ROBYTES • ROBOT COMMUNICATIONS • FLOCKBOTS

SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

www.servomagazine.com

JUNE 2007

Machine ART

- Dial-A-Voltage
- Making Great Connections
- Robot Simulation for Everyone
- Personality and Autonomous Robots

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...

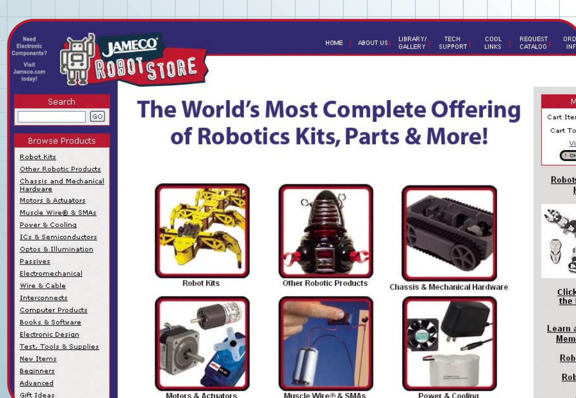
...there's something for everyone!

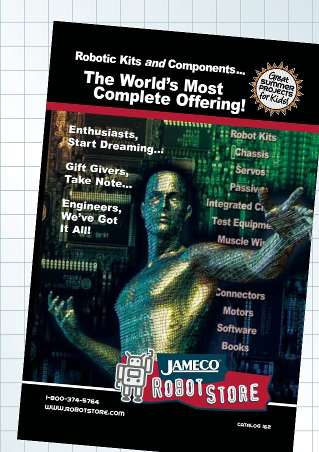
Robotic kits help you and your child to experience and learn about perception and control using a variety of sensors and actuators. Challenge yourself with Jameco's selection of fun and interactive kits! You or your child can assemble the kits and then enjoy endless hours of discovery.

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.

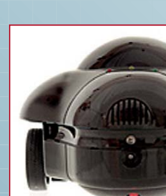


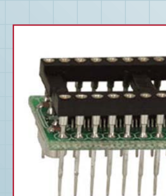


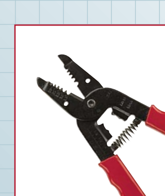
WWW.ROBOTSTORE.COM/SRD

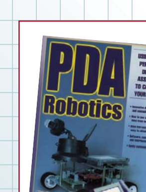
1-800-374-5764

Call for your free catalog today!









SERVO MAGAZINE

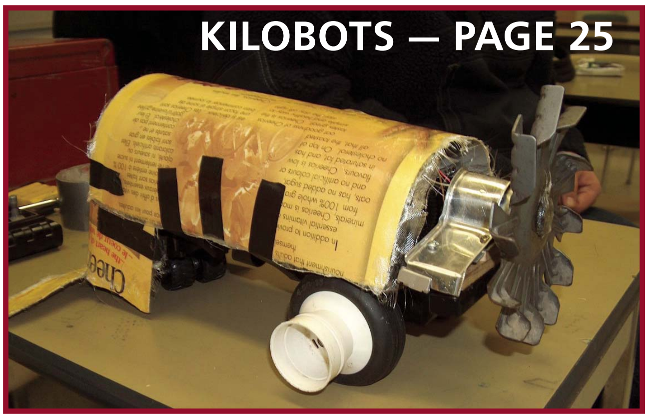
Columns

PAGE 63

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

06.2007
VOL. 5 NO. 6

Features & Projects


KILOBOTS — PAGE 25

Departments

Mind / Iron

by Bryan Bergeron, Editor

Roboticists, as a group, invest a great deal of mental capital contemplating the future. Moreover, unlike mere theoretical futurists, we also devote our time and resources to actively creating the future that we envision. Robotics is, after all, a hands-on activity — whether the hands are on a keyboard developing a new learning algorithm or on a lathe creating a new gear design. As every reader who has designed and built a robot knows first-hand, transforming a vision or even a well-planned functional specification of what could be into reality can be daunting at times, even to a seasoned roboticist.

Consider the challenges inherent in creating a robot from scratch. Algorithms that execute flawlessly in simulations may prove useless under real-world conditions, where imperfect sensors, actuators, and mechanics are the norm. Armed with what appears to be a sound design, the appropriate electronic components and mechanical parts must be located and ordered. Often, limited supplies or prohibitive costs dictate the use of substitute components that may adversely affect performance and require revisiting the original design.

Even with the optimum components and materials in hand, an experienced electronics or mechanical engineer may toil for weeks or more, interfacing mechanical and electronic systems originally designed for non-robotics applications. Because robotics is a dynamic, evolving field, hardware and software standards are at best fleeting. Furthermore, writing custom code for ordinary functions and reinventing mechanical assemblies is often required to provide the infrastructure upon which a new robot can be built.

Despite numerous challenges, we press on — often for very different reasons. Some of us are quintessential early adopters, drawn by the technology, perhaps eager to explore the potential of a new sensor or microprocessor that might hold the key to a revolutionary form of robotic behavior. Others are motivated to explore the universe of new application areas that robotics offers, from robotic surgical assistants and robotic ambulation aids for the elderly to smart homes, energy-saving vehicles, and planetary explorers.

Some roboticists are intrigued by the nuances of human-robot interaction. They're drawn to issues such as how humans and robots can form collaborations, how to best leverage human-robot emotional bonds to address the healthcare needs of patients, and how robots can serve as prosthetics and amplifiers for the physically challenged. The more competitive among us are drawn to the combat arenas where the best designs — and visionary roboticists — prevail. Still others devote their energies to robotics as a means of learning — or teaching — science, math, engineering, and creative thinking.

Regardless of motivation, we all face the challenges of acquiring quality information and of investing our design and development time in a way that provides the greatest return. For example, a roboticist with a vision for a better robotic wheelchair, exoskeleton, or other assistive device for the elderly could either begin working from first

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

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS

Jeff Eckert	Tom Carroll
Gordon McComb	David Geer
Pete Miles	R. Steven Rainwater
Michael Simpson	Kevin Berry
Paul Weijers	Jim Miller
John Blankenship	Samuel Mishal
John Square	Fred Eady
Paul Pawelski	Kurtis Wanner
Robert Woodhead	Karl Wolter
Ray Billings	James Isom

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux
Brad Stoddard

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

BIO-FEEDBACK

Dear SERVO:

While I have always been a fan of *SERVO*, and will always continue to be both a fan and a contributor, I am thoroughly disgusted with you. The May issues' Mind/Iron was used specifically to plea for government subsidies for a single, private organization — not robotics and engineering education in general.

While I obviously write articles encouraging people to attend my own events — along with a full year's series of article encouraging people to attend OTHER events around the world — I certainly have never used *SERVO Magazine* as a personal platform for financial enrichment from the taxpayers — or anyone else, for that matter. I completely agree with USFIRST's goal of encouraging science and engineering education.

However, that they charge schools $6,000 per year, every year, for about $2,000 worth of parts — 90% of which are redundant from year to year — in my opinion, amounts to extortion. Especially considering the amount of corporate sponsorship funds they receive above and beyond team fees.

Baseball teams aren't required to buy new bats, balls, and gloves each year and programmers aren't forced to buy a new computer every time they write new software. So high school students shouldn't be forced to pay $6,000 each year for what is essentially the same kit that they bought the previous year. (How many IFI controller pairs or surplus drill motors does any one team need?)

That you should allow your magazine to be used as a political tool for the enrichment of a single organization — one which continues to over-charge high school students and exclude those who cannot pay beyond the first two years — is unconscionable.

I realize that my view on this is generally perceived as professional envy, even though I have mentored a FIRST team for three years and donated many hours volunteering for regionals (as regional FIRST staff are unpaid volunteers — which further begs the question where the money goes). It is not envy. It is disgust at what I view as the immoral over-charging of minors. And now, they are asking the federal government for even MORE money.

I know damn well how much money it takes to run a large-scale robot competition (to the penny), and what they charge kids above and beyond their corporate donors FAR exceeds reasonable costs. RoboGames does not now, nor will we ever, charge high school teams to compete.

Although this letter is not an exercise in comparative charity, it is merely one of shock that you would allow *SERVO Magazine* to be used as a lobbyist tool for the benefit of a single organization.

David E. Calkins
Director, SFSU Engineering Design Center & Robotics
President, Robotics Society of America/ROBOlympics

We at SERVO Magazine happen to disagree with David's perspective. Any worthwhile program that will raise the consciousness of our youth through science and technology to produce engineers instead of video gamers goes a long way. SERVO Magazine has had a long involvement with FIRST (and other events) and agree with the concept of getting robotic programs into schools. There was no attempt on SERVO's part to provide space for personal gain or to discount other events. We believe that the more people who get involved in robotics, the better it is for all of us.

principles or take a more rational approach and leverage the intellectual capital expended by others.

Publications such as *SERVO*, quality websites, and personal interchanges with other roboticists serve as information filters that minimize thrashing time. Building on the strength, knowledge, successes, and failures of others has the effect of multiplying time because we don't waste valuable time or resources determining first-hand what works and what doesn't. Using the appropriate information filters also maximizes the likelihood of a serendipitous encounter with a vetted article or advertisement that may suggest a new robot design or application area.

SERVO is both a resource and a launch point for further research for hands-on robotics experimenters. Moreover, this publication is an adaptive filter that adjusts to reflect changes in the rapidly evolving field of robotics and your expanding needs. If you're looking for more coverage of a particular topic in robotics, let us know, and we'll do our best to adjust our content accordingly. If you're an avid roboticist with experience that would be valuable to other readers, then please consider contributing an article.

Whether you read *SERVO* on a flat panel or printed page, you're participating in a symbiotic relationship in which our editorial staff works to identify areas that are of interest to you, our talented and knowledgeable contributors develop content, and our advertisers offer components and services that you'd be hard-pressed to find on your own. I look forward to hearing your suggestions and, for contributors, your ideas for articles that can help transport our readers into their future. SV

Robytes

by Jeff Eckert

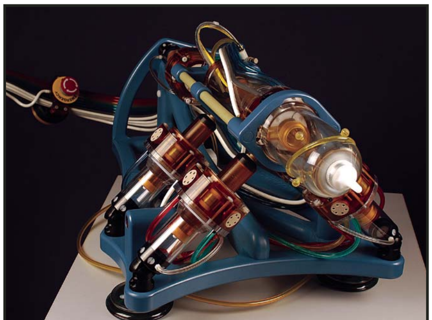
Forget the Rocking Chair, Here's a Walking Chair

The HUBO-FX1 walking chair. Photo courtesy of Humonoid Robot Research Center.

Most chairs have four legs and pretty much just stand there, but the HUBO-FX1, from S. Korea's Humonoid (sic) Robot Research Center (ohzlab.kaist.ac.kr) has two legs and actually walks. The machine — which operates with 12 degrees of freedom (three per hip, one per knee, and two per ankle) — can carry a person or other load of up to 268 lbs (100 kg). When the payload is human, it can be controlled using a built-in joystick controller.

The machine is about 10 ft (3 m) tall, weighs in at about 400 lbs (150 kg), and is fitted with a range of sensors. Each ankle employs a three-axis force/torque sensor that measures normal force and two moments. Each foot has a sensor that measures the gradient beneath it and relays information to the internal gyro for stabilization. Because of a bit of a language barrier, it wasn't quite possible to figure out if the machine is a prototype or currently for sale, but it is intended for a variety of uses such as "carrying old and feeble people," industrial and military transportation, and accident/disaster assistance. Long-term improvements include fitting it with a battery power system (yes, that's a power cord, not a tail), teaching it to climb stairs, and giving it more strength and speed.

Motor Drives Robotic Biopsies

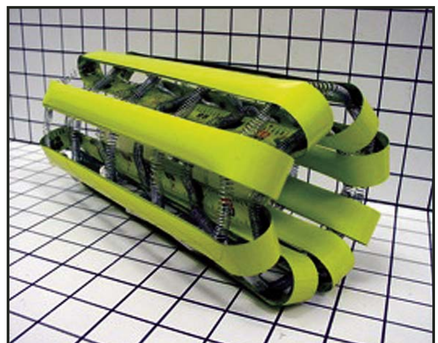


The PneuStep motor — developed by Johns Hopkins — employs no metal and uses no electricity. Photo courtesy of the Johns Hopkins Urology Robotics Lab.

As this month's offering in the category of "things that make you squirm," we offer the PneuStep motor — a recent development of the Johns Hopkins Urology Robotics Lab (the very existence of which is vaguely unsettling, but you can pay a visit at **urology.jhu.edu/urobotics/lab/**). It seems that prostate biopsies are generally performed "blind," because the only high-tech way of spotting small, early tumors is by MRI. Unfortunately, it has not been possible to employ robotic control of the biopsy needles, because metals are unsafe in that environment and electric currents distort MR images. But the PneuStep is built entirely out of plastics, ceramics, and rubber, and it operates entirely on air and light.

As a result, it is now possible to let a surgical robot ride into the MRI scanner with you where — controlled by a computer in an adjacent room — it prods you more precisely than any human could and reports back via fiber optics. A trial run has already been accomplished using six of the motors. According to lab representatives, the system, which can produce precise motion down to 50 µm, came within 1 mm of the target in all cases. "This remarkable robot has a lot of promise, "added one observer. "The wave of the future is image-guided surgery to better target, diagnose, and treat cancers with minimally invasive techniques." Okay, but put me to sleep first.

$400,000 Mechanical Amoeba



The Whole Skin Locomotion mechanism under development at Virginia Tech will mock the pseudopod of an amoeba. Photo courtesy of RoMeLa.

As part of its Faculty Early Career Development (CAREER) Program, the National Science Foundation has awarded a five-year, $400,000 grant to Virginia Tech researcher Dennis Hong, who will use it to design and create what he calls the Whole Skin Locomotion (WSL) mechanism, which is largely based on the principle of an amoeba's pseudopod.

With its elongated cylindrical shape and expanding and contracting actuating rings, the WSL can turn itself inside out continuously, mimicking the motion of the cytoplasmic tube an amoeba generates for propulsion.

"Our preliminary experiments show that a robot using the WSL mechanism can easily squeeze between obstacles or under a collapsed ceiling," Hong said. The mechanism, which can use all of its contact surfaces for traction, can even squeeze through holes with diameters smaller than its normal width. This could be a useful movement method for search-and-rescue robots and (back to the squirming effect) medical equipment such as robotic endoscopes.

Hong is director of Virginia Tech's Robotics and Mechanisms Laboratory (RoMeLa, **www.me.vt.edu/romela**), where WSL actuation models will be analyzed and prototypes will be built and tested. Other lab projects include IMPASS (Intelligent Mobility Platform with Active Spoke System), DARwin (Dynamic Anthropomorphic Robot with Intelligence), and STriDER (Self-Excited Tripedal Dynamic Experimental Robot).

Wi-Fi Spy Bot Coming

Rumors abound regarding the price (probably about $400) and availability (before Christmas) of Mecanno's Spyke robot, which was announced at the last Consumer Electronics Show in Las Vegas, NV. But no one really knows as of this writing. The bot has stirred up considerable interest, though, because it connects to your PC via Wi-Fi, so you can control it from anywhere in the world via the Internet. It comes with a video camera, a speaker and mike, and motion detectors, so it is said to make an excellent surveillance bot.

Mecanno's Spyke robot was announced at the last CES, but has yet to emerge. Photo courtesy of Mecanno.

Spyke also acts as a Skype-compatible VOIP telephone and even comes with a built-in MP3 player. Apparently, you can build at least three different versions using the 210 parts that come in the box, and it will be compatible with existing Mecanno parts.

By the way, Mecanno (a French outfit) now owns the "Erector" brand name and uses it to sell its wares in the USA. However, the new Erector gadgets apparently are not compatible with the old Gilbert toys and are just repackaged Mecanno products. The Erector website (**www.erector-sets.com**) doesn't display any Spyke information, so the mystery will continue for now.

Agro-UAV Planned

We'll have to wait to see this one, too, but it was recently announced that the new Geospatial Research Centre (a partnership among New Zealand's University of Canterbury, the UK's University of Nottingham, and Canterbury Development Corp.) has taken up the task of developing an unmanned robotic airplane designed to allow farmers to collect data on animal health, crop and soil conditions, water uptake and water use. The Centre (**www.ict.canterbury.ac.nz**) is based at the New Zealand ICT Innovation Institute, a part of UC's College of Engineering. Trials are already underway with an unmanned aircraft fitted with a GPS, imaging systems, and communications equipment.

According to Centre Director David Park, "The range of actual physical environments that is available for research in the South Island within a few hours of Christchurch in terms of oceans, rain forest, glaciers, mountains, cliffs, and agriculture of all types, makes it all very exciting ... The idea is to develop a model which would retail for about $10,000 and which would be no more than a couple of meters in size, and packed with electronics and sensor devices."

The project has received $2 million in government funding and $900,000 in regional funding, but it should be self-supporting within 2.5 years.

Ugly Rumor Department

According to various sources, including Yahoo! Music, Michael Jackson has been looking into the concept of creating a 50-foot robotic replica of himself to tramp around in the Las Vegas desert, flashing laser beams and otherwise catching the attention of people flying into McLaren Airport. Apparently, Jackson intends to launch a comeback in Vegas. I'm just wondering if the Jackobot will detach and reattach its nose as it moonwalks across the sand. **SV**

GEERHEAD

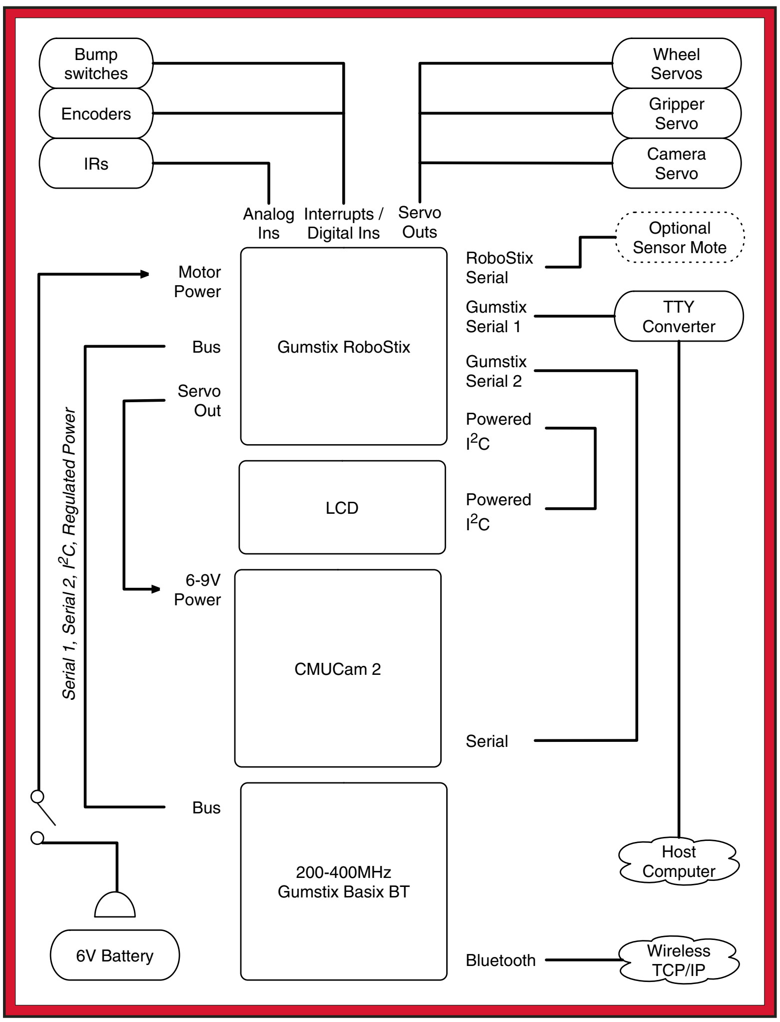
by David Geer

Contact the author at geercom@alltel.net

When Flockbots Flock, The Research Rocks!

Flockbots are tiny flocking robots forged from George Mason University's (GMU) Applied Robotics Club (ARC). The eventual purpose and functionality is experimentation with flocking behavior, similar to the swarm robots or other cooperative robot activities of which you may have heard.

FIGURE 1. Flockbot schematic as described in the text.



And, the Robostix Connected to the Thighbone!

Well, not really, but to understand Robostix and other components of Flockbots and how they integrate, check this discourse on the just completed schematic (see Figure 1).

Starting bottom center, the Flockbot schematic shows a 200 MHz Bluetooth "Gumstix" with a "Way Small" board from Gumstix (a vendor). "It's a 200 MHz Linux computer in the form factor of a stick of gum," says Randall S. Steck, president, applied robotics club, George Mason University. (These bots photograph much bigger than they are.)

The Gumstix comes with 64MB RAM, 4MB ROM, and Bluetooth on board. It also comes in a 400 MHz model (explaining the 200-400 MHz designation) though Steck and company went with 200 MHz, which was sufficient to meet their needs. "We log in to it on a command line and have a full suite of Linux tools available to us," details Steck.

Just above that is a CMUCam 2 mounted on a servo, which gives it pitch control (up and down, but not side-to-side). That connects over serial to the Gumstix/Robostix stack. The Gumstix handles communications with the camera. The camera does the color tracking.

"The CMUCam has an on-board processor; we can feed it six numbers representing the colors we want it to track. The processor forwards a tracking packet. The packet contains the central color location in the image; it

Photos are courtesy of Evan Cantwell, George Mason University.

also contains a pixel count and confidence level. Using those values, we can do some basic computer vision," Steck explains.

Above that is a 2 x 16 character LCD. "This lets us spit out some text to the LCD and get some idea of what is happening," says Steck.

The Robostix is above that. The LCD connects to the Robostix (a tiny robotics controller) via an embedded on-board connector. An ATMEL Mega 128 microprocessor is the foundation for the Robostix. It can handle 16 servos: eight analog and eight digital, according to Steck.

"We wrote our own custom firmware to run on the Robostix. That provides a link over I^2C (where you see serial connections). It monitors and updates the current position of the bots based on the wheel watchers," says Steck.

The Robostix pulls the wheel watcher data (see Figure 2), reads it, and updates the bot position, which the Robostix also calculates. The Gumstix can also request this data from the Robostix.

As Steck explains it, the Robostix maintains the physical connections to analog and digital I/O (pictured at top). When the Gumstix makes a request to read an analog sensor, the Robostix executes that, performs the conversion, and then sends the data back over I^2C to the Gumstix. "So, it's really a slave processor."

The Robostix also connects to four servos. Two are wheel servos (top right) that function as the drive train. There is a gripper servo (top right, below the wheel servos) for the gripper in the well of the Flockbot on the lower tier. It holds on to small objects like the half-sized pop cans in one of the ARC's demonstrations (the bots put different colored cans into different piles).

The final servo attaches to the bot's top plate or deck; this servo handles the camera. There are also five infrared sensors for basic distance readings around the bot (see top left). "Those are plugged into the analog boards," Steck notes.

"There are two simple bump sensors (top left), one for the top deck so we know when we bump into something, and one in the gripper well on the bottom half so we know when a can is properly collected in the gripper well," he adds.

The diagram also shows encoders, an optional sensor mote, Bluetooth wireless, motor power connection, data busses, and a 6V battery.

