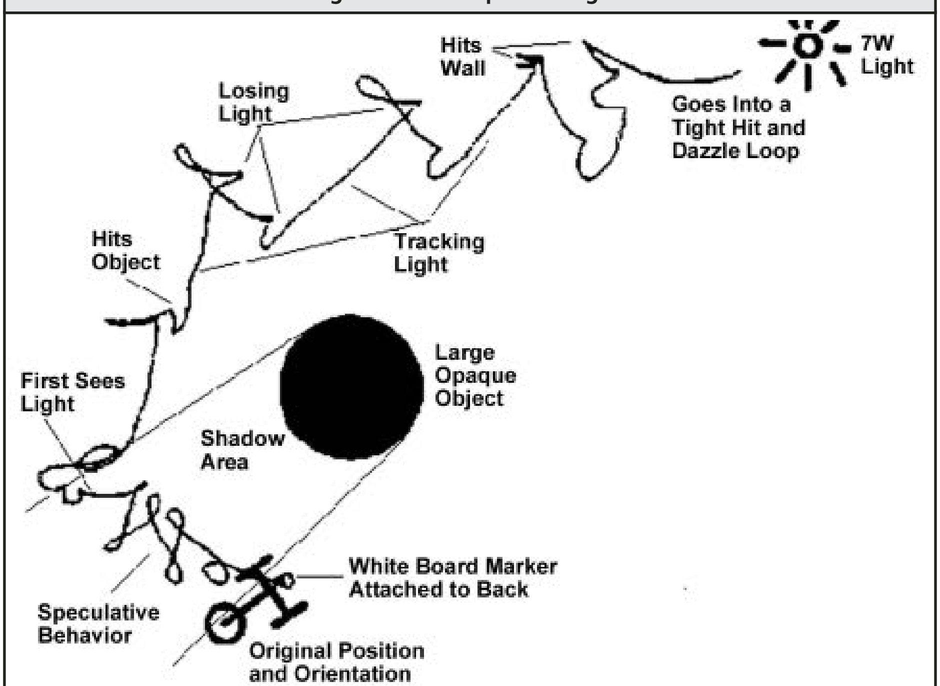


Figure 3 shows the actual path of one run highlighted by a candle on top of the robot in a timed exposure. Walter used an early form of behavior-based logic that is now used in the designs of many small robots today. He developed the first artificial intelligence, artificial life form, and perceptual navigation to be used in a robot. This is pretty amazing navigation for a robot of six decades ago.

Modern Robot Navigation

Let's jump forward 60 years to the age of microcontrollers, GPS, and precise navigation. I purposely skipped mentioning a robot's ability to determine its position until I could discuss the methods that we have today. In course determination, you have a starting point and a goal or ending point. In many robot contests, a robot is placed randomly at a location and it scouts out its goal. In many of the fire-fighting contests like the Trinity College contest in Connecticut, a robot starts at a gate in a maze and must navigate its way to an open flame (candle) and extinguish it.

In other applications, a robot can start anywhere and search for its goal. I can randomly place my Roomba floor-sweeping robot on the floor, turn it on, and it navigates its way about the room, sweeping in a definite pattern. I have two virtual wall units that I can place by an open door or the edge of a stairwell to tell it to not go there. These units act as beacons that tell the robot that these IR beams are the same as hard walls. These devices act just like the aids to navigation that boaters use, such as lighted buoys, to tell them to avoid dangerous areas.

Unlike an unmanned airplane or autonomous underwater vehicle that operates in a 3D environment, most of our robots operate in a two dimensional, flat plane environment. Robots can use XY coordinates to determine their position, the direction it is facing, their eventual target area and course to follow, and the required distance to travel. Resourceful experimenters have placed retro-reflectors that act somewhat like my virtual wall units to tell a robot where a door is located or a dangerous area such as a drop-off. These can be made of simple reflective tape that you can buy in many stores. The robot can bit map a room by a series of random searches and store this in its memory for use at later times.

Various vision systems such as the CMU cam and AVR cam can look at an area, determine obstacle locations, map out a course, and actually steer the robot around these objects to arrive at its target location. The new WowWee Roboquad (see Figure 4) crab robot uses a suite of various sensors in its head to navigate about a tabletop.

The many types of sensors that a robot experimenter has available determines just how his robot will navigate its way about its environment. Direction-finding devices such as compasses are not the only tools he has available. Mechanical whiskers connected to simple SPST switches are the most popular devices to detect walls and obstructions. Active ultrasonic and IR distance sensors are very popular to detect barriers, walls, and obstructions without the robot having to actually go to these objects to 'feel' them.