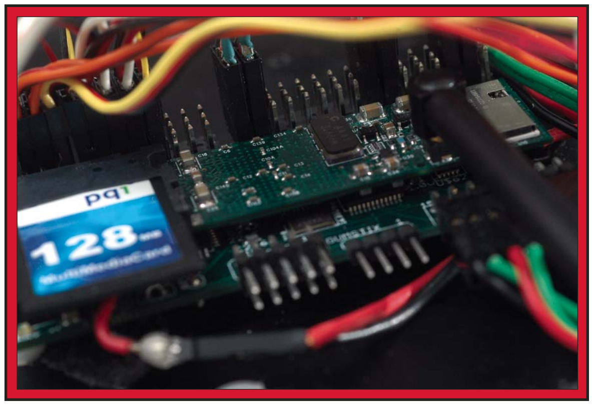


FIGURE 2. This is the Gumstix, with a 128MB MMC card (blue). The red and green wires on the bottom right are the I^2C connection to the LCD. Connections at the top, from left to right, are for the ADC (IR sensors), digital I/O (bump sensors), and the Wheel Watchers (two data and two interrupt connections).

Future Flockbot Interaction Gaining Traction

All the Flockbots use on-board Bluetooth for wireless intercommunications. A personal area network (PAN) connects them; it supports 15 devices per network. "We have a 15 unit PAN in the lab. We could run up to 15 Flockbots; we have eight now. We have desires to build seven more," Steck affirms.

While the ARC hasn't demonstrated any actual flocking or swarming with multiple bots yet, it has tested two bots using the same operating code locomoting in the same vicinity. They weren't coordinating with each other, however. That and more are coming very soon.

"We plan to have them communicating over Bluetooth on an ad-hoc network, rather than with one acting as a controller robot, because it breaks the symmetry and naturalness, if you will, of the solution," explains Steck.

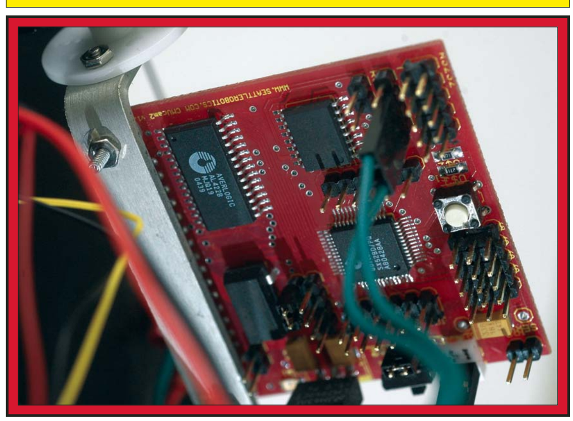
The eventual flocking behaviors will emerge out of what the ARC thinks to be very simple behaviors. Take, for example, a comparison from nature. As Steck pictures it, there is a desire for one creature — a bird — to remain an equidistance from its neighbors (in flight) and as soon as one of them turns, it changes that whole equation and it kind-of back propagates across the whole flock and they all make that turn because it's the only way they can all satisfy their mutual constraints (to preserve the equidistance).

Flocking, he continues, is a large scale behavior that emerges from smaller behaviors; "and that was, in some measure, our purpose: to build up these small behaviors in individual Flockbots and see how they worked with those in unison."

The bots will be able to make decisions based on shared information as in cooperative swarming. There are also things they can do without sharing data.

The Flockbots are going to

FIGURE 3. This is the back of the CMUCam. The green wires are data connections to the Gumstix.



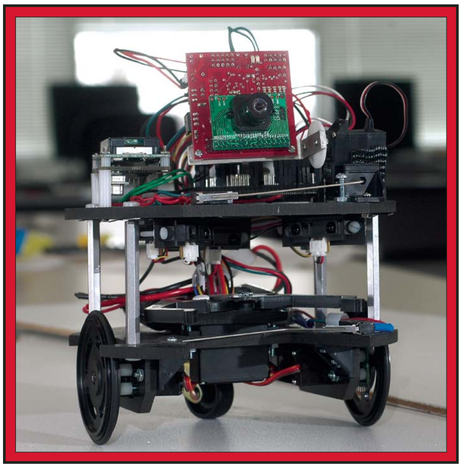


FIGURE 4. Front of Flockbot with gripper platform (bottom).

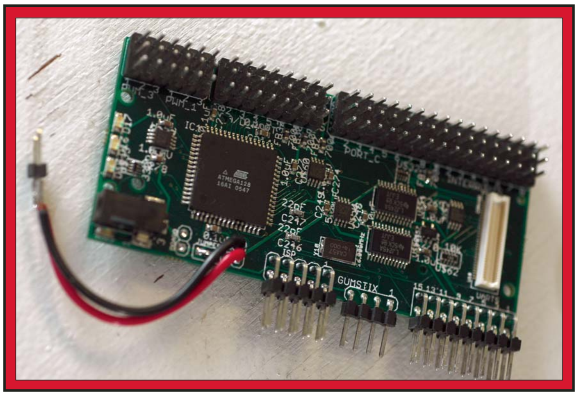
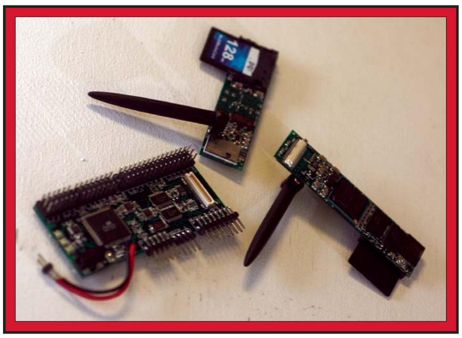


FIGURE 5. Robostix, top. Right is the Gumstix connector. Bottom pins left to right are for the ISP (programming), console, and I^{2}C/serial. The top pins left to right are for the 6x servo, 8x analog (ADC), 8x digital GPIO, and various interrupts and ports.

perform a kind of flocking behavior that is technically referred to as multi-agent simultaneous localization and mapping (SLAM). The bots will use their sensors to map the space that they share and then — using both sensors and the map — they will each determine their position in the map.

FIGURE 6. Gumstix and antennas.



Challenges

Challenges to mapping include situations where one part of the area being mapped looks like another part and the bots can't tell one from the other. "There are also advantages (in this kind of swarming) in multitasking, where the robots can go off in different directions," Steck says.

Two of the Flockbots will also be able to act as message relays. Because Bluetooth is a limited range technology and doesn't navigate corners and obstacles well, two of the Flockbots can "arch" themselves and provide that link around corners.

Each bot forwards sensor information back through the loop to every other bot via the Bluetooth. The data holds each bot's vantage point on the "world" they are in — the area they are trying to map. "The question then becomes: How do you coordinate those views; how do you align them one to another? It usually requires some outside arbiter. We are looking at a couple of ways of doing that," Steck leads.

One possible selection to remedy the bots' need for an arbiter is an MIT technology called Crickets. The indoor GPS, sonar, and radio wave technology performs the same functions and conventional GPS. The bots would broadcast their location, Steck explains, and a receiver unit on each bot would receive the broadcast from multiple bots and do the triangulation needed to determine the exact location of each bot in the three-dimensional space.

As another solution, GMU's lab will be outfitted with a "god's-eye"

ROBOTICS — THE CHAOS THAT IT IS

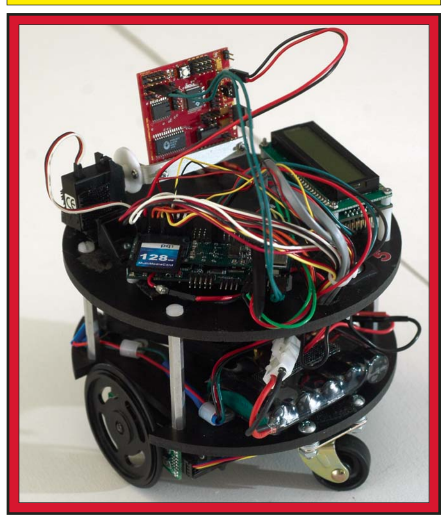
In the spring of 2005, Randall Steck (president of the Applied Robotics Club at GMU) and crew were just writing the first programs for the Flockbots. "We had these brand new robots with nothing programmed into them. We had to execute projects one after another to show basic capabilities (that the bots were put together correctly)," says Steck.

While Steck had worked with remote control and tele-presence before, autonomy was, as he puts it, "an entirely different ball game."

"The world is a lot more chaotic place than we ever thought possible when looking at it through the eyes of an IR sensor, a 255-pixel camera, and wheel watchers at 128 line resolution. We're operating with very limited information trying to pull off what seem to be very simple tasks like wall following and driving a straight line. It's amazing how non-uniform and chaotic the world is when you have to deal with all the little things that can go wrong with such limited information."

Robotics, Steck explains, requires looking at the problem from the perspective of "how would this work assuming that everything went wrong?" Steck and crew carved out solutions for each chaotic reality one at a time, dealing with exceptions and coming up with rules. "Then we had to adapt our software to the rules," Steck exclaims with wonder.

FIGURE 7. A Flockbot. The red square is the CMUCam; the LCD is to the right. The Gumsitx is at the bottom left.



camera, looking down from above at a small "playpen" of only about 8′ x 8′ (another reference to the comparably minute size of the bots themselves). The Flockbots will receive commands from the camera related to their exact XY position in the pen. They will know their relative location to every other bot because they will have their XY positions, as well.

"We will be able to see whether we can get them to go in the same direction (North, for example) simultaneously, or whether we can get them to get as far away from each other as possible," Steck says. The bots will also be able to use their current XY coordinates to move into the correct position.

Firmware and Software

The ARC team took a basic Linux character driver and modified it to develop their own simple command structure to the Robostix, to read the Analog-to-Digital data. Steck wrote a Java software library to communicate with the Linux character driver and the firmware.

"The Java library is designed for use with a variety of robots. We will eventually be using it to handle other robots in the lab," says Steck. These include a free-wheel holonomic robot, for example. The library will work with any robot equipped with the Gumstix Robostix stack that the ARC designed.

The next step is to test everything and move to software that demonstrates the bots' ultimate capabilities, like follow the leader and wall following. GMU underclassmen will design this software.

The next big steps, Steck continues, are to get the bots talking to each other over an ad-hoc network (the Bluetooth PAN) and then to write a communications protocol for information distribution between them. Finally, the ARC must determine the kinds of information the bots need to share, which will determine their problem solving capabilities.

"But, by next fall we should have at least two Flockbots working together," says Steck. **SV**

RESOURCES

Initial Flockbot schematic
www.cs.gmu.edu/~eclab/projects/robots/flockbots/uploads/Main/RobotSchematic.pdf

Flockbots homepage and Wiki
www.cs.gmu.edu/~eclab/projects/robots/flockbots/pmwiki.php

GMU Applied Robotics Club
www.cs.gmu.edu/~eclab/projects/robots/arc/pmwiki.php

Gumstix Way Small computing and circuitry
www.gumstix.com

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

roboto@servomagazine.com

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

ASK MR. ROBOTO

by
Pete Miles

Q. I want to build a two-legged walking robot. Right now, I am planning on using the 32 servo controller from Lynxmotion to control all the servos in my robot. It is controlled by a PC using an RS-232 serial cable, but the cable length is going to limit the range of my robot. Is there some way to control the servo controller via some sort of a wireless method instead of using the RS-232 cable? If so, can you tell me how to do it and what I need to purchase?

— Bill
Schenectady, NY

A. With all of the Bluetooth devices on the market today, this turns out to be a relatively simple and inexpensive thing to do. What you are going to need to obtain are two things: A Bluetooth dongle that connects to your computer and a Bluetooth serial modem that connects to the SSC-32 (serial servo controller from Lynxmotion; **www.lynxmotion.com**; see Figure 1).

To demonstrate how to do this, I have chosen the Bluetooth Modem — BlueSMiRF from Spark Fun Electronics (**www.sparkfun.com**) — to be the wireless device that will connect to the SSC-32. This is a direct wireless serial cable replacement that is ideal for robotic applications, especially smaller due to its size. Table 1 lists some of the specifications for the BlueSMiRF modem from the original equipment manufacturer and Figure 2 shows a photo of the modem. One of the nice features of this modem is that it uses a frequency hopping scheme that enables it to be used in harsh RF environments, which is needed with all the different 2.4 GHz devices operating around us these days.

On the computer side of things, I have chosen the Bluetooth USB Module (also from Spark Fun Electronics). This module — also known as a dongle — is about the same size as a standard USB memory stick, so it will work well with laptop computers. Figure 3 shows a photograph of the USB dongle. Make sure that you look inside the package cover for the half-size CD that contains the installation software.

Prior to connecting the BlueSMiRF modem to the SSC-32 and the USB Bluetooth dongle to your computer, you will need to install the software for the USB Bluetooth module on your computer. After the software is installed, plug in the USB Bluetooth dongle into the computer, and a "Welcome to Bluetooth" window will appear. Make sure that the "Use security level medium" checkbox is UNCHECKED, then press the OK

Size	1.9" L x 0.6" W x 0.15" T
Weight	2 grams
Antenna	Built-in
Output Power	12 dBm
Receive Sensitivity	-83 dBm
Range	100 m (300 ft) **
Frequency Range	2.4-2.542 GHz
Supply Voltage	3-6V
Current Draw:	
Receive, RX	40 ma
Transmit, TX	80 ma
Idle	1.4 ma
Average	25 ma

*** See discussion at the end of the article*

Table 1. Specifications of the BlueSMiRF Bluetooth serial modem.

Figure 1. The 32 servo serial servo controller from Lynxmotion.

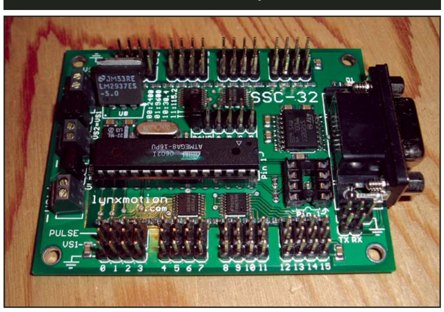
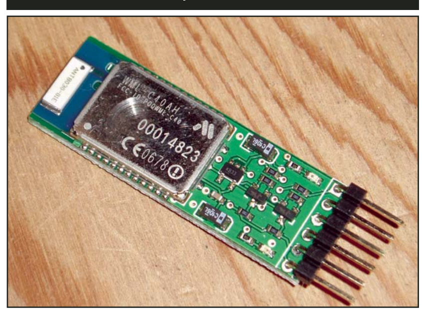


Figure 2. BlueSMIRF Bluetooth serial modem from Spark Fun Electronics.



button. Figure 4 shows the main window that will appear next.

Every Bluetooth device that is in range of this computer and that is recognized as a valid Bluetooth service, will be seen orbiting the "sun." Clicking on the sun will cause it to search for new Bluetooth devices. At this point, power up the BlueSMiRF modem (without connecting it to the SSC-32) with a +5V power source and power ground. Once the BlueSMiRF modem is powered up, click on the sun and after a few moments, an icon with the name "Spark Fun - BT" will be in orbit (see Figure 5).

To establish a connection between your computer and the BlueSMiRF, you will need to right-click the Spark Fun - BT icon, select the Connect menu item, and click the Bluetooth Serial Port Service sub-menu item (see Figure 6). Once the connection is made, you will see a dotted line appear between the Spark Fun - BT icon and the sun. The Bluetooth tutorial section on the Spark Fun website provides more information about setting up the Bluetooth software.

There are two LEDs — green and red — on the BlueSMiRF modem. When the modem is first powered up, the green LED will blink with a period of about 2 Hz. The blinking green LED means that it has power, but it is not communicating with anything else. When the connection is made, the green LED will turn off, and the red LED will turn on.

Now connecting the BlueSMiRF to the SSC-32 is a straightforward process. The first thing you need to do is remove the two jumpers on the SSC-32 that are connecting the two TX pins and the two RX pins together, which are located next to the DB-9 connector (the RS-232 cable connector; see Figure 7). Next, using a three wire cable, connect the RX-I pin on the BlueSMiRF to the TX pin on the SSC-32 and connect the TX-0 pin on the BlueSMiRF to the RX pin on the SSC-32. The third wire connects the grounds together on both devices. Always remember, connect the TX (transmit) line on one device to the RX (receive) line on the other device. Since the SSC-32 doesn't use flow control in its serial communications, you will need to connect the CTS-I and RTS-0 on the BlueSMiRF together. If you don't do this, the modem won't work. Table 2 lists the wiring connections, and Figure 8 illustrates how to physically wire the two devices together.

Figure 3. Miniature Bluetooth USB dongle/module.

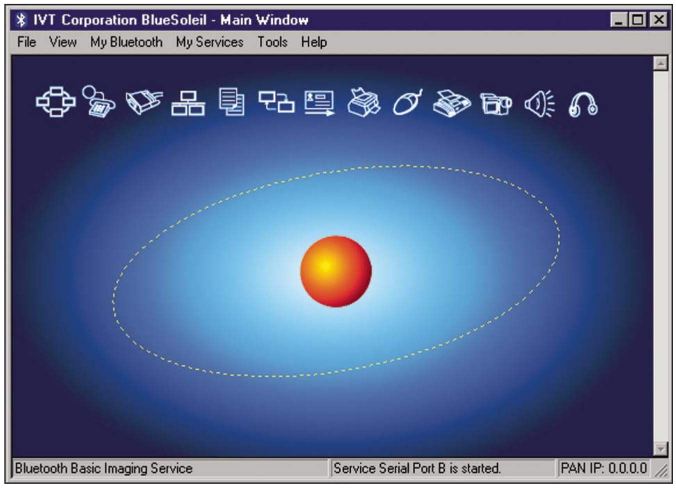


Figure 4. Main connection window for the BlueSoleil software.

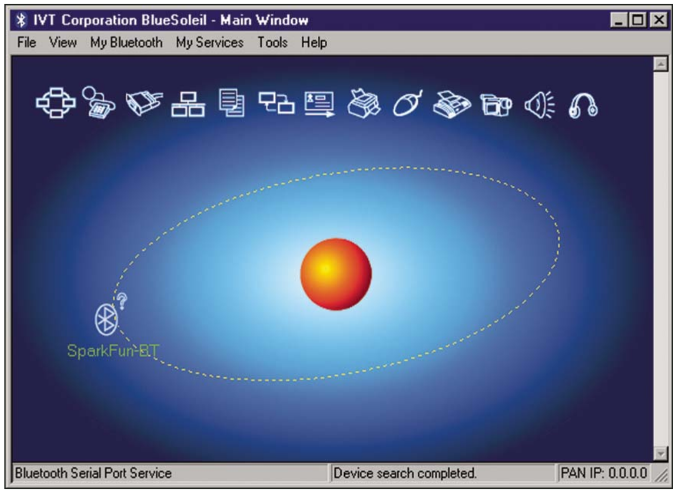


Figure 5. BlueSMiRF modem has been found.

Figure 6. Connecting to the BlueSMiRF modem.

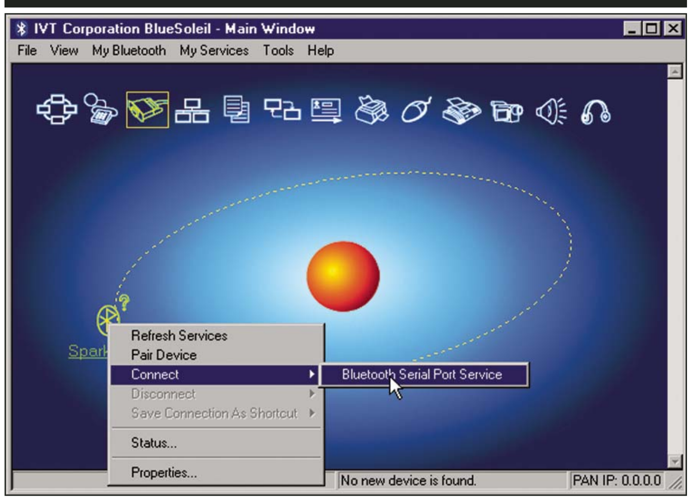
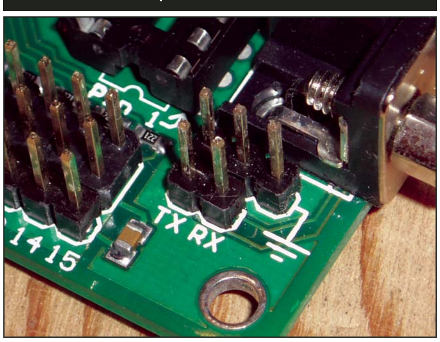


Figure 7. Close-up view of the TX and RX pins on the SSC-32.



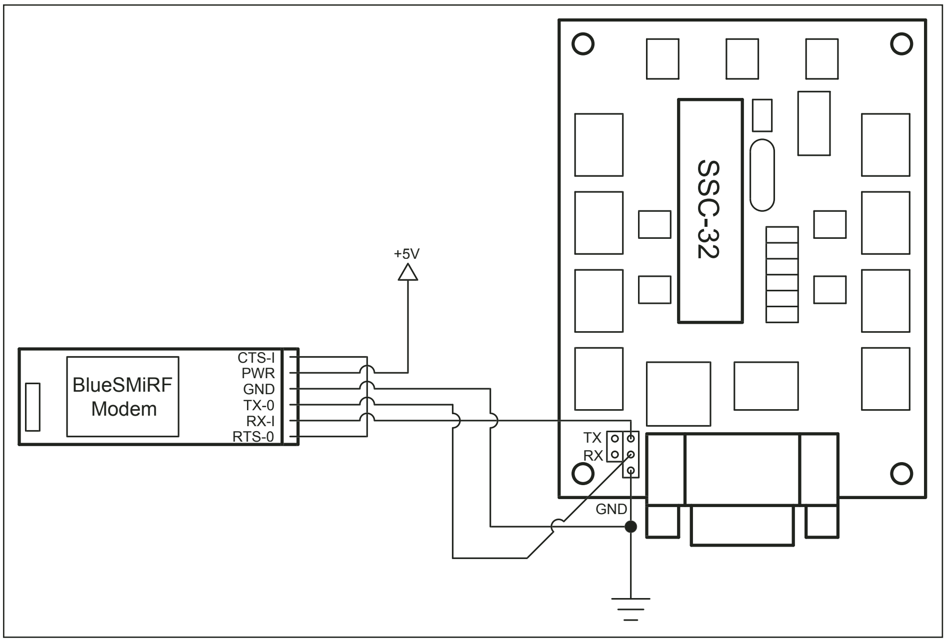


Figure 8. Illustration of wiring the BlueSMiRF to the SSC-32.

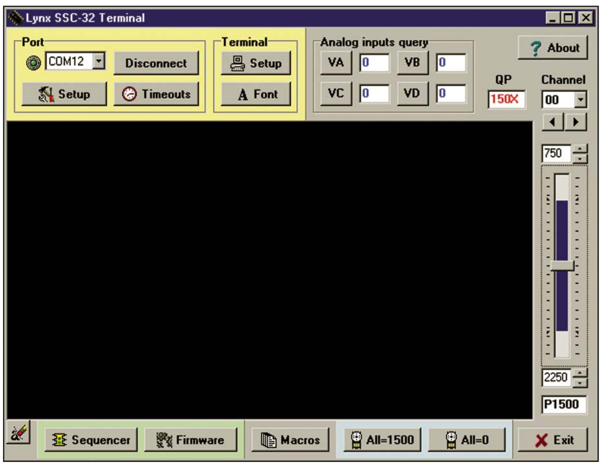


Figure 9. Lynx SSC-32 terminal program, version 1.05.

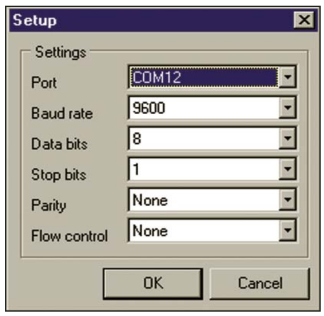
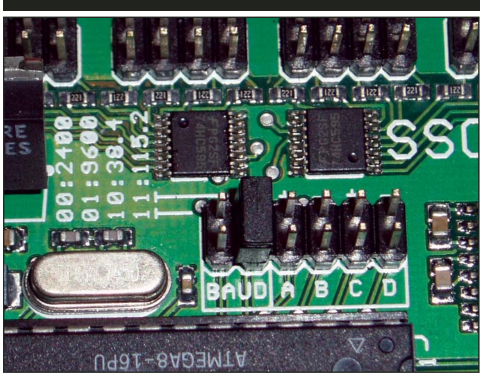


Figure 10. Initial COM port setting for communicating with the BlueSMiRF (port COM12 will be different on your machine).

Figure 11. Jumper settings on the SSC-32 for 9600 baud serial communication rate.



The SSC-32 does not provide any regulated power outputs to power any external devices, so you will need to provide your own regulated 5V power for the BlueSMiRF modem. Do not use the servo power from the SSC-32 to provide power to the BlueSMiRF modem. This may result in unreliable operation due to voltage dips and spikes.

I don't know what software you are planning on using to control the SSC-32, so I will demonstrate how to set up and run the Lynxterm software that is available for free at Lynxmotion. Power up the SSC-32 and the BlueSMiRF modem, establish a communication connection with the USB Bluetooth module, and then start the Lynxterm application. Figure 9 shows the main window for the Lynxterm program. Next, you will need to determine which COM port the USB module is operating on. To do this, right click on the Spark Fun - BT icon on the BlueSoleil program, and then click on the Status menu item. This will show you which COM port you are connected to.

The default baud rate for the BlueSMiRF modem is 9600 bps. So in the Port window in the Lynxterm program, click the Setup button, and change the COM settings to 9600 baud and change the COM port to the actual port your USB module is located. Figure 10 shows what all of the settings needs to be for the initial communication. Note: The COM12 is the COM port my Bluetooth Module was connected to for this demonstration. Your system will most likely be different.

The default baud rate on the SSC-32 is set to 115,200 bps, since the two jumpers are pre-installed on the SSC-32. This needs to be changed to 9600 baud by removing one of the jumpers. See Figure 11 for a close-up view of the proper jumper setting for 9600 baud. The SSC-32 manual explains how to set other baud rate speeds. If the baud rates for the SSC-32, the BlueSMiRF, and the Lynxterm programs are not all operating at the same speed, then the whole system won't work properly.

Since the BlueSMiRF modem doesn't locally echo the characters sent to it, you will have to adjust the Terminal setting of the Lynxterm program to see the characters you type on the screen. Figure 12 shows what the terminal settings need to be to communicate with the BlueSMiRF modem. The

BlueSMiRF	SSC-32	Notes
CTS-I		Connect to RTS-0 on BlueSMiRF
PWR		Connect to +5V Source
GND	GND	Connect to GND of PWR Source
TX-0	RX	
RX-I	TX	
RTS-0		Connect to CTS-I on BlueSMiRF

Table 2. BlueSMiRF to SSC-32 wiring connections.

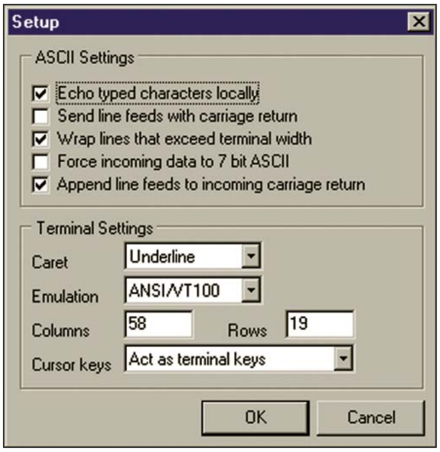


Figure 12. Terminal setup for communicating with the BlueSMiRF modem.

AT Command	Description
+++	Enter Configuration Module
AT	The Attention command prefix, should return OK
ATMD	Exit out of configuration mode
ATVER, ver1	Get BlueSMiRF firmware version
ATSI,1	Status Information: returns Bluetooth address ID
ATSI,2	Status Information: returns modem name
ATSI,8	Status Information: returns current baud rate in HEX
ATFRST	Reset factory defaults
Changing Baud Rate: ATSW20,10,0,0,1 ATSW20,39,0,0,1 ATSW20,157,0,0,1 ATSW20,472,0,0,1	 Change to 2400 Baud (10 ASCII) = (0A Hex) Change to 9600 Baud (39 ASCII) = (27 Hex) Change to 38,400 Baud (157 ASCII) = 9D Hex) Change to 115,200 Baud (472 ASCII) = (1D8 Hex)

Table 3. BlueSMiRF modem configuration and status commands.

Lynxterm program will still be able to control the SSC-32 with different terminal settings, but you won't be able to change the internal configuration settings on the BlueSMiRF modem itself if the settings are different from what is shown in Figure 12. Once all the settings have been adjusted, press the Connect button, and you should be ready to go.

The first thing you should do with the modem is verify that you have good communications between the BlueSMiRF and the SSC-32. On the Lynxterm program, type the word "ver" and press the carriage return key. When you are doing this, you should see the green LED on the SSC-32 servo controller blink after each time a key is pressed on the keyboard. After you press the carriage return button, you should see "SSC32-1.06XE" on the screen (this may change slightly depending on which firmware version you have in your SSC-32). If you don't see anything or some garbage characters, then there is a communications problem. Check your settings or wiring.

After completing all these steps, you are ready to install some servos and servo power to the SSC-32. If all is working right, you will be able to command the servos to move with the Lynxterm software.

If you are controlling a lot of servos at one time, you will probably want to increase the baud rate on the BlueSMiRF. To do this, you will need to make a configuration change in the BlueSMiRF modem.

Now, to change any of the configurations in the BlueSMiRF modem, type "+++" followed by a carriage return to enter the configuration mode. When you do this, you should see OK on the Lynxterm window. While you are in the configuration mode, you will not be able to control the SSC-32. To exit out of the configuration mode, type "ATMD" followed by a carriage return; you will get an "OK" on the screen, and the Lynxterm program will have control of the SSC-32 again. Table 3 lists a small set of the many configuration commands the BlueSMiRF modem has. A complete user guide for changing configurations can be obtained from Spark Fun Electronics.

It is recommended that before making any configuration changes to the BlueSMiRF modem, test that you can enter and exit the configuration mode by typing "+++" then "ATMD" (with a carriage return after each command). You should see an OK after each command. If you don't get this, then there is a communication problem. Remember, the default baud rate for the BlueSMiRF modem is 9600 baud. If you ever forget what you changed the baud rate to, type the ATSI,8 command to see the current setting.

Now for a short discussion on effective communication range. The BlueSMiRF modem is a Class 1 Bluetooth device, which means it should have a maximum range of 100 meters (330 feet) or so. But there are many factors that will affect the actual range that you will obtain. Factors include other 2.4 GHz RF devices operating in the same area, microwave ovens (they operate at 2.45 GHz), obstructions between the transmitter and receiver like walls and doors, and the size of the room. In addition to these, the type of antennas both devices are using, the orientation of the antennas relative to one another, and output power and receiver sensitivity differences between the two devices can factor in.

In my testing, I was only able to get about seven meters of direct line-of-site communication between the USB dongle and the BlueSMiRF. This was surprising since both devices are Class 1 devices. After reviewing the specifications for the USB module further, I found that its maximum output power is only 4 dBm, which (according to Table 4) is the maximum output power for a Class 2 device. It would then have a maximum range of 10 meters. I don't know why the USB dongle packaging advertises that it was a Class 1 device and then lists a contrary maximum output power of 4 dBm.

To make things at bit more

Device Class	Maximum Output Power	Maximum Range
Class 1	100 mW (20 dBm)	100 m
Class 2	2.5 mW (4 dBm)	10 m
Class 3	1 mW (0 dBm)	1 m

Table 4. Bluetooth output power classes.

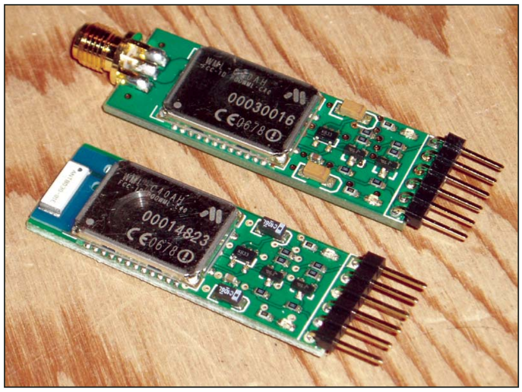


Figure 13. BlueSMiRF modems with built-in ceramic antenna (left) and SMA external antenna mount (right).

interesting, I changed the regular BlueSMiRF modem with a BlueSMiRF RP-SMA modem with a 2.4 GHz duck antenna. Both of these modems are identical, with the exception that the first one has a built-in ceramic chip antenna and the other has an SMA external antenna mount (see Figure 13). With everything else being the same, the transmission range increased to well over 15 meters (the maximum range wasn't tested since I didn't go outside the house to increase the testing range). Though the USB dongle's output power wasn't increased, adding the antenna to the BlueSMiRF improved its reception sensitivity, thus effectively increasing its range.

When long range transmission requirements are needed, the components with the lowest output power rating need to be able to transmit over that range. Otherwise, there may be reliability issues in the data link. In this example, both devices were advertised to be Class 1 devices and advertised to have 100 meter ranges, but in actual testing, the advertised output power rating provided a better indicator of the communication range between the two devices.

A lot of information was presented here, which might give the impression that putting together a wireless robot control system is complicated, but in reality, with the right components, it is almost a plug-and-play. The BlueSMiRF modems are simple and easy replacements for serial cables to put together a wireless communication link. You will probably spend more time reading this article than setting this system up. Here are some interesting notes about the SSC-32 and the BlueSMiRF modem that can help diagnose setup problems:

- The active baud rate on the SSC-32 is determined by the jumper settings at power-up. Changing the jumpers after power-up doesn't change the baud rate settings on the SSC-32.

- When a physical RS-232 serial cable is connected to the SSC-32, the servos will not move if the baud rate settings between the Lynxterm program and the SSC-32 don't match. (Obvious, but see what follows.)

- When the baud rate settings of the BlueSMiRF modem and the SSC-32 are matched at power-up, the baud rate setting on the Lynxterm program can be anything you want it to be, and you can still control the SSC-32.

- When the baud rate settings of the BlueSMiRF modem and the SSC-32 do not match at power-up, the servos will not move regardless of what the baud rate settings are on the Lynxterm program. But, the internal configuration of the BlueSMiRF modem can still be changed by the Lynxterm program. Note also, the green LED on the SSC-32 will still blink as it receives data. It just won't process it.

- After the SSC-32 is powered up, changing the BlueSMiRF modem's baud rate to match the baud rate of the SSC-32 will enable the Lynxterm program to control the servos.

- After power-up, changing the BlueSMiRF modem's baud rate to something that is different from what the SSC-32 is set at disables all servo motion control. The LED on the SSC-32 will not blink when receiving commands.

- Baud rate changes on the SSC-32 require a power-reset to take effect. **SV**

EVENTS CALENDAR

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

Usually my participation in robot competitions is limited to watching them, reading about them, and collecting the event information for this list. So I was a bit surprised recently to find myself volunteering to plan a last-minute outdoor robot competition. The contest had to be tailored to the particular robots that would likely show up and to the available contest space, which turned out to be a relatively flat vacant lot. There were no interesting obstacles or hazards and the only variation was the choice of areas covered in gravel or grass.

The contest ended up consisting of three waypoints. Robots started at waypoint one and traversed a completely uncluttered gravel path to waypoint two. Then they had to negotiate alternating gravel and grass to reach waypoint three. The final leg was a return to the starting point across a path covered in medium high grass. Each waypoint was specified by GPS coordinates, distance azimuth, and a visual marker. These three options allowed a wide range of robots to participate.

Getting all this done involved multiple visits to the site, assisting a friend who took the GPS measurements for waypoints, assembling rules, descriptions, and photos. Overall, it was a good reminder of how much work goes into planning and executing even the simplest robot contest. An astonishing amount of work by robot group members goes into the much more elaborate contests listed in this events calendar. If you participate in one of these events, don't forget to thank the staff for all their hard work!

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

June

2 **ION Autonomous Lawnmower Competition**
Dayton, OH
In this event, robot lawnmowers are judged on speed and accuracy.
www.automow.com

2 **Milford Autonomous Robotics Competition**
Milford, OH
Several events for autonomous robots include mini Sumo, standard Sumo, and robot rescue/object retrieval.
www.robocap.org

8-11 **AUVS International Ground Robotics Competition**
Rochester, MI
In this event, university-built autonomous ground robots navigate an outdoor obstacle course.
www.igvc.org/deploy

14-17 **FIRA Robot World Cup**
San Francisco, CA
The world cup of robot soccer. There are leagues for all sorts of robots ranging from kheperas to Aibos to humanoids.
www.fira.net

15-17 **RoboGames**
San Francisco, CA
This is the world's largest robot competition with robots from around the world competing in more than 70 events.
www.robogames.net

22-24 **MATE ROV Competition**
St. John's, Newfoundland, Canada
High school and college teams build ROVs that must complete a different mission each year. See the website for details on this year's mission.
www.marinetech.org/rov_competition

30 **UK National Micromouse Competition**
Birmingham, United Kingdom
Micromouse builders compete for the coveted Brass Cheese. If you've never seen a Micromouse event, try to make it to this one. These are amazing little robots.
www.tic.ac.uk/micromouse

July

1-10 **RoboCup Robot Soccer World Cup**
Atlanta, GA

All the usual soccer events: small, mid, humanoid, and AIBO. Also a NIST rescue robot contest. In addition to these events, the RobotCup@Home competition will be held in conjunction with the World Cup again this year.
www.robocup.org

10-13 Botball National Tournament
Norman, OK
Teams compete with autonomous robots built from standardized kits. The contest involves moving black and white balls on a game board.
www.botball.org

11-15 AUVS International Undersea Robotics Competition
US Navy TRANSDEC, San Diego, CA
Autonomous underwater robots must complete a course with various requirements that change each year.
www.auvsi.org/competitions/water.cfm

16-20 K'NEX K*bot World Championships
Las Vegas, NV
Includes three events: Two-wheel drive K*bots (autonomous), Four-wheel drive K*bots (autonomous), and Cyber K*bot Division (R/C).
www.livingjungle.com

21-22 War-Bots Xtreme
Saskatoon Saskatchewan, Canada
Radio-controlled vehicles destroy each other Canadian-style.
www.warbotsxtreme.com

NEW PRODUCTS

ACCESSORIES

Tiny, Tough, and Chemically Inert

A full line of precision miniature sapphire and ruby balls for use as check valve elements, bearings, spacers, and other applications requiring a highly spherical, wear-resistant, and chemically inert ball are available from Meller Optics, Inc.

Meller Sapphire and Ruby Balls are offered in 42 different inch and metric sizes from 0.1 mm to 1/2" dia. and feature 0.000025" (AFBMA Grade 25 tolerance) per ball sphericity. Exhibiting Moh 9 hardness, they are ideally suited for use as check valve elements, bearings, and spacers in scientific and medical devices.

Impervious to common acids, alkalis, solvents, and bodily fluids, Meller Sapphire and Ruby Balls are fracture-resistant and the red ruby balls are easier to see than the clear sapphire, where high visibility is important. They have finishes of 20-10 scratch-dig or better and can be precision ground and polished into inexpensive plano-convex lenses, if required.

Meller Sapphire and Ruby Balls are priced according to material, size, and quantity; delivered from stock. Samples, literature, and pricing are available upon request.

For further information, please contact:

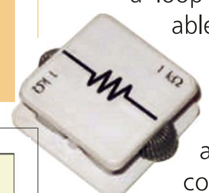
Meller Optics, Inc.
PO Box 6001
Providence, RI 02940
800•821•0180
Fax: **401•331•0519**
Web:
www.melleroptics.com

ELECTRONICS/ CIRCUITS

SpringCircuits

RMD3 announces the launch of its product line — a set of electronic circuit elements placed in modular insulating housings with bent-spring connectors.

The circuit element housing is called SpringCircuits "piece." The bent-spring connectors make for fast connect and disconnect with other pieces. The laminated graphic label on each piece depicts which circuit element is within the housing and allows for dry-erase marker notations on the circuit. A hook fastener at the bottom of each piece allows for placement on a loop-fabric board (also available from RMD3) and portability of circuits built with SpringCircuits.



RMD3 also offers a traditional breadboard compatible with Spring Circuits which is mountable on the loop-fabric board with the SpringCircuits pieces. More circuit elements are available at the website.

SpringCircuits is suitable for hobbyists, enthusiasts, educators, and even professionals who need to throw together a circuit on-the-fly or make a presentation.

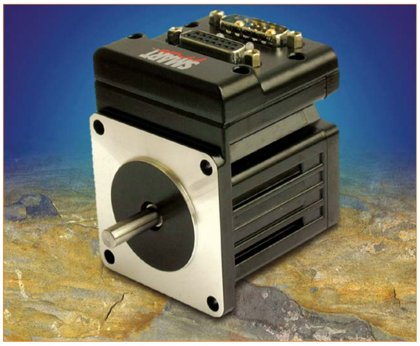
The SpringCircuits Starter Kit includes 31 pieces, six jumper wires, a loop-fabric board, a battery pack, and dry-erase marker and costs $125 plus shipping and handling. Orders are taken at the RMD3 website.

For further information, please contact:

RMD3
Website:
www.rmd3.com

MOTORS

SM2315DT — Double Torque SmartMotor



Animatics Corporation — designers and manufacturers of highly innovative motion control products — announces the latest release in their OEM series of SmartMotors: the new SM2315DT. The SM2315DT features an innovated new eight-pole compact rotor design with higher copper filling factor compared to conventional techniques and the new design achieves a higher energy density, as well as better efficiency that results in more torque capacity in the same physical package size. "Animatics can generate two times (2X) more continuous torque than the world's best selling integrated servo system, our very own SM2315D," says Hack Summer's, Senior Application Engineer for Animatics. The OEM Series SmartMotor™ is a complete motion control system that is known for being compact, user friendly, and cost effective. Using a patented design, SmartMotors incorporate a servo Motor, Amplifier, and motion controller into the same integral frame.

Similar to other members of the OEM series, SM2315DT is unique by virtue of its extreme low cost and high continuous and peak output torques. It delivers as much as 112 oz-in of peak torque while maintaining as much as 57 oz-in of continuous torque with a maximum speed capability of 4,700 rpm. The entire line of SmartMotors is priced competitively for high quantity, OEM applications. Featuring standard NEMA 23 frame dimensions, the SM2315DT will meet unique customer requirements with a very familiar package. For further information, please contact:

Animatics Corporation
3050 Tasman Dr.
Santa Clara, CA 95054
408•748•8721
Fax: **408•748•8725**
Email: **sales@animatics.com**
Web: **www.animatics.com**

Show Us What You've Got!

Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:

newproducts@servomagazine.com

COMBAT ZONE

Featured This Month

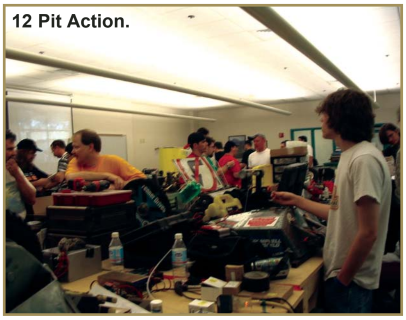
PARTICIPATION

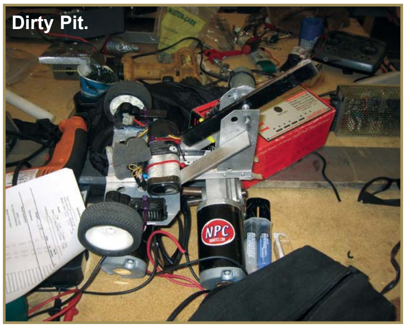
Clean Pit/Dirty Pit — Safety First!

● by Kevin Berry

Virtually all injuries at events occur in the pits, not in the arena. Frantic rebuilding activities on torn metal and shredded plastic with power tools result in cuts, scrapes, and splinters. Other contributors to pit injuries are clutter, cramped quarters, and shared resources.

To be fair to event organizers (EOs), I usually have had enough space — a whole table to myself, plenty of open area around it, and a dedicated power source. Any clutter is purely my own fault. At Battle Beach 2 — due to my own haste — I laid a hot soldering iron onto the AC power cord for my battery charger. Luckily, my son noticed the smoke and yanked it off before we developed a "hot short" and possibly a fire. Even though quarters there were amazingly cramped, this was purely my fault for not watching. At the same event, I unplugged what I *thought* was my battery charger. Turns out it belonged to the team across the table, sending their batteries into a rapid discharge mode. I found that one myself, killed their charger, and humbly begged forgiveness. That


12 Pit Action.


Dirty Pit.

taught me to "tag" all my power cables at the plug with a piece of masking tape with my team name on it.

At one glorious event — Robot Rebellion — I had a whole, 15' pit table to myself. I laid out my tools neatly and ran the most organized repair activity ever. I didn't have to watch for someone behind me, could swing around with a hot iron or long drill bit without skewering a neighbor, and never confused my leads and power cords. Once in 20 events isn't too bad, I guess!

So the short message for good pit safety: Neatness counts! **SV**

Rise of the Kilobots

● by Kurtis Wanner

Two robotic combatants stood poised, ready to annihilate each other. The announcer yelled "FIGHT!" and the bots inched towards each other, respective weapons slowly getting up to speed. One crossed the 4 ft x 4 ft fighting surface while the other sat spinning, apparently having drive problems. The stationary robot "Whirlwind of Doom" (Photo 1) was donned in cardboard armor made from a cereal box and sported an alternator fan blade. Its opponent "Robopope" (Photo 2) had small R/C car motors friction-driving a large overhead aluminum blade. The bots finally met and blades collided with a tinny clang. Both bots tipped backwards and waited for their weapons to spin up so they could attack again.

The match ended without more than a scratch to either bot, but the crowd was cheering wildly. It was the most action they had seen all afternoon. The event was five robots in a round-robin tournament that only lasted a few hours ... not too exciting by today's standards, but everything has to start somewhere!

It was March 2003 when the Kilobots premier event took place (Photo 3). It was a small room in the engineering building at the University of Saskatchewan. Four years later, the Saskatoon Combat Robotics Club (SCRC) has just completed its tenth event, "Kilobots X." It has grown from five entries in the single weight class of 1 kg kilobots to four classes with over 50 bots.

That first event started the ball rolling. The word was out that there was a combat robotics event in central Canada. Emails started trickling in; most from local people, but some from out of the province, as well. The second event was held at an art gallery in conjunction with their "Cyborg" display. The audience draw was greatly underestimated as two hundred spectators tried to crowd into a room that should have held half as many. The name Kilobots was becoming better known, as television interviews were aired and a full-page article in the Sunday newspaper was printed.