Maze and micromouse robots can use reflective phototransistor/LED pairs to detect drawn lines or tops of walls. Gyros designed for RC helicopter applications, ring laser gyros, aircraft instruments, accelerometers, specifically located active beacons and bar codes, and even hacked computer mice for X-Y positioning can all assist the robot in determining its location and orientation. An experimenter can place multiple IR beacons in a room and the robot can triangulate the angles to the beacons and determine its exact position by simple trig calculations.

GPS — The Ultimate Navigation Aid for Autonomous Vehicles

We've all watched with interest the two DARPA Grand Challenges where competing autonomous vehicles had to traverse over a hundred miles of desert environment, following a course set out by DARPA just before the start of the race. The first race basically fizzled with the leader only making a few miles and most of the rest failed just past the starting gate. The next race was much more successful with several vehicles completing the 132-mile course.

The VW Touareg SUV from Stanford University won the two million-dollar prize in just under seven hours. The next planned race — The DARPA Urban Challenge — is an even more exciting competition with the vehicles driving through a *city*. DARPA announced opportunities for urban challenge teams to receive funding in amounts up to $1M to develop their autonomous vehicle. Sixty-five proposals were reviewed and evaluated, and 11 recipients were announced, including a smaller VW Passat from Stanford that is a bit more street-friendly. The prizes are $2 million for the fastest qualifying vehicle, and $1 million and $500,000 for second and third place. The original race vehicles relied on sophisticated vision systems to avoid obstacles and keep on the rugged roadway, but their main navigation aid was GPS.

On June 26, 1993, the Air Force launched the 24th Navstar satellite into orbit, completing a constellation of 24 satellites known as the *Global Positioning System*, or GPS. In May of 2000, a signal degradation system that the military applied to the GPS satellite signals to prevent possible enemy use of the highest accuracy part of GPS, *Selective Availability*, was removed, thus allowing greater accuracy for all users. This new DARPA challenge will also rely heavily on this system.

GPS really hit home for the robot experimenter when the Seattle Robotics Society and other groups began what is called Robo-Magellan contests. These small, under 100 pound robots use GPS to navigate a specific course outdoors. These competitions have proved to be very popular.

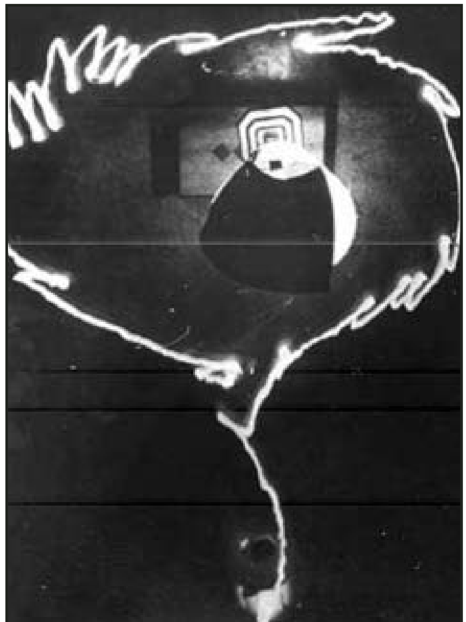


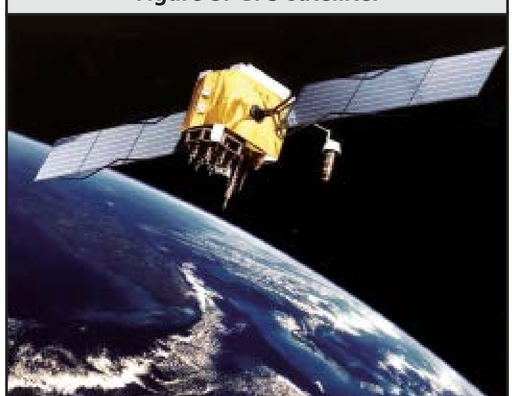
Figure 3. Elsie circles light using a candle as a beacon.

Figure 4. The new WowWee Roboquad robot.

How Can I Use GPS With Robots?

A constellation of 24-30 or more GPS satellites is in orbit around the earth (see Figure 5). They are in what is known as 'half geosync,' or in an approximate 12-hour orbit and three or more are visible at any place on earth at any time. Newer ones are orbited as older ones are taken out of service and most GPS equipment is updated as to the location of specific satellites. Each

Figure 5. GPS satellite.



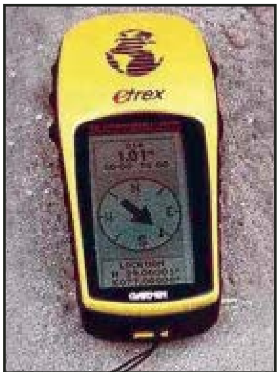


Figure 6. Typical hand-held GPS receiver.