The third Kilobots event was a huge step forward, as the SCRC debuted an upgraded arena at "Spectrum 2004," Canada's largest student-organized science and engineering expo (Photo 4). Over 10 thousand spectators passed through the Kilobots room, with crowds swelling to over a hundred at a time during the last two of four days. At the conclusion of Spectrum, the committee informed the SCRC that Kilobots was the most popular display in the past few expos, and invited the event back in four years.

Meanwhile, another combat robot event was being born in Saskatoon: War-Bots Xtreme. WBX already had a steady foundation of combat robot builders to grow from. Its premier event boasted the largest arena ever seen in Canada with robots weighing up to one hundred times more than the kilobots. A number of big-time teams crossed the border to fight for the huge prize purse. There were the usual hiccups for a premier event, but all in all, WBX was a success and has proceeded to hold one event a year since then. The SCRC joined up for WBX-II and organized the 1 lb, 1 kg, 3 lb, and 6 lb weight classes alongside WBX's 30 lb, 60 lb, 120 lb, and 220 lb behemoths.

Four years since the very first Kilobots event, the SCRC was back in the U of S engineering building — the largest and once again most popular display at Spectrum 2007, Kilobots X (Photo 5). Two long days of school demonstrations preceded this tournament. The two drivers who were now waiting for the lights to turn from red

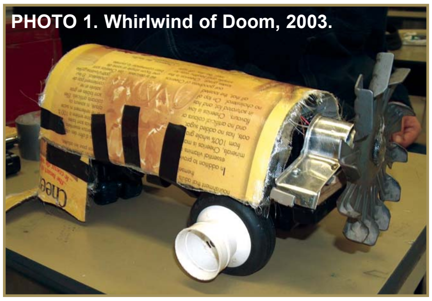


PHOTO 1. Whirlwind of Doom, 2003.

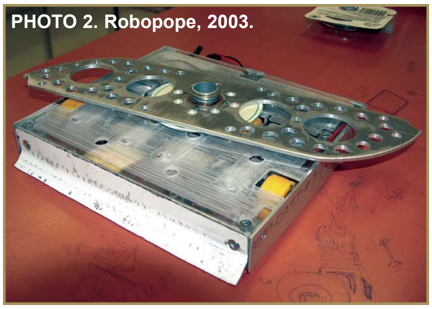


PHOTO 2. Robopope, 2003.

PHOTO 3. Builders relaxing before the premier Kilobots event, March 2003.

PHOTO 4. School demonstrations in an upgraded arena at Spectrum 2004.


PHOTO 5. Three years later, demonstrating to another school group, Spectrum 2007.


PHOTO 6. Swiss Chef, 2007.


PHOTO 7. Underkill, 2007.


PHOTO 8. Sparks fly as Swiss Chef maneuvers around the deadly blade of Underkill.


PHOTO 9. Moments before the fight ends, the crowd cheers as a front support is ripped off of Swiss Chef.

to yellow to green had spent those days teaching the finer points of robot combat to thousands of elementary and high school students. At least three all-night build sessions in a row had pushed these drivers into a trance-like state, yet both were intently focussed on their opponent and the strategies they had seen in events past.

In the blue square stood Swiss Chef, the fifth incarnation of a robot that had won second place at the premier event (Photo 6). Numerous events had honed the driver's skill at keeping the "drumless drum" weapon aimed at the opponent, but for this match, Swiss Chef was doing just the opposite — keeping the rear of the bot towards the enemy — because it was covered by a new flexible titanium guard designed specifically for this match-up.

Underkill sat in the red square, eagerly awaiting its chance to fight the longest surviving kilobot. Its horizontal titanium blade gleamed under the spotlights of the arena (Photo 7). Underkill was known for tearing its way through the competition, but spun so fast that bots with hard armor prevented it from getting a good bite. No one was sure how this match would end.

The lights went green and both bots attacked each other with speed and skill. Sparks rained down inside the arena when Underkill crashed against the rear armor of Swiss Chef. Each bot turned and tried to get position on the other, and met again in a shower of sparks (Photo 8). Swiss Chef managed to push the blur of the horizontal blade into the arena's steel barricade, momentarily stopping the weapon. Before Swiss Chef was able to bring its sharpened drum teeth around into the plastic sides of its trapped enemy, Underkill backed away from the wall, instantly revved up its blade to 12,000 rpm, and spun around into the front support of Swiss Chef, snapping it off and sending it across the arena (Photo 9).

The drum was now grinding against the steel floor, making the whole bot hop around erratically. Now lacking the ability to steer and push, Swiss Chef faced the enemy with drum spinning at full speed, balancing without the use of the front support. Both bots met weapon to weapon, spinning so fast that they would not engage. Underkill's blade finally caught Swiss Chef's drum and sent the wounded bot careening into the corner of the arena, where it would do nothing but roll back and forth in a semicircle. Having nothing left to fight with, Swiss Chef was counted out. Underkill went on to win second place, taking the spot that its opponent had claimed four years earlier.

If you are interested in seeing these bots and more, be sure to check out the next events, WBX IV (July 20-21, 2007) and Kilobots XI (planned for September 2007). Event reports, photos, and videos can be seen at **www.Kilobots.com**. For more information, contact Kurtis Wanner at kwanner@fingertechrobotics.com **SV**

VIDEOTAPING THE ACTION

● by Robert Woodhead

*While Robert Woodhead isn't a professional videographer, he did found AnimEigo, a company that releases Anime and Samurai films in the United States. So it was only natural that he should start videotaping his favorite hobby — combat robotics. His "Metal Munching Maniacs" DVD series, available on **Amazon.com**, is a "must see" for combat fans. SERVO bribed him to reveal the tricks of getting good footage of bad robots.*

Lights!

Cameras are not as good as eyes. You want as much arena lighting as you can get, and then some. Most events use cheap halogen work lights; these work quite well. Put an even array of lights around the arena, plus add some extras that you can reposition as needed (say, when some shrapnel takes out a light). Extra light in the corners (which will tend to be far from most of the cameras) is usually a good idea.

If you want to do a colored light effect to highlight the starting squares, just go get some translucent colored report covers from Cheap-Mart and tape them over the grillwork that comes with the work light. Buy a few extras, as they do tend to get a bit melty!

Plan the lighting with the event organizer beforehand. Make sure you have enough juice to power the lights and keep the exhaust fan on inside the arena all the time to reduce the "easy-bake oven effect." And never shine a light directly at one of the camera positions.

Cameras!

Get the best cameras you can afford. I used the first generation of cheap "three-chip" miniDV camcorders (one chip per color equals a better picture), which came out at about $800 each. These days, you can get similar cameras for even less, but I'd probably go for one of the entry-level HD camcorders, even a single-chipper, because when downconverted to DVD format, it'll still look better that a good miniDV, which is standard definition.

You should also get a neutral-density filter (to protect your camera's lens) and a small metal conical lens hood (which will come in handy later). Finally, don't be cheap on your tripods! Spend a couple of hundred and get a decent, entry-level fluid-head adjustable tripod. They will make your camera moves look professional.

The #1 thing you *must* insist on is fresh, virgin Lexan in front of your cameras. This is absolutely essential. If you can't get it, you'll spend hours trying to find a "window" through the scratches. Most event organizers buy some new Lexan for each event so they can rotate out the worst panels. Insist that this new Lexan goes in front of your cameras.

Camera positioning is also an important decision. The prime locations are the panels next to the corner panels. This lets each camera sweep most of the arena, including getting a close-up of the corner it is next to; it can see three of the four corners. Also, this keeps the cameras away from the prime human viewing areas, which are in the center of the sides.

If I were building an arena myself, I'd consider making it octagonal, with one-panel wide corners; cameras in the corners would then be able to sweep the entire arena.

I have sometimes placed a camera inside the box. The best place is high in a corner, with a wide-angle adapter lens so it gets most of the arena. Smart people will build some sort of Lexan shield for in-box cameras, but I never had the time, and it would require some sort of custom mount. Instead, I use some perforated metal strip as a mount, and good thoughts for armor. This worked well, but at the 2004 Nationals, a shrapnel hit ripped the wide-angle lens off the camera, but the lens gallantly sacrificed itself to protect the camera. So use a cheap camera for this placement!

In order to eliminate glare reflections, you need to wrap each camera in a light box. Here's how to build a low-cost one. You will need a couple of closet rod brackets, an extendable curtain rod, some black blackout fabric (often sold as "backpack fabric" at the fabric store; it's heavy and 100% opaque), cable-ties, black gaffer tape (not duct tape if you can avoid it, since it can leave a residue on the Lexan), and a fabric punch that lets you punch small holes in the fabric.

Attach the two brackets to either side of the Lexan panel supports about 8" above where the camera lens is going to go. Put the curtain rod in the half-moon bracket elements, wrap it in gaffer tape to make it black, and tie it down. Lay some of the fabric on top of the curtain rods, with excess going off on both sides (you'll have to experiment with this before you cut the fabric to find out exactly how much you need). Use the punch to make holes you can thread wire-ties through, and attach the fabric to the brackets and curtain rod. Giving the fabric lots of slack, attach it to the panel supports at the sides so that the edge that hangs off the curtain rod has lots of slack. Use the tape to seal the top and bottom edges that touch the Lexan.

At this point, the only way for the light to get in is via the Lexan. Now put that cheap metal lens hood on the camera. Place the camera and tripod under the light box where you expect to use it, tucked under the back fabric, with two of the legs snugged up against the arena. Mark a point on the fabric an inch below the bottom of the lens hood, then make a small horizontal slit in the fabric, wide enough so you can push the lens hood through it.

That hood will now hold the fabric in position behind it, which is why it has to be conical. Position the tripod, perhaps adjusting its height, so that you can move the camera freely without being impeded by the

fabric, and test (using a TV set, not the in-camera LCD!) that there aren't any Lexan imperfections getting in the way of a clean shot.

If there are, move the camera a bit; because the lens is so close to the Lexan, the window you are shooting through is actually quite small. Seal any unused slits in the fabric with tape. Lock down the tripods to the arena and floor as best you can with cable-ties and tape, to minimize accidental tilting. Finally, take some photos of your picture-taking handiwork. Believe it or not, I never did!

Action!

The number one rule to give your camera people is "stay off the zoom." Zooming looks unprofessional, and amateurs are always zoom-happy. If you have three to four cameras, you can dedicate one to getting close-up shots and that should be the ONLY camera that zooms. When it does, it should be done as quickly as possible to frame the new close-up shot; no slow zooming! Impress this upon your camera operators: Time spent zooming rarely makes the final edit of the fight.

Before the fight, it's a different story. Have each camera zoom in on each robot and get a 5-10 second static shot of the bot. You can later edit these in at the start of the fight. But as soon as the fight is about to begin ("Red Square, are you ready?"), they should zoom out and get a wide shot of the starting positions, usually with both robots in frame.

Give your camera operators some time to practice pans and tilts with the tripods. Good tripods can be moved with a couple of fingers, and unless the action is right in their laps, most of their moves will be pans. They should try to keep the action centered, obviously, and never tilt so high as to let the lights get into the shot. All moves should be smooth.

What I do is imagine the camera is connected to the robots by a rubber band; the farther they get from the center of the image, the more the band gets stretched, and the faster I pan to catch up. If the action is such that you can't keep both robots in the frame at the same time, follow the closer one.

A great practice exercise is to put a couple of kids in the arena and have them run around pretending to be robots. And by the way, kids can be excellent camera operators, and it's a job they really like to do. Have all your cameras record all the time. Do NOT turn them off between fights. Instead, when a fight ends and any camera has less than 10 minutes record time left, change the tapes in ALL of the cameras and start recording again. This way, when you start editing, you won't have to sync up the tapes once per fight; just once per set of tapes. This will literally save you hours of labor at the cost of a few extra tapes. Needless to say, mark the tapes beforehand (i.e.: A3 for camera A, tape 3). Don't be surprised at how many tapes you need; for a big event with four cameras, arrive with 100 blank one-hour tapes! It's nice you can get them in bulk online quite cheaply.

Editing!

Once you've got all your footage, you've got to edit it together. Exactly how you do that depends on what editing software you are using (being a Mac Zealot, I use Final Cut Pro), but here are some tips:

• Import each group of tapes and lay them out in a timeline, one tape per track. Use several points in the video to line them up so they are frame accurate; I like to use sudden events like hits that generate sparks or collisions that bounce a robot up into the air. You will find that once you do this, the cameras will rarely drift more than a frame out of lock of each other over the entire hour (if they do, you'll have to do a little cutting and shifting every so often).

• Use effects to create a "picture-in-picture" (PiP) view of your shots. If you have four cameras, each one gets 1/4 of the screen. Render this, import it into the timeline, and put it on top of your camera views.

• To edit the fight, you look at the PiP view, decide which shot looks best, and copy it on top of the PiP view, changing cameras as each camera gets a better view of the action. The exact details of how this is done will depend on your software, of course. I use "macro" software such as QuicKeys to automate this process, and it's a great time-saver that I recommend.

• All fights have boring "running around not hitting each other" and "waiting for the judges to agree that a robot is dead" parts. If you want to get fancy, shrink the real-time action to a picture-in-picture in the corner and show a slo-mo replay of the good hits (remember, you'll often have several angles on them, so you can replay them several times). This gets laid in after you've edited the fights.

• Once you have the fights in a timeline edited out, you can export them individually, then reassemble them into a sequence and put them on a DVD or put them directly on the net.

• Editing is the real work; expect to

spend an hour per fight or more (video import and export time, layout, tweaking, captions, it all adds up). If you video an event and use Final Cut, contact me and I'll send you my Final Cut and QuicKeys setup.

Final Random Comments

Most of the time, you can just use the audio that the cameras capture, though when a big robot rams into the camera position, you're likely to have to edit in a bleep when the young lady operating the camera says a word she isn't supposed to know yet.

If you want to get audio from inside the box, the best way is to hang a couple of cheap omnidirectional microphones and record them on a laptop. I've tried running external mikes into the cameras themselves, but it's never worked well. Keep in mind that most laptop recording software can only record audio files of 2 gigaytes or so, which is about four hours of uncompressed stereo. So you should create a new file when you change videotapes.

If you can get a clean feed of the announcer's mike, so much the better; best is a stereo feed, one channel from the box, the other containing the announcer. SV

EVENTS

RESULTS — March 12 - April 15

Seattle Bot Battle 5 was held in Seattle, WA on 3/10/2007. Presented by Western Allied Robotics. Results are as follows:

- *Beetles (3 lb)* — 1st: "Hurty Gurty," Drum Spinner, Team Death by Monkeys; 2nd: "Gutter Monkey," Plow, Team Death by Monkeys; 3rd: "Wobble Wopper," Wedge, Team Gausswave.

- *Hobbyweights (12 lb)* — 1st: "Taurus," Wedge, Team DMZ; 2nd: "Death Dealer," Wedge, Team DMZ; 3rd: "Hexi-Macro," Pneumatic Flipper, Team WhoopAss Jr.

Central Illinois Bot Brawl 2007 was held in Peoria, IL on 3/10/2007. Presented by the Central Illinois Robotics Club. Results are as follows:

- *RC Combat (1 lb)* — 1st: "Wykydtron," Team Delta Strike Force 2001; 2nd: "Hillbilly Claw Of Death," Team Delta Strike Force 2001; 3rd: "Aluminum Sandwich," Iron Fist Robotics.

- *LEGO Sumo (1 kg)* — 1st: "Rocket," Team Hassenplug; 2nd: "Not Just Brute Force," Iron Fist Robotics; 3rd: "Pluto," Team Hassenplug.

- *500 g Sumo* — 1st: "ExSpurt," Brooksbots; 2nd: "Orthos," dbots; 3rd: "Wild Tire," Eisenmann.

- *3 kg Sumo* — 1st: "Cheeky-san," dbots; 2nd: "Brutus," dbots; 3rd: "Executioner," Brooksbots.

- *Line Following* — 1st: "Twisted," Team Hassenplug; 2nd: "Expy," Brooksbots; 3rd: "Max-TL," Team Hassenplug.

- *Line Maze* — 1st: "Max-T," Team Hassenplug; 2nd: "WhoseLost," dbots; 3rd: "Grace," Black Bots.

2007 Gilroy Bot Gauntlet was held in Gilroy, CA on 3/24/2007. Presented by California Insect Bots. Results are as follows:

- *Antweight (1 lb)* — 1st: "Pooky," Team ICE; 2nd: "Stumpy," Team DMV; 3rd: "Front Kick" on loan from Team Kick Me, driven by Pat from Team Bad Bot.

- *Beetle Weight (3 lb)* — 1st: "Unknown Avenger," Team ICE; 2nd: "Itsa?," Team Bad Bot.

Upstate NY Robot Battle V was held in Syracuse, NY on 4/14/2007. Presented by the Upstate NY Robot Combat Club. Results are as follows:

- *Antweights* — 1st: "Switchblade," Spinner, Team Sawzall; 2nd: "Firemoth," Undercutter, Team Sawzall; 3rd: "Otis," Lifter, Team Basenji; 4th: "Froogin," Wedge, Team Fishnecks.

- *Beetleweights* — 1st: "Aggravator," 4WD Pushbot, Team Dreadfully Wicked Robots; 2nd: "Uptech," wedge, Team Fishnecks; 3rd: "RipBlade," Undercutter, Team Sawzall; 4th: "Buster Blade," Vertical Spinner, Team Basenji.

- *Antweight Rumble* — SwitchBlade.

- *Beetleweight Rumble* — Buster Blade. SV

TECHNICAL KNOWLEDGE

Milling Basics

● by Karl Wolter

The vertical milling machine is arguably the most versatile of all metalworking tools. It is capable of performing operations and attaining precision that would be inconceivable by any other means. It is a machine that few amateur craftsmen own, but for people interested in hobbies such as robotics, remote control vehicles, or model making, a mill can be an excellent addition to any shop. Someone with no background in machining faces some challenges when considering buying a mill.

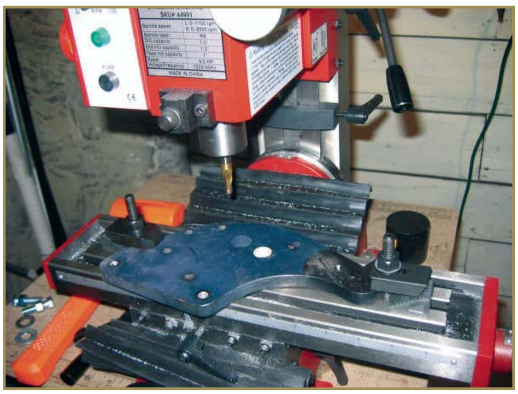
First, you have to decide which machine is best to buy. There are several good options for the home shop, the most common of which is the "mini mill." This Chinese-made machine is small and inexpensive. It would be just right for someone planning to make only small parts or having only a limited workspace. It is sold by companies like Grizzly Industrial, Harbor Freight, Homier, and Cummins. The typical model usually sells for somewhere around $500. They perform well right out of the box, but there are many products sold to modify the mini mill to get the most out of it. Many of the companies listed above also have models that are larger, but would still be a good option for a home shop. These models are similar to the mini mill in quality and features, but would allow you to work on bigger parts.

For those who prefer to buy American products, Taig and Sherline are two US companies that make a similar mill. They appear to be a higher quality machine, but cost a little bit more. Their mills are both similar and seem to be aimed at small, delicate work such as model making.

A big challenge facing someone without machining knowledge is the operation of a mill. It can seem like a daunting task, but once you learn a few of the basics, logical thinking will help you figure out a lot of other tricks. Here are a few things a starting millwright should know.

Get to Know Your Mill

Read the instruction manual! This contains important information about the controls and capabilities. It is also a good idea to research your specific milling machine. There are some great websites devoted to small mills. Like any tool, every model of mill has its share of quirks, and finding out about them from someone else's experiences can save you major hassles later on.



A part too large for a vice is held to the table of a mini mill with strap clamps.

Mounting Work

There are two main ways of securing work to the milling table. For most work, a milling vice is used. A milling vice is similar to one that you might see on a drill press, but it is more rigid, and usually has a swivel-base to facilitate cutting angles. Often the part you want to make is too small to protrude above the jaws of the vice.

In cases such as this, a set of parallels are used to raise the work up to a height that makes working on it easier. A parallel is simply a precision ground steel bar that is placed under the work. They come in sets with matched pairs of different heights to fit every size part. The other option for mounting work involves using strap clamps and step blocks. These usually come in kits including all the parts needed to hold down any size workpiece. These kits are especially useful for oddly shaped or oversized parts, because they secure the part directly to the table.

Cutting Tools

There are hundreds of different cutting tools available for a milling machine. Many of them are designed for specialized jobs such as boring holes or cutting Woodruff key-slots. For general-purpose cutting, an end mill is used almost exclusively. End mills are a good general-purpose cutter because they cut both on their sides and on their face (a drill bit, for example, cuts only on its face). They are usually held in either a collet or an end mill holder. Both of these holding devices accomplish about the same thing. They both have their pros and cons; as to which is better depends on who you talk to. The collet/end mill holder is held in the spindle by the taper of the spindle (similar to that of a drill press) and held tight with a drawbar. A drawbar is essentially a bolt that pulls the tool into the taper of the spindle to hold it in place.

Running the tool at the proper speed is also important. A good formula to use for speed is (cutting speed of the material in feet per minute) x 4 / (diameter of cutter) = RPM. This formula also applies to drilling holes and lathe work, too. If the cutter "chatters," or the chips turn blue when cutting, you are most likely either running the end mill too fast, or the end mill is dull. A general guideline for cutting speeds is: steel 100 feet per minute; brass 200 fpm; aluminum 300 fpm. If you are working with a specific alloy of metal, the exact speed can be looked up in a machinist's reference book.


An edge-finder held in a collet is used to set up a work piece.

Edge Finding

The key to making precise parts is to know where your cutter is in relation to the edge of the part. There are two ways of doing this. The easiest way to find an edge is to turn the machine on with a cutter mounted in the spindle and simply "touch off" by slowly bringing the cutter in contact with the work. This is not the most precise method, and it will also leave a small mark on the edge of the part. The other method is to use an edge-finder.

An edge-finder is a steel bar of a known diameter with a magnetic piece of the same diameter on the end. The bar is mounted in the spindle and the magnet is slid to a position that is visibly offset from the bar. Then with the spindle turned on, you carefully bring the magnet into the side of the work until it runs concentrically with the bar as it rotates. Once you find an edge, you should set the calibrated collars on the feed hand-wheel to zero. It is also important to remember that the cutter/edge-finder is still not truly on the edge of the part; you must first move the table over one half the diameter of the cutter/edge-finder to center it over the work piece.

Milling Accessories

Most mills do not come with many accessories, so here are a few things you will need to get started:

- End mills
- Milling vice
- Parallels
- Strap clamp kit
- Micrometer and caliper


The mini mill is the most popular type of mill for hobbyists and home shop machinists.

- A machinist's square
- Edge finders
- Collets or end mill holders
- Cutting tools for specific operations you wish to perform

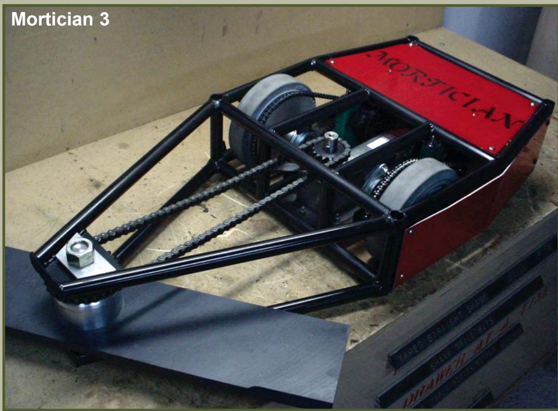
Keep in mind that this is only a very brief description of the basic principles of running a milling machine. There are many other operations that can be performed using a milling machine such as cutting radiuses, boring holes, and cutting gears. For more complete information on operating a mill, some excellent resources are: **www.thelittlemachineshop.com** and **www.minilathe.com**. Both of these sites go into much more detail about operating milling machines and they are both specifically geared toward small mills. SV

PRODUCT REVIEW — Team Whyachi Gearboxes

● by Ray Billings

Team Whyachi (**www.teamwhyachi.com**) manufactures many custom components just for the sport of robotic combat. Among them is the line of gearboxes designed to be used with the popular S28-400 3" Magmotor. The Magmotor is a powerful unit, but using that power in combat requires proper gearing, and a rock solid platform. The Whyachi gearboxes provide that in a solid, battle proven package.

For drive platforms, the TWM3 works great. With just over 7:1 gear reduction, this delivers 686 output rpm, with 1,664 in-lb. of torque (when the S28 is run at 24V). This is more than enough for a pair of these to drive a really pushy middleweight, heavyweight, or even



Mortician 3

TWM3R (motor not included).

a weaponed superheavyweight that doesn't need to rely on sheer pushing power. Four of these in a heavyweight would be a pushing monster! The beefy 1" titanium output shafts are designed for the direct mounting of wheels, and Team Whyachi also makes custom wheels specifically for these gearboxes.

For weapon platforms, the TWM3R is a great choice. This right angle gearbox allows you to supply power to a horizontal weapon and keep a low profile to the robot. A lower center of gravity is almost always a major advantage in combat. The 2:1 gear ratio provides a no-load output rpm of 2,450 at 24V and delivers 466 in-lb. of torque. The Fortal® aluminum construction is both lightweight and strong, and this unit also sports a 1" titanium output shaft. There is even an option for changing the output shaft so you can directly mount a spinning weapon to the gearbox.

Cost for these gearboxes is $450, which does not include the 3" Magmotor. The S28-400 Magmotor is available separately from **www.robotbooks.com** and has proven to be the motor of choice for many top teams. The Whyachi gearboxes provide an off-the-shelf solution to putting the power of this motor to its best use. I have recently upgraded to the TWM3R combination to spin the weapon bar on my middleweight robot, The Mortician. So far, I have been very pleased with its performance, and I expect to do very well with it at this month's Robogames (**www.robogames.net**) event in San Francisco, CA June 15-17. Stop by and see how effective this combination can be! **SV**

EVENTS

UPCOMING — June and July

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.

WBX-IV Bushwacked — This event will take place in Saskatoon, Saskatchwan, Canada on 7/21/2007 through 7/22/2007. It's presented by War-Bots Xtreme. WBX will adopt a NEW location for this event. A rural setting, 25 minutes south west of Saskatoon, will see combat robots competing for prizes and cash. Go to **www.warbotsxtreme.com** for more information. **SV**

Robotics Showcase

DIAL-A-VOLTAGE

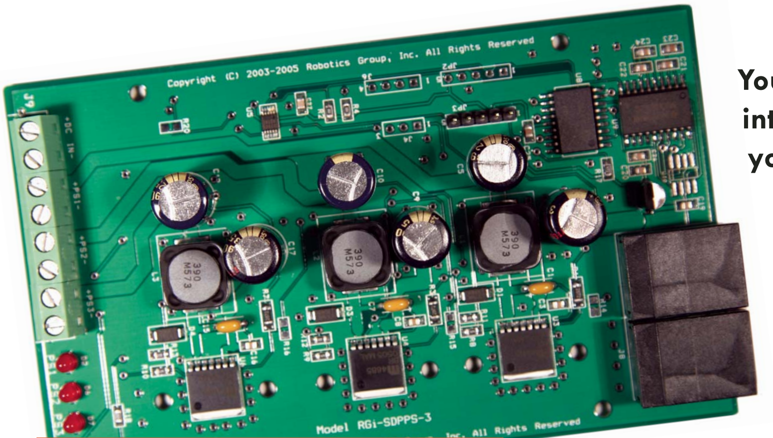


PHOTO 1. This is an overhead shot of the RGi-SDPPS-3. Note that the RGi-SDPPS-3 is made up of a trio of identical buck power supplies based on the Micrel MIC4685. The large blocks at the bottom right of this shot are 10-pin female RJ-45 jacks that are used to connect and power other RGi or user-designed peripherals.

You're reading this magazine because you're interested in robotics. Right? At least that's what you think you're reading this magazine for. In reality — my robotically inclined friend — you are a control freak. If you really get down to it, the real reason you're reading this magazine is to get the low-down on how to control things robotic. Just keep on reading. I will feed your control need and try not to disappoint you.

BY FRED EADY

I have a really tricky little robotic peripheral device to show you. I'm always preaching motor control. However, you can't have motor control without having control of the power that is being supplied to the motor. Motors aren't the only pieces of a robotic unit that must have a rock-solid power source. Today, you are building robotic equipment that can "see" (using an embedded camera) and "hear" (using ultrasonic technology and high-gain sonic amplification) among other things and those sensory subsystems require a clean, stable power source for reliable operation.

A challenge can arise when you deploy a robotic peripheral that requires a non-standard voltage. A similar challenge is also presented when that robotic peripheral needs to have its power "rationed." Rationed, in this case, means that the efficient operation of the robotic peripheral (and possibly the robot itself) requires the robotic peripheral's power source to be applied and removed on command. You can't run everything with a nine-volt transistor battery. So, in addition to having the ability to receive and execute power-saving commands, it would also be to our advantage to have a programmable power source with a punch.

A Bucking Bronco Power Supply

The aforementioned tricky little robotic peripheral device I want to show you is a three-station programmable power supply that is manufactured by RGi (Robotics Group Incorporated). The RGi-SDPPS-3 I used to assemble the words in this article is shown in Photo 1.

The RGi-SDPPS-3 consists of three identical buck switching power supplies, which are based on the Micrel MIC4685 stepdown switching regulator. The MIC4685 is a high-efficiency stepdown (buck) switching regulator that operates at a fixed switching frequency of 200 kHz. The Micrel MIC4685 is capable of attaining a power conversion efficiency of 85% while servicing loads up to 3A. A logical view of the Micrel MIC4685 is shown in Figure 1.

The MIC4685 uses a boost capacitor to increase the voltage that drives the MIC4685's internal power switch. The boost capacitor is connected between the SW (Switch Output) and BS (Bootstrap Voltage Node) pins in Figure 1. If you've ever worked with MOSFETs in a power switching application, you know that the harder you drive the MOSFET gate into turn-on, the lower the drain-to-source resistance with the idea being to reduce the device's internal resistance to the specified device minimum.

The reduced drain-to-source resistance results in the MOSFET running cooler and more efficiently. Utilizing the boost capacitor allows the MIC4685 to work in a similar manner as the hard-driven MOSFET I just described. The result is the elimination of the MIC4685's need of an external heatsink. The ground plane of the printed circuit board (PCB) supporting the MIC4685 is all the heatsink that the MIC4685 requires.

The Micrel MIC4685 switching buck regulator IC is designed to replace traditional TO-220 cased linear voltage regulators such as the LM7805 fixed voltage regulator and LM317 adjustable voltage regulator. To that end, the MIC4685 needs to incorporate the safety features that are designed into the linear voltage regulators. The MIC4685 provides over-current protection and thermal shutdown just like its linear counterparts.

In addition, in short-circuit conditions the MIC4685 invokes frequency-foldback, which drastically reduces the MIC4685's switching frequency and duty cycle. The reduction in the MIC4685's switching frequency and duty cycle results in a huge reduction in the energy that the MIC4685 has to process and ultimately protects the MIC4685 from damage.

The LM7805 datasheet states that the device has a maximum input voltage of +35V. The LM317 operates a bit differently and can tolerate an input-to-output voltage differential of +40V. The Micrel MIC4685's maximum input voltage is rated at +30V with a transient rating of +34V. Thus, the MIC4685 can easily fit into an LM7805 power supply application. If you are using input voltages that exceed +34V in an adjustable voltage regulator application, the LM317 or another adjustable voltage regulator may be a better choice than the MIC4685.

You can obtain the Micrel MIC4685 in both fixed and adjustable variants. If you've ever worked with the LM317, you know that you can't "turn it off" without applying a negative bias to the LM317's adjust pin. Without the negative bias on the LM317 adjust pin, the minimum output voltage of the LM317 is its reference voltage, which is approximately +1.25V. The MIC4685's minimum adjustable voltage is +1.235V. However, unlike the LM317, you can apply a logical low level to the MIC4685's EN (Enable) pin and completely shut down the MIC4685's output.

The MIC4685's EN pin directly controls the MIC4685's internal regulator output. Thus, when the MIC4685's internal regulator is shut down, the MIC4685's output is disabled. Now that you know the function that the MIC4685's EN pin (pin 5) performs, let's walk around the MIC4685 and discuss the functionality of the remaining MIC4685 pins.

The MIC4685's NPN transistor-based output subsystem operates just like any other NPN transistor circuit wired in this configuration. If a voltage is applied to the NPN transistor's collector and the base of the transistor is properly biased, the output voltage at the transistor's emitter will be slightly less than the voltage applied to the transistor's collector. Said another way, the voltage at the transistor's emitter will be equal to the voltage applied to the collector minus the transistor's VSAT voltage value. In the case of the MIC4685, the NPN output stage exhibits a VSAT of 1.8V without the assistance of the boost capacitor.

The MIC4685's BS pin (pin 1) when used with a boost capacitor provides a bias voltage that is higher than the input voltage applied to the MIC4685's NPN transistor subsystem. Changes in the voltage at the MIC4685's SW pin versus time are seen as an AC voltage by the boost capacitor. The boost capacitor passes this AC voltage to the BS pin where it is rectified and added to the input voltage applied to the NPN transistor subsystem.

This rectification and voltage addition provides additional drive to the NPN transistor subsystem. The additional drive provided by the boost capacitor and BS circuitry reduces the NPN transistor subsystem's VSAT to 0.5V, which raises the MIC4685's operating efficiency from 75% to 88%. Highest efficiency is achieved when the MIC4685's unregulated input voltage is around +12V.

The collector of the NPN transistor pass subsystem is connected to pin 2 (IN), the MIC4685's unregulated voltage input. The minimum voltage that can be applied to this pin is +4V. If you take another look at Figure 1, you'll see that this pin also feeds the MIC4685's internal regulator subsystem. The MIC4685's IN pin's alter ego is the MIC4685's ground pin (pin 4).

I noted earlier that the Micrel MIC4685 could be had as an adjustable buck switching regulator or as a fixed buck switching regulator. The MIC4685's FB pin (pin 3) is internally connected to the inverting side of an error amplifier as shown in Figure 1. The 1.235V bandgap reference is tied to the error amplifier's noninverting input.

When an adjustable version of the MIC4685 is used, an external precision voltage divider must be employed with the output (center) of the precision voltage divider feeding the FB pin. The fixed voltage versions of the MIC4685 incorporate the precision voltage divider into the IC with the FB pin being fed by the regulated output voltage on the C_{OUT} side of the inductor.

The MIC4685's error amplifier compares the signal at the FB pin with

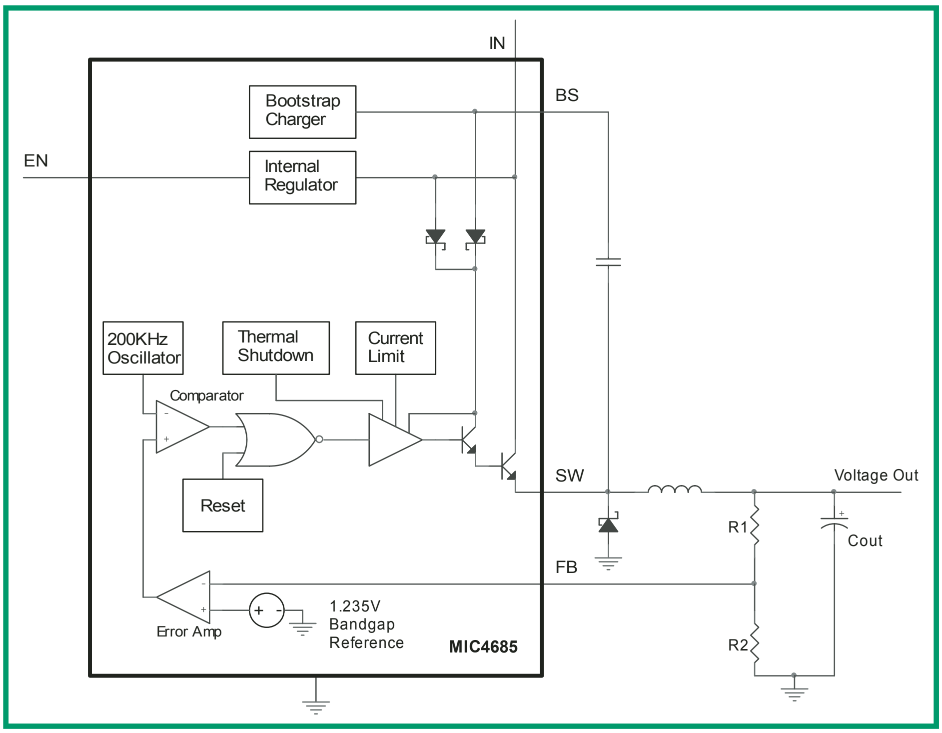
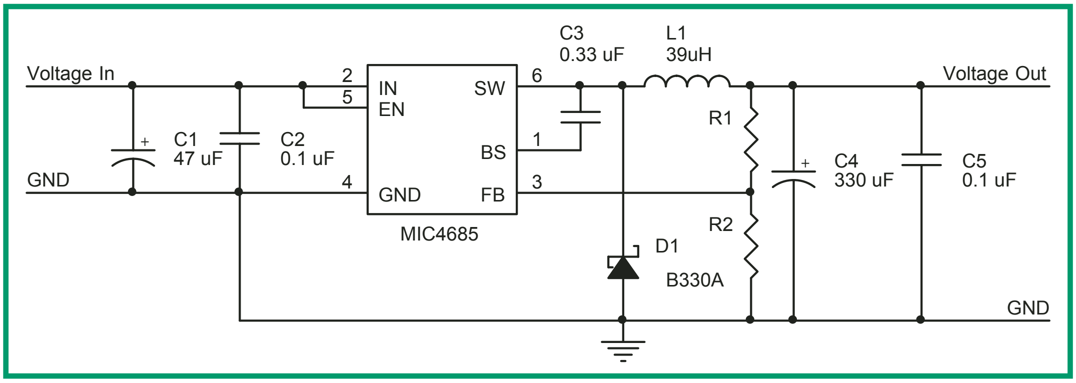


FIGURE 1. All you have to worry about is setting up the precision voltage divider to get the regulated output voltage you require. If you need a standard voltage (+5.0V, +3.3V, etc.), all you need to do is consult the voltage/resistor value chart in the MIC4685 datasheet.



SCHEMATIC 1. The MIC4685 datasheet sets the value of R1 at 3.01K to obtain the standard set of voltages (+5.0, +3.3, +2.5, +1.8). The value of R2 never exceeds 20K, which makes the DS3905 an idea digital potentiometer choice for the RGi-SDPPS-3.

the 1.235V bandgap voltage reference tied to its noninverting input. The resultant output voltage generated by the error amplifier is fed to the inverting input of the MIC4685's comparator, which has its noninverting input fed by a 200 kHz sawtooth signal generated by the MIC4685's internal 200 kHz oscillator.

The output of the error amplifier follows the trend of the voltage applied to the FB pin. Thus, the higher the voltage at FB, the higher the error amplifier output voltage. The voltage level of the 200 kHz sawtooth signal must exceed the error amplifier output voltage level to trip the comparator. In other words, the higher the error amplifier output voltage, the lesser of the 200 kHz sawtooth waveform is applied to the driver, resulting in a reduced duty cycle. As the error amplifier voltage decreases, more of the 200 kHz sawtooth is presented from the comparator output to the driver and the duty cycle presented to the NPN pass transistor subsystem increases accordingly.

The MIC4685's SW pin (pin 6) is connected directly to the emitter of the MIC4685's NPN pass transistor subsystem. Voltage at this pin is equal to the unregulated input voltage applied to the MIC4685's IN pin minus the NPN pass transistor's VSAT voltage. When the MIC4685's internal switch is ON, current flows from the IN pin through the NPN pass transistor subsystem to the external storage inductor and on to the output capacitor to the load. The inductor stores energy over time as the current flowing through it is increasing over time.

The term "duty cycle" implies that there is an OFF time associated with the ON time. When the duty cycle voltages take the MIC4685's NPN switch elements to an OFF state, the magnetic field contained within the external storage inductor collapses and forces current to flow through the diode connected to the SW pin and charge the C_{OUT} capacitor.

In the OFF state, the diode connected to the SW pin provides a high-current return path for the inductor's stored energy that allows the inductor's energy to be channeled to the output capacitor and the load. The output capacitor's job is to reduce regulated output voltage ripple. The presence of the output capacitor also provides a degree of stabilization to the buck switching system. Now that you know how the MIC4685 goes

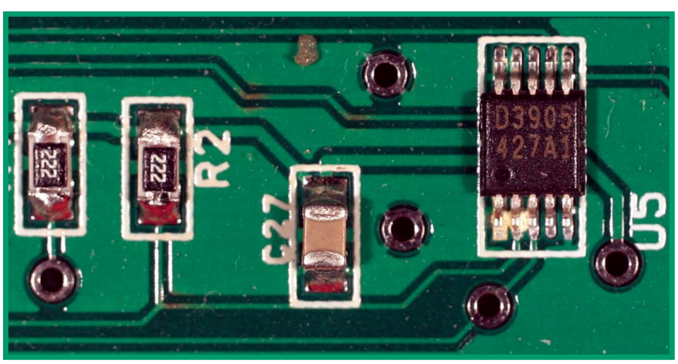


PHOTO 2. This shot shows the I²C (two-wire interface) pullup resistors and what must be the DS3905's 0.1 µF power supply bypass capacitor.

about providing a regulated output voltage, here's the math behind obtaining the output voltage you desire:

$$V_{OUT} = V_{REF}\left(\frac{R1}{R2} + 1\right)$$

$$R1 = R2\left(\frac{V_{OUT}}{V_{REF}} - 1\right)$$

$$V_{REF} = 1.235\text{V}$$

A typical MIC4685 circuit configuration is depicted in Schematic 1. Run the MIC4685 regulated output equations with R1 fixed at 3.01K to get MIC4685 datasheet values for R2 versus the standard voltages (+5.0, +3.3, +2.5, +1.8).

Manipulating the Voltages

Judging from the simple math behind the generation of the MIC4685's regulated voltage output, the resistors, R1 and R2, and thus the precision voltage divider are the determining factors of the MIC4685's regulated output voltage. Over the years, I have discovered that simple observation can yield a mountain of useful information. Since the RGi-SDPPS-3 is electronically controlled and I don't see any motors driving mechanical potentiometers, logic would dictate the use of digital potentiometers in the RGi-SDPPS-3's voltage control circuitry. A closer look at the RGi-SDPPS-3 componentry that makes up the RGi-SDPPS-3 reveals a tiny 10-pin DS3905 digital potentiometer. I got as close to the DS3905 as I could for you in Photo 2.

As you can see in Figure 2, the DS3905 houses three nonvolatile,

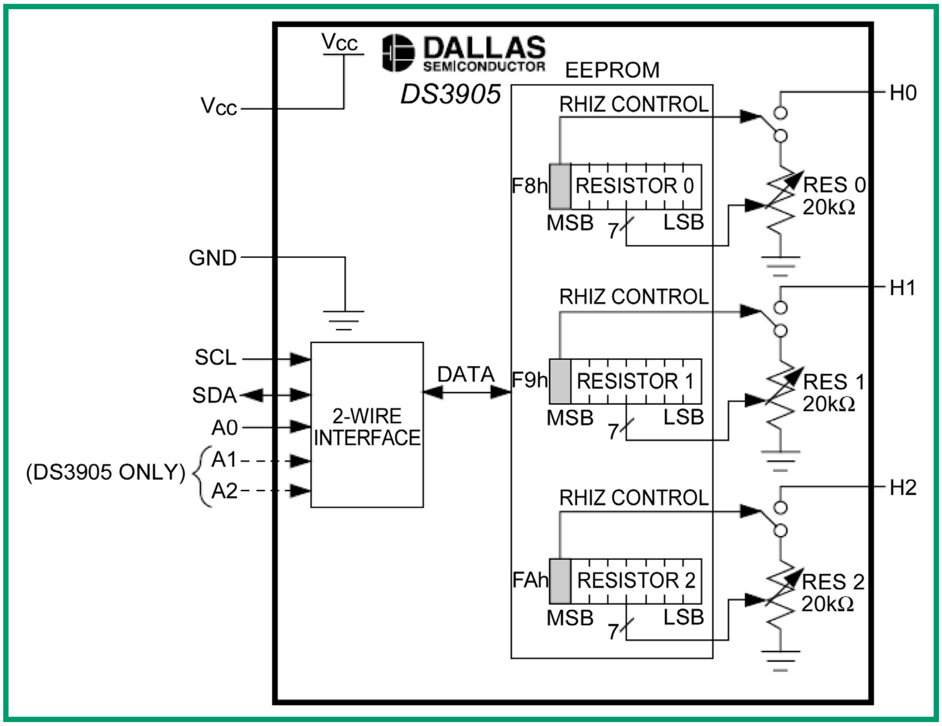


FIGURE 2. The DS3905 only houses a trio of digital pots. That only accounts for half of the three voltage dividers we need to control the RGi-SDPPS-3's three MIC4685s. Note that the DS3905 potentiometer outputs can also be configured as digitally-controlled logic outputs.

digitally controlled potentiometers, which can also be configured as logic outputs. Each of the DS3905's three 20K 128-position digital potentiometers falls under the control of a set of seven-bit (0x00 through 0x7F) control registers addressed as 0xF8, 0xF9, and 0xFA, respectively.

Hanging a pullup resistor on the potentiometer pin and writing a 1 or 0 to the most significant bit of the associated potentiometer's control register will put the potentiometer position into logic output mode. I don't have a schematic of the RGi-SDPPS-3. However, I don't believe the logic output mode of the DS3905 is used in the RGi-SDPPS-3 firmware. The lack of obvious pullup resistors at the DS3905's potentiometer pins supports my no-logic-mode theory. So, I won't go any further into the way the DS3905 logic outputs work.

One thing that I do know is that according to Photo 2, there is only one DS3905, which equates to only three digital potentiometers. That means that the other side of the trio of precision voltage dividers must be a fixed precision resistor. Closer observation of Figure 2 tells us that all of the DS3905's potentiometers are referenced to ground. That adds to the probability that the DS3905's potentiometers are acting in the role of R2 in Schematic 1.

Twisting the DS3905 Potentiometers

The DS3905's three digital potentiometers are controlled via a two-wire interface, which is just another name for I^2C. If you're I^2C challenged, here's a one-minute all-you-need-to-know I^2C tutorial. The SDA pin carries the data and is pulled high with an external resistor. You will normally find the SCL (clock) pin pulled high, as well (see the pair of resistors in Photo 2). Data on the SDA pin can only change when the SCL pin is low.