Figure 7. Winner of 2006 RoboMagellan contest at Robothon. Notice the small black GPS external antenna.

Figure 8. Bluetooth MotoNav.

satellite downlinks a high power, narrow bandwidth, 1575.42 MHz signal that is received on larger, professional receivers, vehicle-mounted, and hand-held receivers.

The receivers receive signals simultaneously from all the satellites visible in a particular area. This is a key point as the signals are so weak that inside of buildings, tall buildings, terrain, and even trees and clouds can mask the signals. The signal strength in a building can be as feeble as -160 dBm — think of one milliwatt with a decimal and 15 zeros in front. Some of the contestants in our SRS Robo-Magellan competitions have had one heck of a time because of a high wall and the nearby Space Needle.

Each GPS satellite has on-board an extremely accurate cesium or rubidium clock so that all the satellites in orbit are in perfect synchronization with each other. The original Block 1 group of satellites used a rubidium clock with an accuracy of one second in 1,000 years. The newer satellites use a cesium beam clock with a thousand-fold increase in accuracy.

The GPS receiver detects the slight differences in the various signal arrival times to plot the receiver's location. The system needs at least three satellites to obtain a latitude and longitude positional fix. A lock on four satellites allows the receiver's height to be determined and the more satellites that can be received, the higher the accuracy of the results. If the position of the "receiver is changing, speed and direction of movement can be determined. If the received satellites are closely grouped, the accuracy will not be a good as if the satellites were spread out across the visible sky. Likewise, if the receiving time is spread out, the accuracy will increase over time.

Differential GPS corrections (DGPS) and a Wide Area Augmentation System (WAAS) that uses ground stations to augment the orbital signals can dramatically increase the accuracy of a simple GPS system in a small robot. Europe's planned $4 billion Galileo nav satellite system should be in place by 2010 and promises a marked increase in accuracy, especially in buildings where the US GPS systems has difficulty due to weak signal strength. Weather, upper atmospheric phenomena, multi-path signal errors, and shadowing can cause system problems but the newer receiver manufacturers are striving to solve these issues.

GPS for the robot experimenter is available for less than $100. Hand-held units such as the Garmin E-Trex (see Figure 6) have been connected to mobile robots and used for outdoor navigation. Figure 7 shows the winner of the Seattle Robotics Society's 2006 Robothon Robo-Magellan Contest from Washington State University. It used GPS and cameras to navigate its way about a thousand-foot course near the base of Seattle's Space Needle.

Figure 8 shows Motorola's new T815 Bluetooth connected MOTONAV unit that was designed to be used with Bluetooth cell phones for car or walking navigation. It talks to your cellphone giving you driving instructions to a destination. I have already heard of plans by robot experimenters to use the system for mobile robot navigation.

You'll find millions of hits looking up "GPS" on a search engine, a great place for further study of this very useful technology. Accuracy approaching the centimeter range will soon be possible for moving vehicles. Robots need no longer to search out a beacon, bump into obstacles, and perimeters to know where they are. GPS coupled with today's inexpensive sensors and cameras can turn a simple mobile platform into a sophisticated robot. **SV**

Advertiser Index

All Electronics Corp. 27, 43
AP Circuits 56
AWIT 43
CrustCrawler 3
Electronics123 27
Futurlec 43
Hitec 7
Hobby Engineering 26
Images Co. 43
Jameco Robot Store 2, 43
Lorax Works 27, 43
Lynxmotion, Inc. 13
Maxbotix 43
Maximum Robotics 24, 43
Net Media 83
Parallax, Inc. ....Back Cover
PCB Fab Express 50
PCB Pool 12, 43
Pololu Robotics & Electronics 25, 43
RCATS 27
ROBO Business 2007 78
Robodyssey Systems, LLC 74
ROBOlympics 67
RoboteQ 9
Robotis Co. Ltd. 57
RobotShop, Inc. 43, 75
Schmartboard 27
Sensory, Inc. 43, 56
Solarbotics 18
Technological Arts 43
The Machine Lab 40
Titanium Joe 35
Vantec 50

Protect the environment
Make the right choice
SerialLCD
Easy to use!
Priced from $34.95
Complete info
at BasicX.com
ALAILABLE IN RED, GREEN AND BLUE.
SitePlayer
TELNET
PoE - Power Over Ethernet
Serial to Ethernet converter that allows your serial device
to be controlled via ethernet or through the internet.
$129.95 www.siteplayer.com
RoHS
2002/95/EC
COMPLIANT PRODUCTS FROM NETMEDIA.
NetMedia
NetMedia Inc., 10940 N Stallard Pl, Tucson Arizona 85737 tel.520.544.4567 fax 520.544.0800 www.BasicX.com