There are three states — or conditions — that proper I^2C operation depends on. A high-to-low transition of the SDA line while the SCL line is high is defined as a START condition. Recall that data on the SDA line can only be changed when the SCL pin is low. The START condition must always precede a command. Conversely, a low-to-high transition of the SDA line while the SCL line is high constitutes a STOP condition. The third common state of an I^2C conversation is called the ACKNOWLEDGE condition. I^2C communication is based on a clocked serial data transfer technique. The DS3905 working in slave mode will pull the SDA line low during the ninth clock pulse to acknowledge the reception of the previously received byte. All of the clocking of the SCL line is done by the master device. Now that you know all you need to know about I^2C and two-wire interfaces, let's look at how the DS3905's potentiometers get "adjusted."

As you've just learned, I^2C is a master/slave communications method. In the case of the RGi-SDPPS-3, the DS3905 is the slave and an on-board Microchip PIC16F648A is most likely the master. The PIC16F648A does not house an on-chip hardware two-wire interface. That means the I^2C transmit and receive routines are all generated in firmware. The RGi-SDPPS-3's serial interface, consisting of a MAX3222 and the PIC16F648A as shown in Photo 3, receives commands from an external source such as a personal computer or another serially equipped microcon-

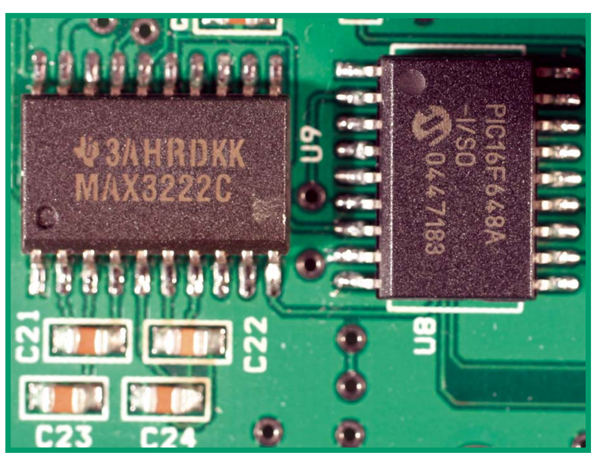


PHOTO 3. There's nothing here that is new to you if you've ever performed RS-232 duty with a Microchip PIC microcontroller. This shot also captured the quartet of boost capacitors used by the MAX3222 to obtain regulation RS-232 voltage levels.

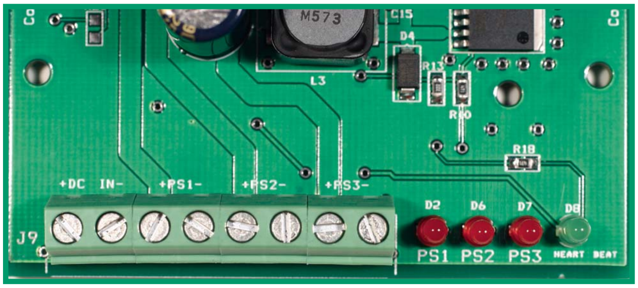


PHOTO 4. As you can see, this is a very clean interface. The Heart Beat LED flashes to indicate that data is being received by the RGi-SDPPS-3's RS-232 interface.

troller-controlled device. The incoming commands to the RGi-SDPPS-3 serial port are parsed and passed to the RGi-SDPPS-3 application's I^{2}C handler routines.

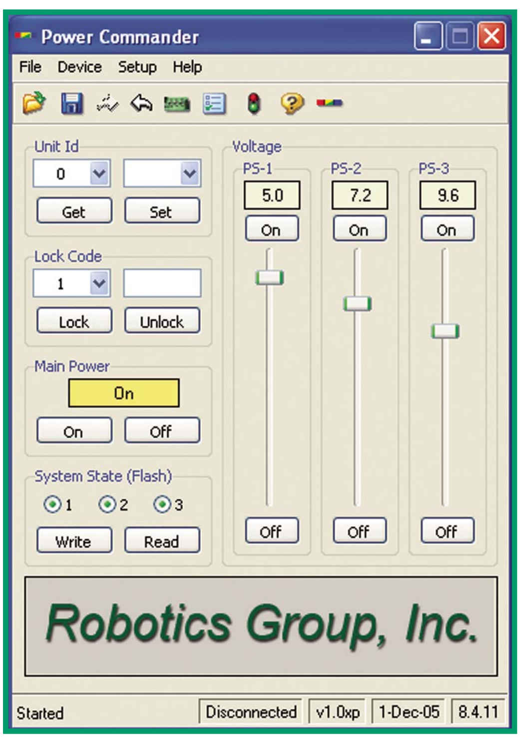
For the sake of our discussion, we will assume that the DS3905 is externally addressed as binary 000 by tying all of the DS3905 address lines (A0, A1, and A2) to ground. As you learned earlier, all I^{2}C commands begin with a START condition. A four-bit family code, a three-bit address, and a read/write bit make up the command byte that immediately follows the I^{2}C START condition. The family code (binary 1010) contained within the four most significant bits of the command byte are defined by the DS3905 manufacturer. Assuming a write to one of the DS3905's control registers will be performed, here is what the first byte that is sent to the DS3905 looks like logically:

START 10100000 ACK

After receiving and interpreting the START condition, the DS3905 will serially receive the byte and issue an ACKNOWLEDGE on the ninth clock pulse. If we further assume that we are writing to potentiometer 0, the next byte that the master will clock out will be the control register address of potentiometer 0, which is 0xF8. Here's the logical I^{2}C bit sequence that represents the potentiometer control register address:

11111000 ACK

Remember that the ACK is generated by the DS3905 in response to the reception of the 0xF8 byte. To set potentiometer 0 to its maximum value, the master device must next clock out 0x7F. If we wanted to set potentiometer 0 at its minimum value, the master would be instructed to clock out 0x00. With that, let's max out potentiometer 0:



01111111 ACK STOP

The DS3905 acknowledges the received byte (0x7F) and the master device issues a STOP to indicate to the DS3905 that this is the end of this particular I^{2}C communications session. Expanding the logic behind our example tells us that we can physically address up to eight DS3905 devices (binary 000 through binary 111 on A0, A1, and A2) on the I^{2}C bus. We already know that we can individually access the DS3905's three seven-bit control registers and twist the DS3905's knobs to dial in a resistance between 0Ω (0x00) and 20K (0x7F) using less than a single byte of information.

SCREENSHOT 1. It's pretty obvious what everything in this shot does. So, I'll shut up.

When the MIC4685, DS3905, and PIC16F648A subsystems all work in harmony, the fruit — in the guise of a highly regulated DC voltage — is delivered via a line of easily accessible screw terminals as seen in Photo 4. The ON/OFF status of each of the MIC4685-controlled power supplies is provided by the status LEDs directly to the right of the line of screw terminals.

Communicating with the RGi-SDPPS-3

We've established the fact that the RGi-SDPPS-3 is a programmable power supply peripheral designed to provide clean and stable power for your robotic endeavors. If you take a closer look at Photo 1, you'll notice large black connectors at the bottom right hand corner of the photo. This pair of 10-pin female connectors is designed to carry control signals and power to other RGi modules. You can also power your own stuff with the RGi-SDPPS-3 by way of these 10-pin connectors. We have a well-designed, programmable power source. Let's see what it takes to control it.

The RGi-SDPPS-3 can be made to pay attention to the serial port of a laptop or desktop personal computer using the Power Commander application. The main window of the Power Commander application is shown in Screenshot 1. You can readily see how everything works by simply observing the controls contained within the Power Commander window.

You can also use a standard microcontroller to send commands and receive status from the RGi-SDPPS-3. An 8051 development board from Silicon Laboratories has been finding its way around the EDTP lab bench and I decided to grab it and use it to demonstrate how easy it is to use the RGi-SDPPS-3 RS-232 command interface. The Silicon Laboratories C8051F120-based development board I will use is shown in Photo 5. What you don't see in the photo is the Silicon Laboratories USB debug adapter that forms an interface between the C8051F120 development board hardware and the Keil 8051 C compiler application that runs on my laptop.

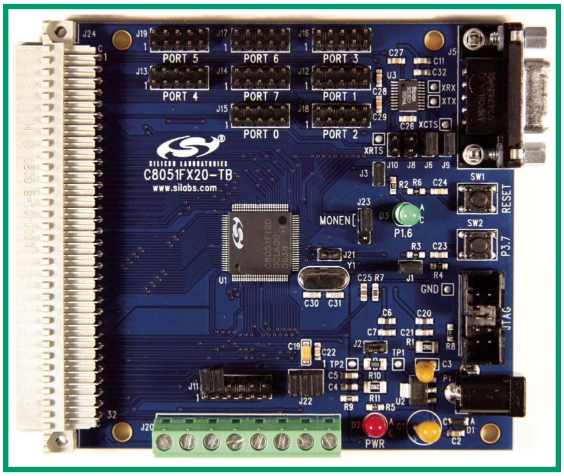


PHOTO 5. There are a bunch of goodies on this development board to play with. However, I'm only interested in coaxing the C8051F120 to deliver some bytes to the RGi-SDPPS-3 via its RS-232 port.

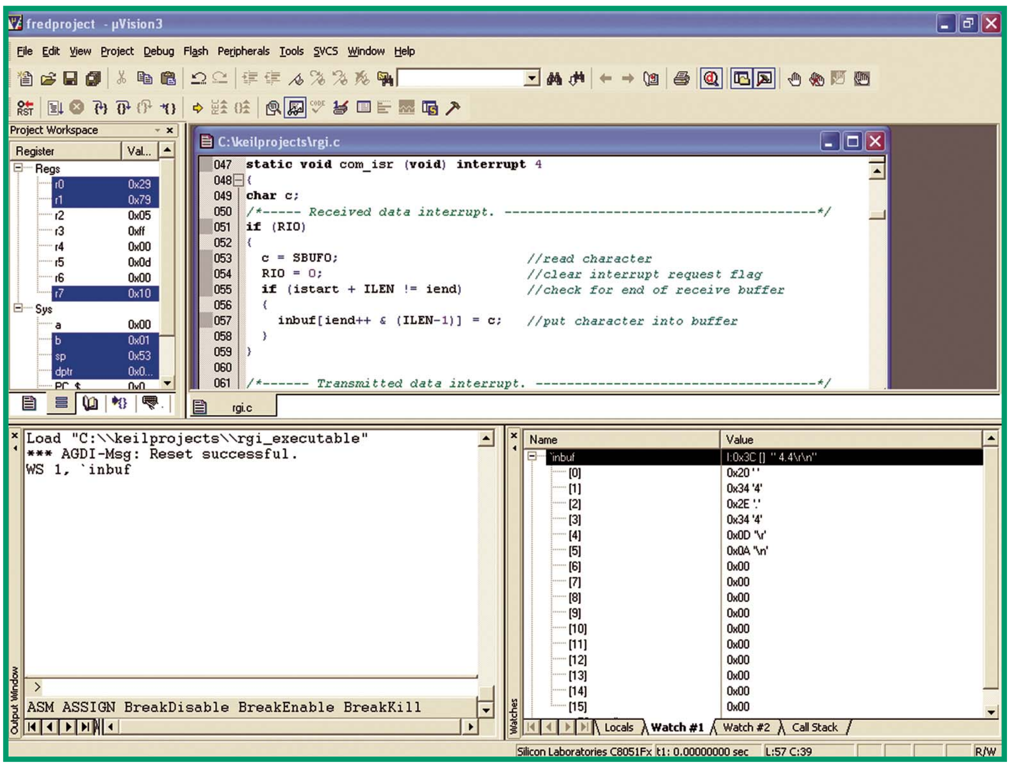
The Silicon Laboratories USB debug adapter allows me to download the C8051F120 firmware I generate with the Keil 8051 C compiler into the C8051F120 Flash and execute it. I can also use the Silicon Laboratories adapter to step through the firmware instruction by instruction and examine the firmware's variables between steps. If you want to get deeper into the C8051F120 and the Keil 8051 C compiler, Peter Best has done some indepth work with the C8051F120 and the Keil 8051 C compiler in his *Nuts & Volts* Design Cycle column. For now, we're only interested in getting some bytes into and out of the C8051F120's UART.

If you're bracing yourself for a convoluted binary communications algorithm, chill. The RGi-SDPPS-3 speaks English. For instance, to set the voltage on power supply 1 to +4.4 VDC, we simply send the ASCII string "set voltage1 4.4" followed by a carriage return character (0x0D). To activate power supply 1, all we have to do is send the ASCII string "set power 1 1" followed by a carriage return character. To turn off power supply 1 we send "set power 1 0" and a carriage return. Get the idea?

I wrote some simple code using the Keil 8051 C compiler that used the C printf statement to send ASCII strings to the RGi-SDPPS-3 serial port. The C printf statements used to communicate with the RGi-SDPPS-3 all take the form of the C printf statement that follows:

```
printf("\r\rset voltage1 4.4\r\r");
```

The \r characters represent the carriage return character, which translates to a hex character of 0x0D. You'll see that I used the \r character rather freely in my RGi-SDPPS-3 command code, which I've posted it in it's entirety for you on the *SERVO* website (**www.servomagazine.com**). I issued a "get voltage1\r" ASCII string to obtain the results you see in the Watch #1 window at the bottom right of Screen Shot 2.



SCREENSHOT 2. This is how the Keil 8051 C compiler and the integral uVision3 IDE appeared to me as I intercepted the message that was sent from the RGi-SDPPS-3 in response to my "get voltage1" command. The "captured" message is actually the contents of the receive buffer I allocated.

I think we've talked enough about the RGi-SDPPS-3 and its design points to spark your own ideas about how to use it. The RGi-SDPPS-3 is easy to talk to and easy to wire into your robotic mesh. All that's left for you to do is get one and put it to work. **SV**

SOURCES

RGi-SDPPS-3
Robotics Group Incorporated
www.roboticsgroup.com

Micrel MIC4685
MICREL
www.micrel.com

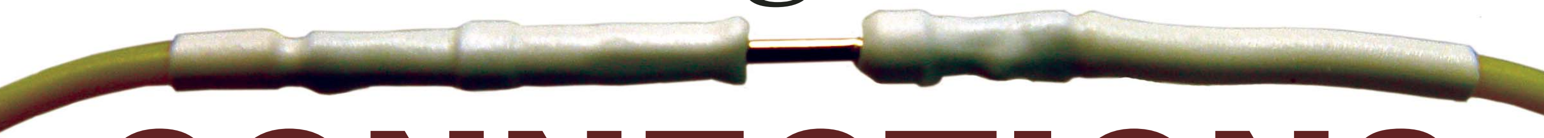
Keil 8051 C Compiler
Keil
www.keil.com

USB Debug Adapter
C8051F120 Development Board
C8051F120 Microcontroller
Silicon Laboratories
www.silabs.com

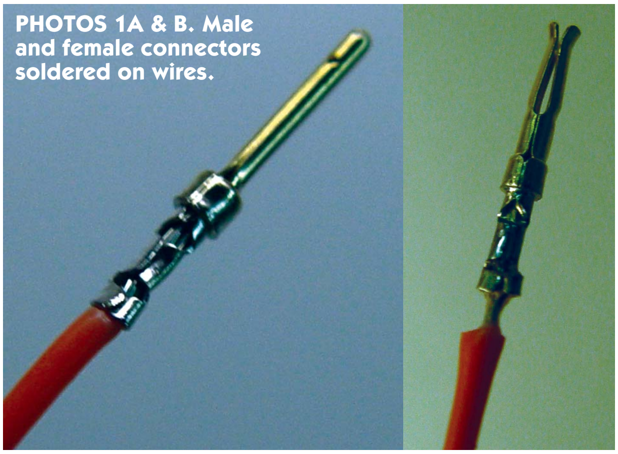
AUTHOR CONTACT INFO

You may reach Fred Eady via email at fred@edtp.com

Making Great



CONNECTIONS



PHOTOS 1A & B. Male and female connectors soldered on wires.

by Jim Miller

I've discovered an incredibly cool trick for hook-up wires, test leads, and sensors. If you're an experimenter like me — and you most likely are if you're reading this article — you've run into the problem of never having the right adapter or configuration of test lead. Better yet, you've got this problem where several alligator clips are precariously attached to a piece of wire on a breadboard and the slightest movement of the board will fry something. The connectors I'm presenting here have saved me from myself many times.

With a little heat shrink

tubing and a package of male and female crimp connectors for a D-sub type connector body (the usual RS-232 is a D-sub), you can vastly improve the utility of your test gear. I have them on just about everything and they have really increased my productivity in the shop. If I need a servo connection with some alligator clips, no problem! I just plug the appropriate connections together and done.

The pins are just the right size for a very solid breadboard connection. The heat shrink helps to protect the electrical integrity of the connection. They also hold well together when used in line. The best part is that they are cheap and really easy to make. Here's how.

STEP 1: Carefully crimp then solder the wire onto the connector. Don't just crimp! If you want them to last, use solder.

STEP 2: Wrap the connector and wire with a piece of 1/16" heat shrink tubing 3/4" long and shrink it down. You'll want the tubing to be flush with the end of a female connector and just onto the base of the pin on the male. That way, when they are connected you won't have any exposed connector.

Voilá! You're done!

Here is a test lead with the connector's connected. Notice that this single test lead is now a set of components in any of several different configurations.

I've also used this technique in my robots. Photo 5 shows the incoming leads from the sensor array on my rover 'Squeeze.'

There is a catch here. This size pin does not go over the standard pins you'll see on a jumper-type pin for a circuit board. For that, you'll need a little different type of female connector. These can be found with certain brands of connectors for mating ribbon connectors. Take a look at Photo 6.

A couple of final suggestions ... Over the years, I've amassed quite a collection of these little guys. I didn't just sit down and make them all up one day. I'll whip them up when I'm waiting on something, trying to make good use of a little time slice in the shop. Anytime I'm making an experimental use connection between two devices, I'll use this technique to wire it up. That ultimately adds to my overall collection and I end up using them again.

I hope you'll find these connectors as handy as I have. Drop me a note at jim@cannibalrobotics.com and let me know what new uses you come up with. **SV**

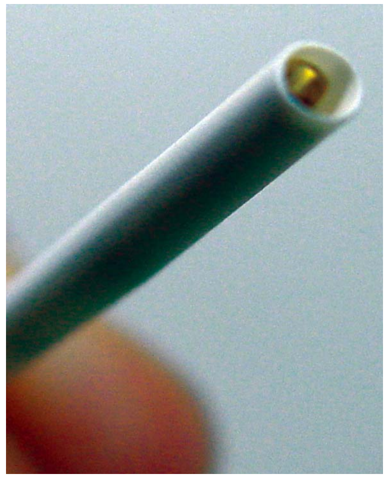


PHOTO 2. Heat shrink placement.

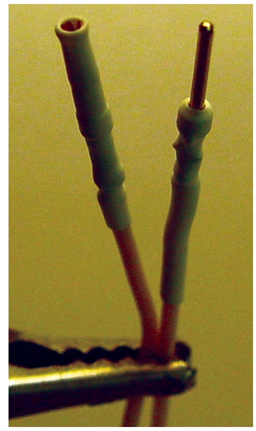


PHOTO 3. A completed pair.

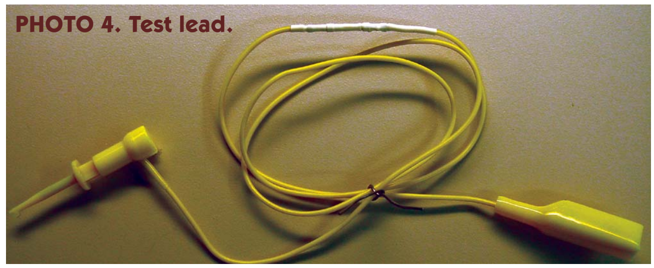


PHOTO 4. Test lead.

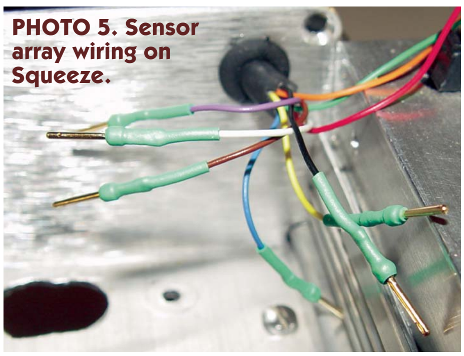


PHOTO 5. Sensor array wiring on Squeeze.

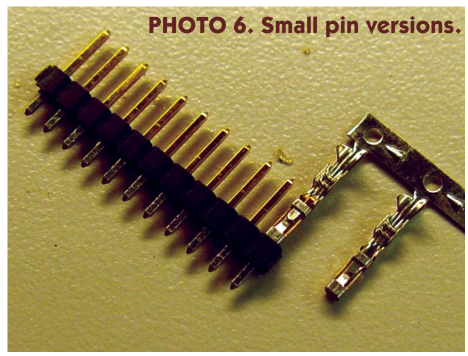


PHOTO 6. Small pin versions.

Build the Ultimate Remote Control

PART 2

by Michael Simpson

Last month, we built the bulk of our Ultimate Remote Control. This month, I'm going to show you how to add a couple of Zigbee units to the system. This will allow you to build a true two-way remote that will allow you to send commands to your robot, as well as collect telemetry and display it on the controller's LCD.

The Vex Transmitter

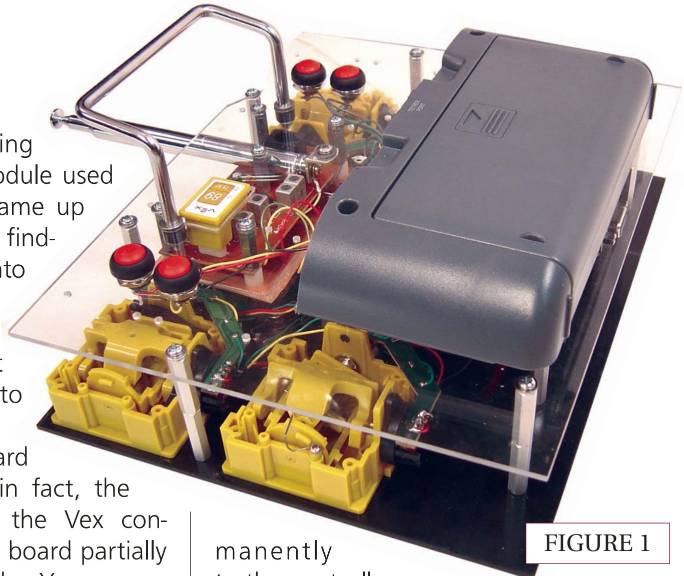


FIGURE 1

I was experimenting with the transmitter module used in the Vex radio and came up with some interesting findings. Before I jump into the Zigbee interface, I want to show you how to actually connect the Vex transmitter to your controller.

The small circuit board shown in Figure 2 is, in fact, the transmitter portion of the Vex controller. This was a circuit board partially covering the two joysticks. You can go ahead and remove it completely from the radio if you wish to duplicate the experiments I am about to show you. For my initial tests, I mounted mine on a small piece of plastic as shown in Figure 3.

The board has three connections we are concerned with. The ANT pad connects to the Vex antenna via a small lug. The GND pad connects to VSS on our controller and the VCC pad is connected to VIN. The MOD pad which is the input to the transmitter is connected to the DiosPro port 28 as shown in Schematic 1.

You can wire the transmitter temporarily to the controller, so make sure the wires you connect are at least 12" long. You can trim them down later if you decide to mount the transmitter permanently to the controller.

We modulate the transmitter by applying a ground to the mod lead on the transmitter. The timing is a bit restrictive, but by experimenting, I have been able to come up with a reliable interface.

Take a look at the program VEXTXanalog.txt. This program has a simple loop where all eight of the analog channels on your remote are read along with the keypad and six buttons. The program then pulls the modulator port low for the number of microseconds corresponding to the A/D reading for the particular channel. The same is done for the keypad and each of the six buttons.

In order to allow you to trim each

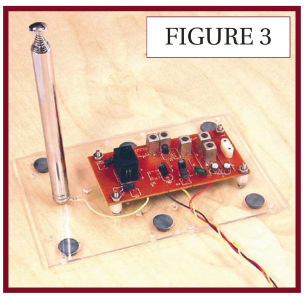


FIGURE 3

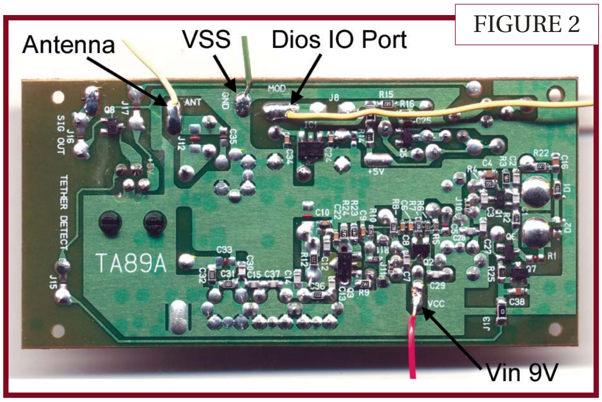
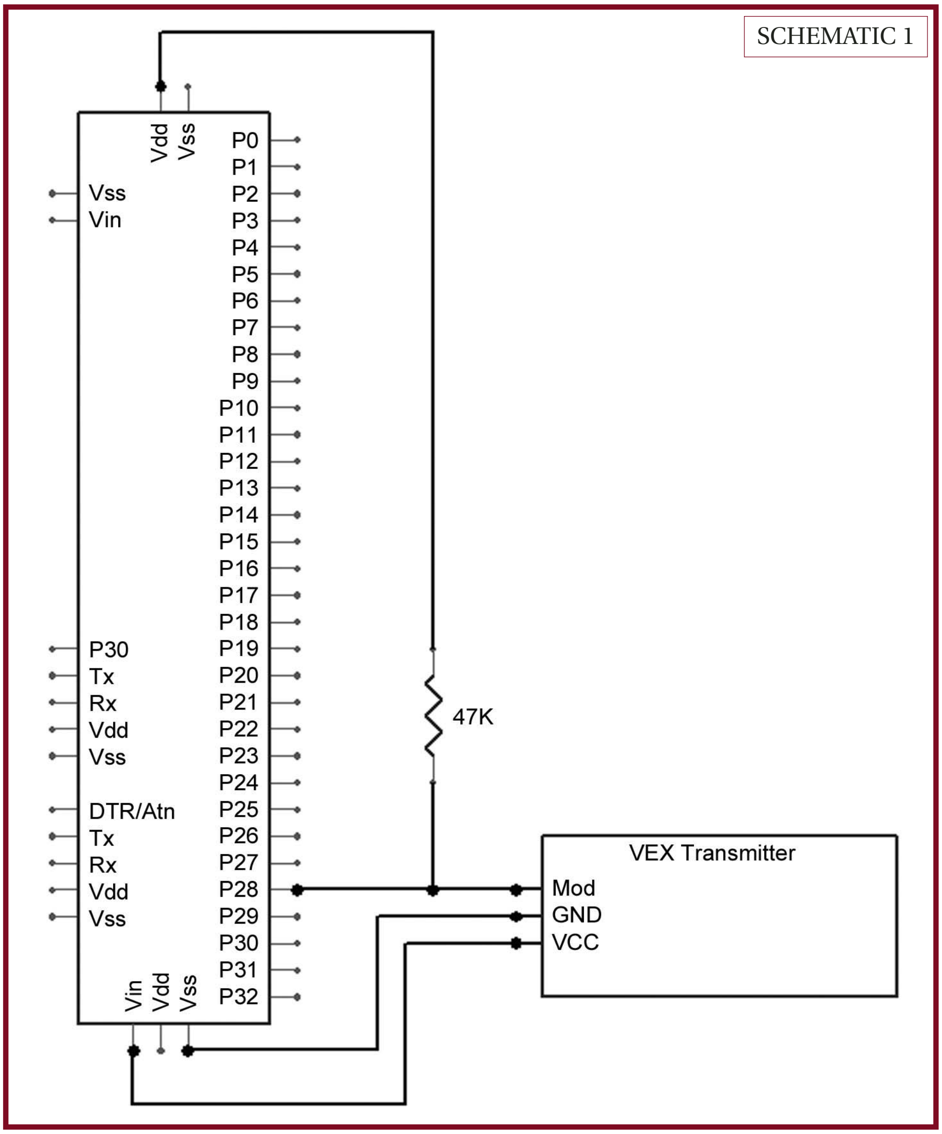


FIGURE 2



SCHEMATIC 1

of the channels, an offset is added to each of the A/D channels. I added a SPDT switch to the remote to allow us to place it in program mode. Tie the center lead on the switch to port 11 and the other leads to Vss and Vdd. When port 11 is high, the program allows you to trim the offset values of the various joystick channels. You can also reverse each axis. The readings can be saved to the DiosPro's EEPROM so that you need only set these values once. The VEXTXanalog.txt program has instructions as to what keypad buttons change which values.

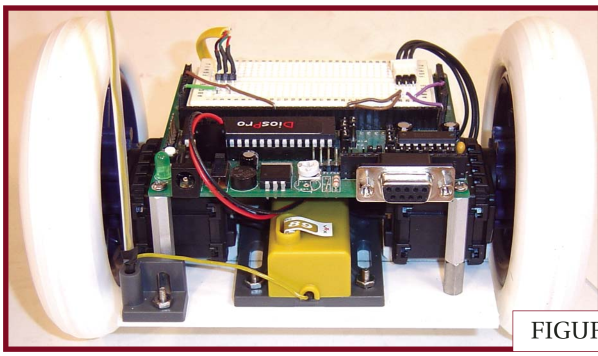


FIGURE 4

To test the transmitter, connect it to the small two-wheeled AX-12 robot we built a few months ago. This bot is a good example because it's simple enough that you should be able to make modifications to any robot application.

The program VEXBOTanalog.txt is meant to be programmed into the actual robot and mates up with the controller program called VEXTXanalog.txt. The Vex receiver comes with all the mounting hardware needed to mount both the antenna and a receiver as shown in Figure 4. I used the small cable that came with the Vex receiver as described last month. You can mount the 22K resistor on the inside or outside of the radio. Schematic 2 shows the complete hookup of the AX-12 servos and Vex radio.

Load the programs and take the robot out for a spin. You have several channels left on the receiver so you can use the buttons and keypad to send special instructions to the robot. For instance, when the ALT key (key 11) is hit, the robot will calibrate itself to the transmitter.

If you decide to use the Vex transmitter in your remote, you need to keep a few things in mind. The original 7.2V battery pack that I suggested last month cannot be used to power the system. You will need to upgrade to a 9.6V or larger pack. The problem is there is not enough room inside the remote to do this. You have a few options, though:

1) Install some Velcro straps to the underside of the remote to hold a larger pack in place.

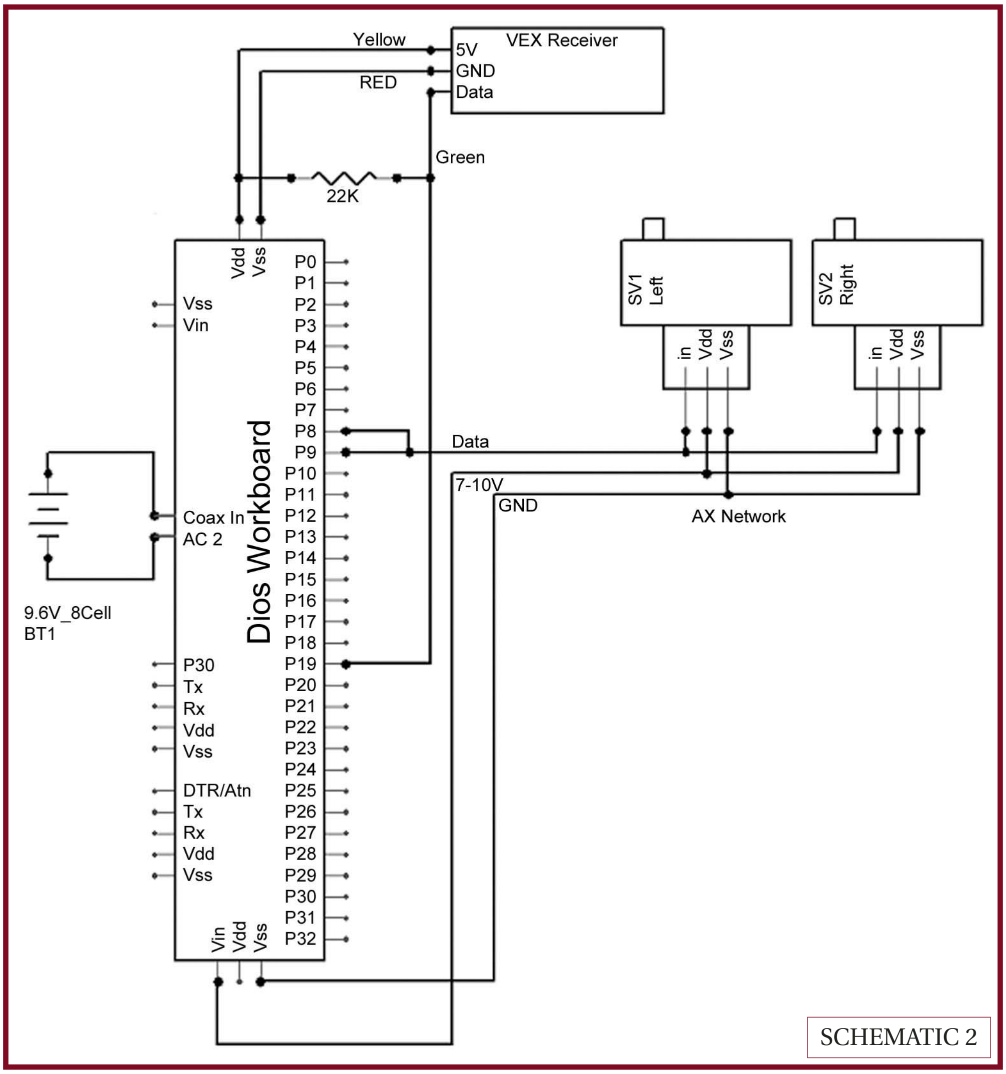
2) Add some additional spacers to the upper and lower bases.

3) Take the battery holder portion of the original Vex transmitter enclosure and attach that to your remote.

As you can see in Figure 1, I chose the third option. I cut the enclosure off so that all the standoffs were intact. I then used some #4 machine screws to attach the enclosure to my base. I drilled a hole to pass the battery connector (also retrieved from the Vex radio), then attached a small header in order to connect to the coax and AC2 pins on the power header on the Dios WorkBoard.

There is space to mount the transmitter just over the serial LCD. It was attached to the rear base between the buttons. I used a small right angle bracket to attach the antenna to the base, as well. I also cut out a small square hole so I could change the transmitter crystal.

All in all, the radio works pretty well. I also included a Bot program called VEXBOTanalog 2.txt. This program works the same as the VEXBOTanalog1.txt except a custom assembly language decoder was built. It does not have the resolution that the original program does, but it is more stable and has a feature where it can detect the loss of the transmitter signal.



SCHEMATIC 2

Final Thoughts on the Vex Interface

The world of analog R/C signals is not a perfect one. There is always a certain amount of drift in the transmitter pulse width. When I created this interface, I added a new command to the DiosPro called "fuzzy" that works very much the same as the lookdown command, except that lookup indexes are supplied in pairs. These pairs define the high range and low range that is acceptable for the various signal pulses used by both the keypad and buttons.

I have included a couple of extra Vex interface programs that let you experiment with creating a full digital interface. The program VEXTXdig.txt encodes the raw A/D signals into a 10-bit, positive parity value that is transmitted to the receiver. The VEXBOTdig.txt is used to decode these signals. The programs are only in the experimental stage and need a bit more work for prime time.

While there is a lot you can do with the Vex transmitter interface, I only show it here so that you can perform your own experiments. While we don't make any changes to the actual transmitter, I'm sure these would be considered modifications and may or may not be allowed by FCC regulations.

Zigbee Interface

Now we're going to open up the real power of our remote control system. By using Zigbee modules, you will have true two-way communications with your robot. This will allow you to transmit the equivalent of several channels, as well as receive telemetry data from the robot. As a bonus, the Zigbee

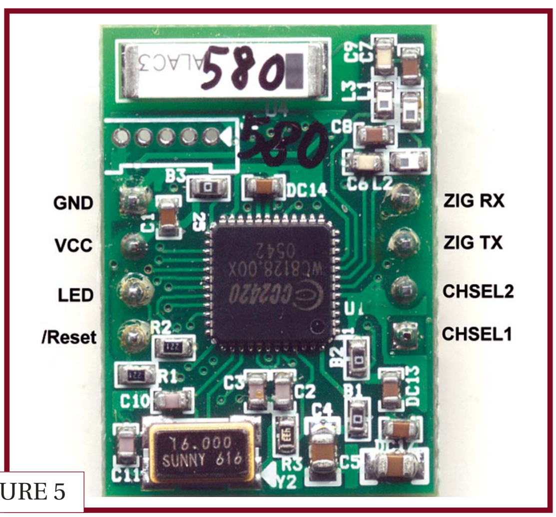


FIGURE 5

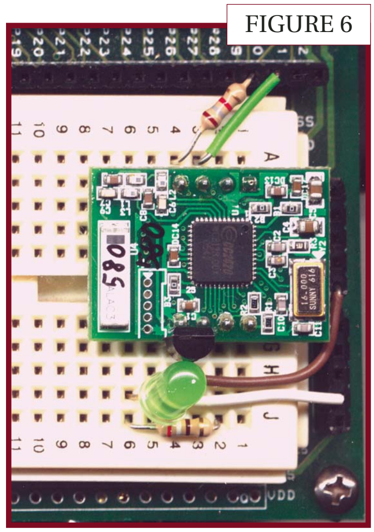


FIGURE 6

FIGURE 7

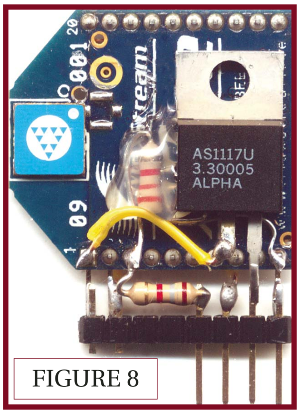


FIGURE 8

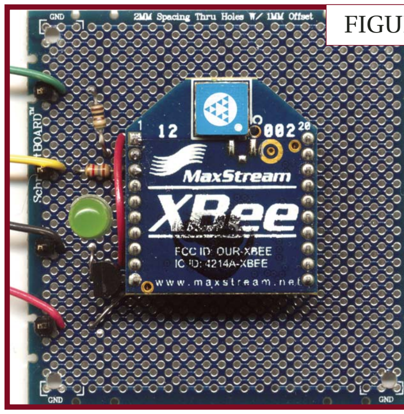


FIGURE 9

modules require less battery power. This allows us to use the original six-cell battery pack described back in Part 1. I also found the Zigbee modules easier to work with and because of their size, I had more options when mounting them in the enclosure. I will show you how to use two different modules from different manufacturers so you can decide which will best suit your needs.

ZIG-100 Module

The Robotis ZIG-100 module shown in Figure 5 is the easiest Zigbee module I have ever had the pleasure to get my hands on. You purchase them in pairs already configured for peer-to-peer operation. This means you don't have to understand any of the underlying Zigbee operations. If this doesn't make the modules easier to work with, they also come with .1" headers installed so that you can plug them right into your breadboard like the module shown in Figure 6. I took these modules right out of the box and wired one into the controller and the other in the AX robot and was up and running in just a few minutes.

XBee Module

The Maxstream XBee module shown in Figure 7 is a very serious Zigbee module. While they are not plug-and-play like the ZIG-100 modules, they do have over 50 configuration parameters that will allow you to create different network configurations, as opposed to the ZIG-100's three parameters. This flexibility does come at the cost of more complexity, however. You must at least have a basic understanding of Zigbee networking and the XBee command set. The XBee module has two 10-pin headers; the down side is that they are 2 mm on center. This adds to a more complicated interface. If you want to use this module, you'll have to build some sort of interface board.

Zigbee Interface Board

Zigbee modules operate at 2.7-3.6V. In order to mount and interface 5V circuits to these modules, we need to build a board. For the ZIG-100 module, this is pretty easy considering the headers are .1" on center so they will plug into just about any PCB (printed circuit board) you can pick up at your local RadioShack. Not so with the XBee modules. You have a couple of choices; you can add components directly to the module as shown in Figure 8, or use one of the Schmartboards shown in Figure 9.

You can purchase these boards online at **www.schmartboard.com/index.asp?a=11&page=a_products_th&id=1**.

The part number for the 2 mm board shown in Figure 9 is #201-0002-01. For the ZIG-100 board shown in Figure 10, I used #201-0001-01. Step-by-step instructions showing how to build these interface boards are on my website at **www.kronosrobotics.com**

If you decide to attach a header as I have done, you need to make sure you leave the original pins intact. They are needed to update the firmware on

SCHEMATIC 3 — XBee Interface

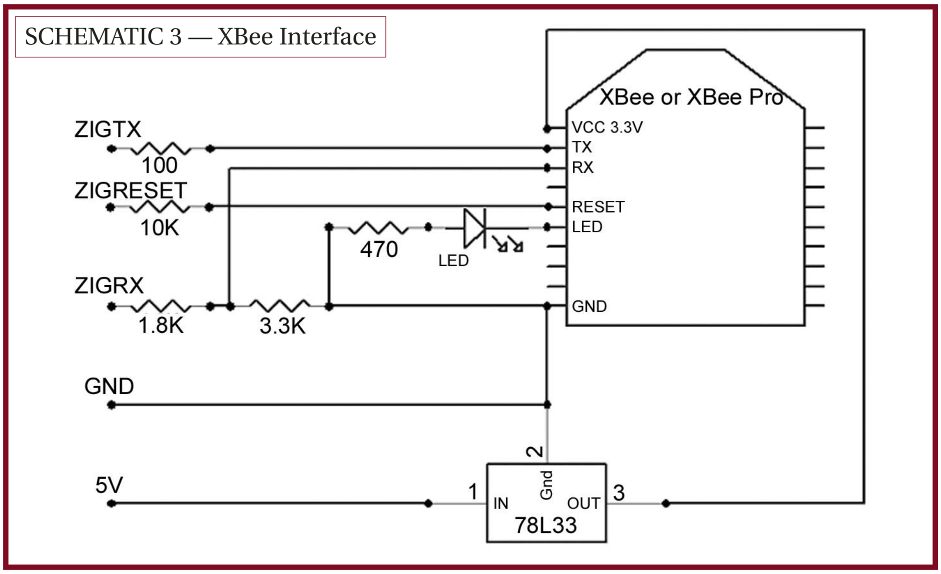
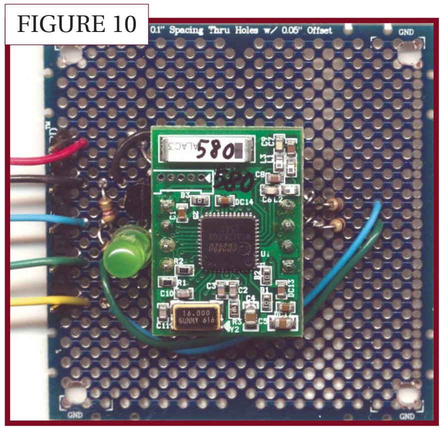


FIGURE 10



the chip and provide support for other operations.

I wanted to create a common interface for using both types of modules, so the controller and robot interface are the same, regardless of the type of Zigbee module you use. Schematics 3 and 4 show the schematics for both the XBee and ZIG-100 module interfaces. You omit the LEDs if you don't want a connection indicator. On my controller, I mounted the LED on the top base. The 1.8K/3.3K resistor pair on the ZIGRX lead is used to drop the 5V I/O port to 3V in order not to damage the module. The 100 ohm resistor on the ZIGTX is not needed, but is there in order to keep a short or other misconnection from damaging the module. If you are going to mount one of the modules on a noisy robot, you may want to add some 100 µF caps to both sides of the regulator.

The ZIGREST lead is only needed for configuration and is not used during normal operation.

Module Status LEDs

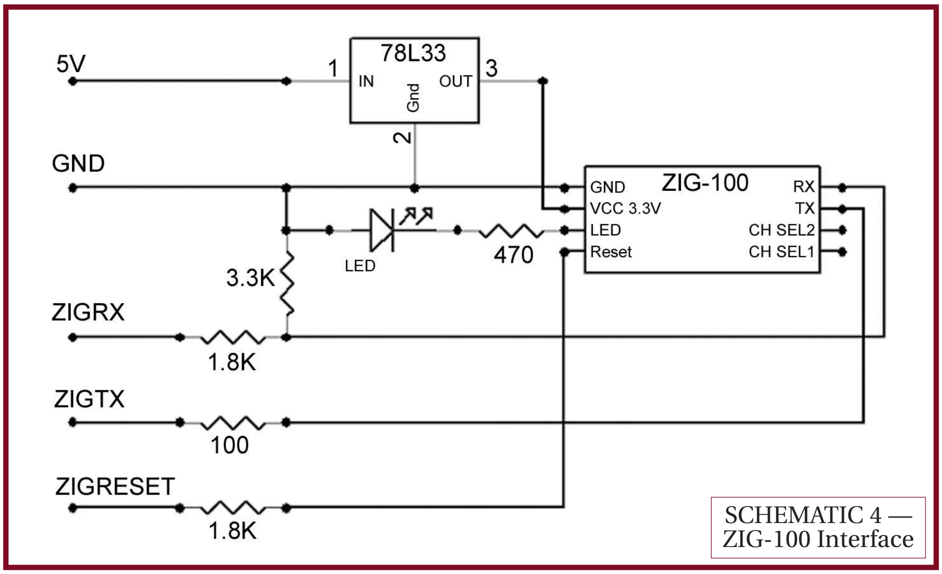
The way the status LED works is different for both modules. On the ZIG-100, the LED flashes when the unit is not connected to another module and goes solid when a connection is achieved. On the XBee, the LED lights when the module receives data and the amount of brightness indicates signal strength.

I also should note that the XBee needs almost 10 seconds from the point both controller and robot modules are turned on to sync up. On the ZIG-100, it takes about one second.

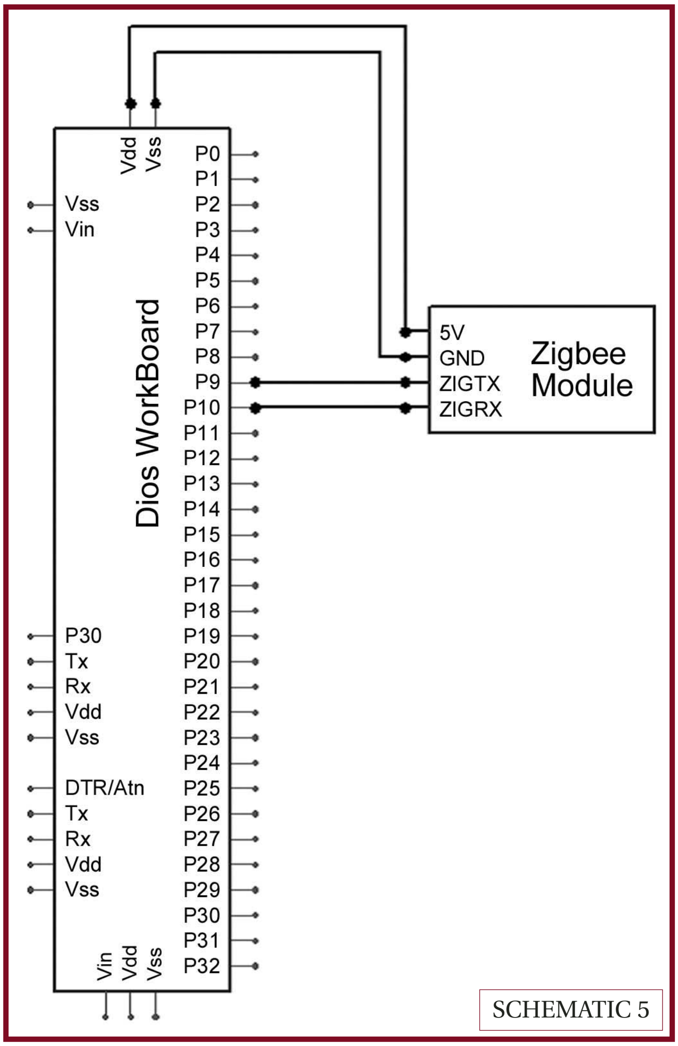
ZIG-100 Library

The DiosPro has libraries for configuring both modules and I have included programs called XBsetup.txt and ZIG100setup.txt to get you up and running as soon as possible. While the ZIG-100 modules can be used immediately without configuration changes, you will need to make changes to the default XBee module settings. To configure either module, connect it to the DiosPro as shown in Schematic 5. Make sure you add the reset lead to port 10 as it is not shown in the schematic. Next, run the included XBsetup.txt or Z100setup.txt program on both the controller and robot modules.

You set the destination address to match the source address to the other module as shown in Figure 11. For example, module 500 sets its destination to 501 and module 501 sets its destination to 500. On the ZIG-100 modules, the source is permanently set and called ID. On the XBee unit, you will need to make up an ID.



SCHEMATIC 4 — ZIG-100 Interface



SCHEMATIC 5

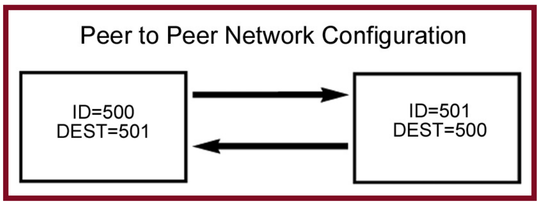
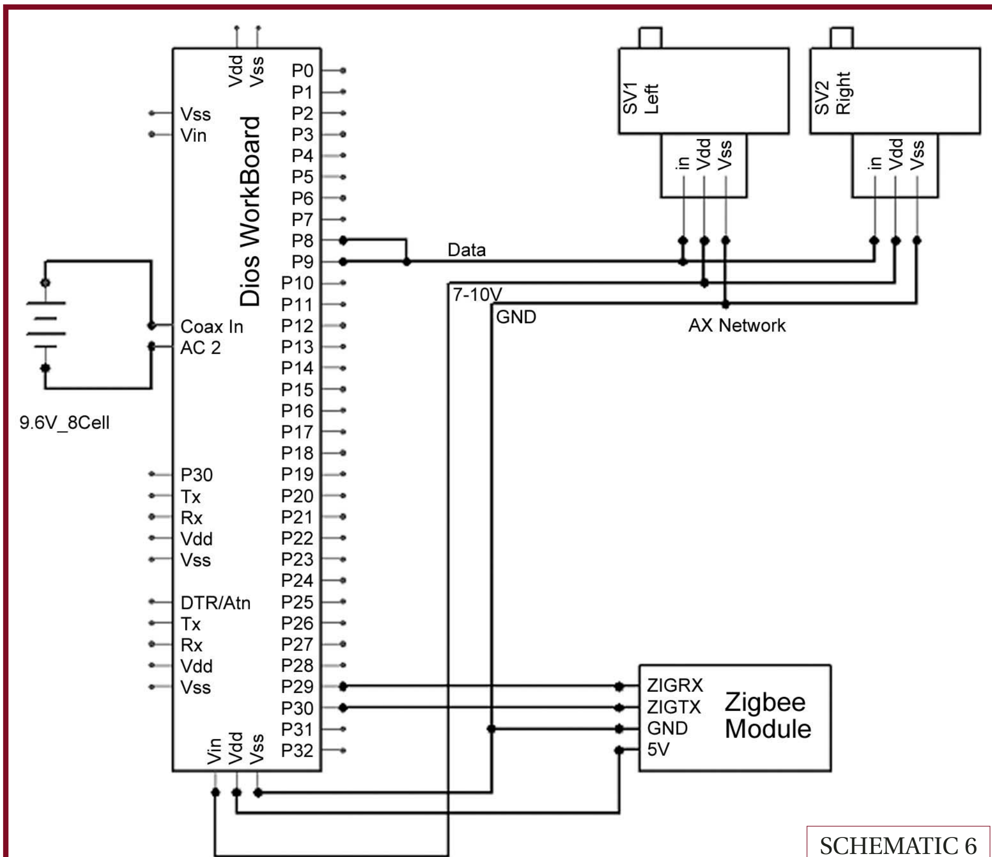


FIGURE 11



SCHEMATIC 6

Zigbee Controller Hookup

Schematic 5 shows the connections needed to add either module to the remote control system.

Zigbee Robot Connection

The Zigbee module connection on the robot is exactly the same as on our remote. The only difference is that the hardware UART is occupied by the AX-12 smart servos so we are going to use ports 29 and 30 on the DiosPro (Schematic 6).

Testing the ZIG-100 Interface

Let's start by loading up the ZIGTX.txt program and programming it into the controller. This program runs much different than the Vex version. The remote actually becomes a slave and monitors the hardware UART port for Zigbee activity. The programmer section has also been upgraded a bit, as well. If the program switch on port 11 is high, it will check the keypad and make changes to the trimmer and reverse options on the controller.

If Zigbee activity is detected, the appropriate command sent by the robot is processed and serviced. The remote is designed to send out a value of 1500 when the joystick is in the center position. The keypad sends a range of 0-12 when requested and the buttons will send a single byte with each of the six buttons representing a single bit.

Load up program ZIGBOT0.txt into a remote DiosPro or robot. In this first program, the program requests data from all the channels on the controller. Use this as a template for adding the code to your own bots. I have created a DiosPro library called URBOT.lib. Just making the call to the URserinit will load the library so that you can access the five main functions: URsergetchannel, URsergetkey, URsergetbut, URsersendLCDchr, and URsersendLCDstring.

The UR (Ultimate Remote) interface is a simple one. The remote robot makes a request via the Zigbee interface to the controller. The controller looks at the appropriate command request and services the request. Table 1 shows the current list of valid commands. You can add your own. Add a new label with the code that handles the request then add that label to the branch command.

Load up program ZIGBOT1.txt into your robot. This program will allow you to control the two AX-12 servos using channels 2 and 4 on the controller. When the program starts, it takes a reading of the two channels and then uses that as a reference for the center

TABLE 1

Command	Description
1	Bot is requesting analog channel 1 data. It is sent back in 2 bytes.
2	Bot is requesting analog channel 2 data. It is sent back in 2 bytes.
3	Bot is requesting analog channel 3 data. It is sent back in 2 bytes.
4	Bot is requesting analog channel 4 data. It is sent back in 2 bytes.
5	Bot is requesting analog channel 5 data. It is sent back in 2 bytes.
6	Bot is requesting analog channel 6 data. It is sent back in 2 bytes.
7	Bot is requesting analog channel 7 data. It is sent back in 2 bytes.
8	Bot is requesting analog channel 8 data. It is sent back in 2 bytes.
9	Bot is requesting keypad data. It is sent back in a single byte 0-12.
10	Bot is requesting button data. It is sent back in a single byte 0-63 with each bit representing one of the buttons.
11	Bot is sending a complete LCD string. First the length is sent, then each of the bytes that are meant to be sent to the serial LCD. This data is just a pass-through.
12	Bot is sending a single LCD character.
13	Bot is sending a single LCD character that is to be routed to the debug port on the controller.

position of the joysticks. The program will also stop the robot if the link between the Zigbee units is lost. This is a safety feature.

Program ZIGBOT2.txt goes to the next level and takes a voltage reading from one of the AX-12s and sends it to the LCD. This allows you to monitor the battery voltage on your robot directly on the remote's LCD display.

Program ZIGBOT3.txt adds even more display data taken from the AX-12 smart servos. The current load and temperature is taken from the servos and sent to the LCD.

Final Thoughts

The real power of the Ultimate Remote will really come into its own once you start building very complicated robots. When I first started using the AX-12 smart servos, I built a very sophisticated biped robot. It had so much functionality I had no way of controlling the thing. Well, the cool part is now I do.

I have played with the range and have been able to control my robots at well over 60 feet. This is good considering I have several other technologies including WiFi operating on the same frequencies.

Going Further

There are quite a few enhancements you could make to the control system. Currently, we are only saving a handful of configuration parameters to the DiosPros' EEPROM. You could save information for five or six different robots.

You could also add a base set of setup commands to allow you to change the ID and channels on the units. On the XBee, changing the channels changes the actual frequency the module is operating on.

You could make the modules accessible on the controller so that it could be used to configure all your modules.

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

PARTS LIST

AVAILABLE FROM KRONOS ROBOTICS

- DiosPro Chip (#16148)
- Dios WorkBoard Deluxe (#16452)
- DiosCompiler (a free download from the Kronos website)
- 3.3V regulator (#16528)
- 1.8K resistors (#16529)
- 100 ohm resistors (#16187)
- 470 ohm resistors (#16177)
- 47K resistors (#16194)
- 22K resistors (#16182)
- LED (#16235)
- LED clip (#16239)

AVAILABLE FROM ALL ELECTRONICS

- Six-Channel Transmitter Receiver (#JS-6)

AVAILABLE FROM CRUSTCRAWLER

- ZIG-100 Kit

AVAILABLE FROM SCHMARTBOARD

- 2 mm board (Used with XBee Modules) **www.schmartboard.com/index.asp?a=11&page=a_products_th&id=1**
- .1" board (Used with ZIG-100 Modules) **www.schmartboard.com/index.asp?a=11&page=a_products_th&id=24**

AVAILABLE FROM MAXSTREAM

- XBee — **www.maxstream.net/products/xbee/xbee-oem-rf-module-zigbee.php**
- XBee Pro — **www.maxstream.net/products/xbee/xbee-pro-oem-rf-module-zigbee.php**

WEBSITES

Kronos Robotics
www.kronosrobotics.com

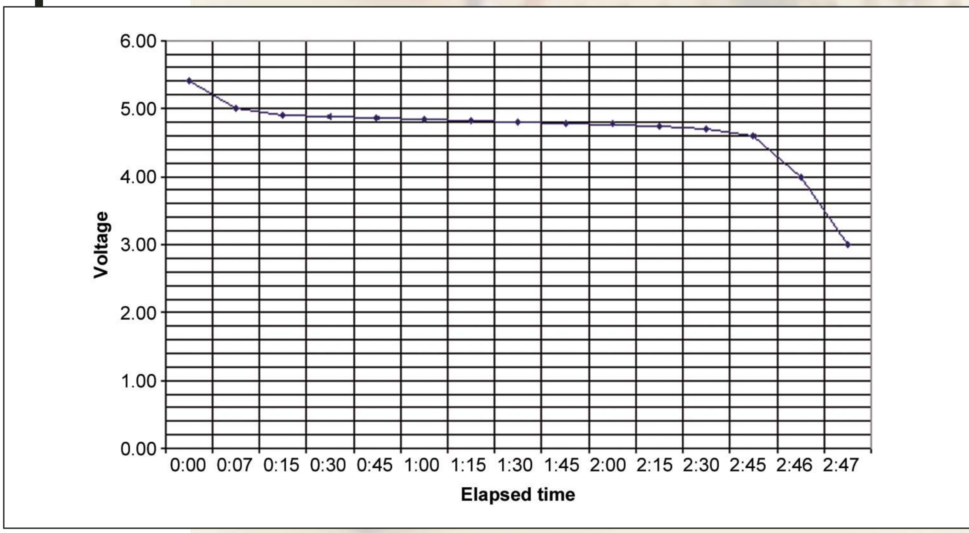
All Electronics
www.allelectronics.com

CrustCrawler
www.crustcrawler.com

SchmartBoard
www.schmartboard.com/mscva

MaxStream
www.maxstream.net

Measuring the Capacity of NiCad and NiMH Batteries



by Paul J. Weijers

FIGURE 1. Typical NiCad discharge curve.

Remote-controlled systems as used in robots, cars, boats, and model airplanes typically use a 4.8 volt battery pack for the receiver and servo system and a 9.6 volt pack for the transmitter. We just trust that the manufacturer has matched the power packs to the system's requirement. What we don't know is how much capacity the packs have under normal operating conditions and after repeated use. That lack of knowledge can lead to some embarrassing and sometimes dangerous situations.

Every so often, we need to test the battery packs to determine if the original capacity is still available. The simple solution is to test the battery pack under near normal working conditions and measure the time it takes to reach a pre-determined minimum acceptable voltage level. There are, of course, systems on the market that do just that. They have a wide range of settings for a large number of battery types. Probably a good investment for someone with a large number of battery packs, but

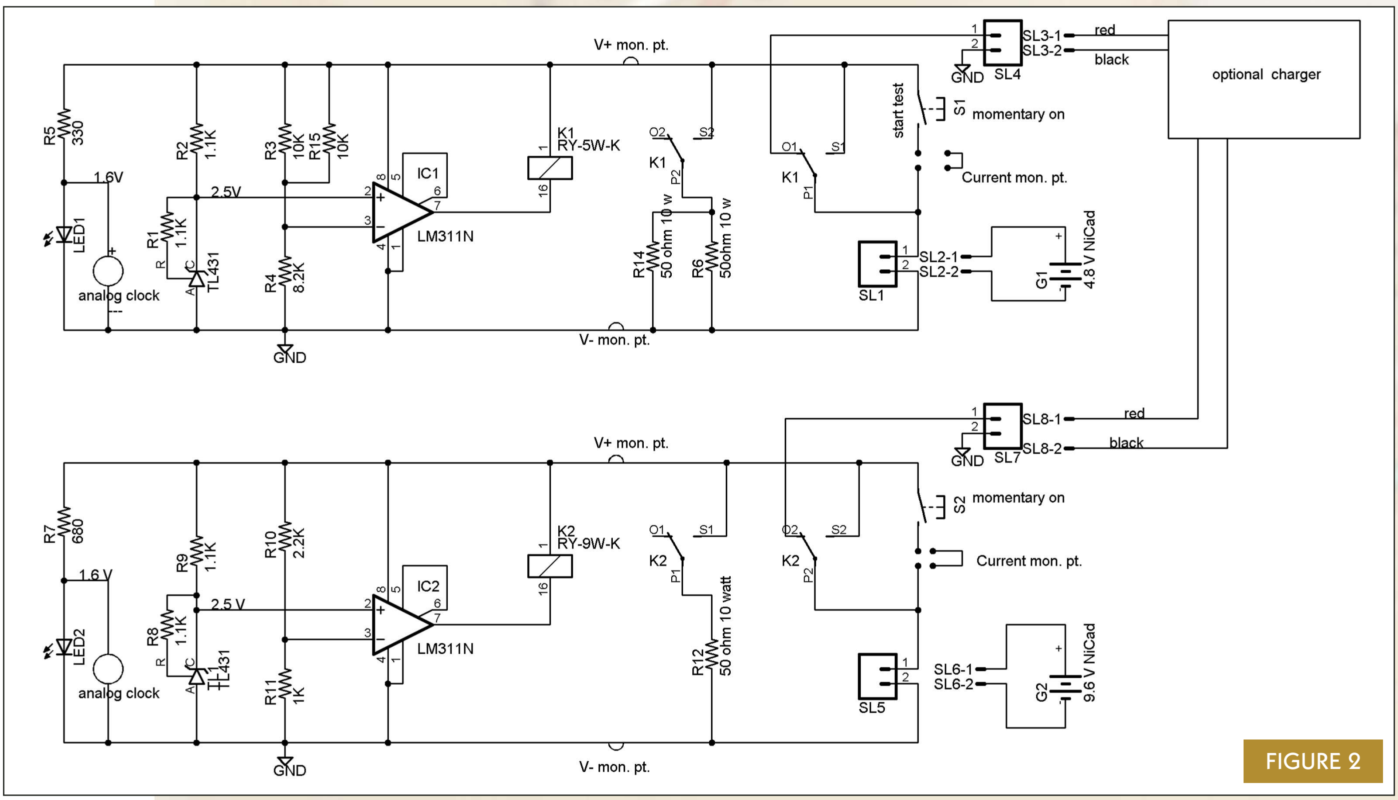


FIGURE 2

for the occasional user with only one or two systems, there needs to be a better solution.

Solutions

The capacity of nickel cadmium (NiCad) and nickel metal hydride (NiMH) batteries is determined by the manufacturing process and is stated in milliampere hours (mAh). The specified mAh holds true for a specific discharge current. There is also the question of the voltage level at which the battery is considered discharged. The normal cell voltage is 1.25 volts. However, some will argue that it is considered discharged at 1.0 volts while others consider 0.8 volts as the point of complete discharge.

The first item to determine is the current drain of the transmitter and the receiver/servo system under normal operating condition. (You can check the specification in the system's manual.) The transmitter typically uses about 200 to 240 mA. The receiver may run at about 20 mA. A servo at rest takes only a few milliamperes, but can be a considerable drain on the system when it is moving. In other words, at the receiving side, the current drain can be rather unpredictable. I take the transmitter specification as the basis for the capacity test.

The simplest test is to take a fully charged battery pack, connect a suitable resistor across the terminals, connect a voltmeter, and measure the voltage at five or 10 minute intervals. Put the data into a spreadsheet and draw a graph of the battery performance. The drawback to this method is that you spend several hours watching the clock and taking measurements. The result will be a graph like the one shown in Figure 1.

The graph shows us the good and bad sides of NiCad and NiMH type battery packs. On the good side, the voltage starts out well above the nominal voltage (in this case, 4.8 volts), rapidly declines to the nominal level, and then maintains this level consistently over a long period. On the bad side, when the pack runs out of steam, it does so with a bang. In a matter of minutes, the voltage drops below usable level.

In the example shown in Figure 1, I consider the four volt level at two hours, 45 minutes as the danger point. Continue to operate and you are in for a disaster. A safe operating time in the case of a remote-controlled plane would be two hours; in the case of a car, you could go to two and a half hours.

Even though this method is time-consuming, you might consider doing it once for a battery pack to get a good baseline. However, this type of measurement can easily be automated.

In order to do this, I set out several rules to reach my objective.

1) It must be simple enough that it can be built by someone with limited electronic skills.

2) It must be inexpensive.

3) It must be independent of any type of external power.

4) It must be able to measure the elapsed time and keep it in memory.

5) It must not contain a microprocessor. (Done purely for the sake of old school. I have nothing against microprocessors.)

6) The discharge must stop at four volts for a 4.8 volt pack and at eight volts for a 9.6 volt pack. This corresponds to one volt per cell.

The Schematic

Refer to Figure 2. We'll discuss the top part of the schematic first — the 4.8 volt section. After that, I'll just point out the differences in the 9.6 volt section. The first order of business is to get the time piece. Find a small electric clock that normally runs on a 1.5 volt alkaline battery. Remove the battery and replace it with a half-inch wooden dowel, about 1/16" shorter than an AA battery. Cut two small pieces of printed circuit board and glue these to the ends of the dowel. This will be a lot easier to attach wires to than trying to solder to the terminals in the clock. See Figure 3 for a close-up of the AA replacement.

The power for the clock is supplied by LED1. Most LEDs operate on two volts, but there are various types available with a 1.6 volt rating. Interestingly, an LED acts like a voltage regulator, so even though the rail voltage will change from about five to four volts, the clock will see a constant voltage.

The next part is the comparator. We'll use the LM311 for several reasons. First, it is widely available and inexpensive. Second, the output is an open collector and thus can drive a small relay directly.

One input of the comparator must be held to a constant voltage; the only restriction is that it must be less than the cut-off voltage, so we'll use 2.5 volts.

Since a 2.5 volt zener diode is not that easy to find, we'll use a TL431 — an adjustable shunt regulator. This device allows us to set a constant voltage from 2.5 to 36 volts and is perfect for use in low power circuits.

While the negative input to the comparator is higher than its positive input, the open collector output of the comparator is drawing current, energizing relay K1. When the battery drops to four volts, the voltage divider made up of R3, R4, and R15 brings the negative input down to 2.5 volts. At this point, the output

FIGURE 3

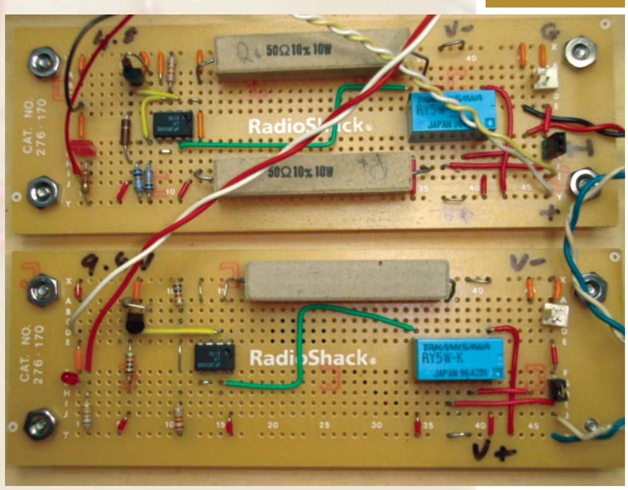


FIGURE 4

stage no longer draws current and K1 de-energizes.

K1 is a double pole relay. While the relay is energized, pole P2 connects a 25 ohm load to the battery while pole P1 connects the battery to the circuit. The load is made up of two 50 ohm, 10 watt resistors. Five watts would do, but I use whatever lies around the shop first before going out and buying more stuff. (See Rule 2.)

While K1 is de-energized, pole P1 connects an optional charger to the battery. This way, you can start charging the battery directly after the discharge is completed.

To start the operation, momentary switch S1 connects the battery to the circuit. If the battery voltage is above four volts, the relay energizes and pole P1 keeps the battery connected to the circuit until the battery drops below four volts (by a few millivolts). At this point, the voltage is removed from the circuit, the clock stops running, and the battery connects to the optional charger.

Differences

There are some minor differences between the two circuits. LED2 is the same as LED1 but the series resistor is double the value, i.e., R5 is 330 ohms but R7 is 680 ohms.

The reference voltage is 2.5 volts. It could be higher as long as it is below eight volts, but you have to add one more resistor and recalculate R8 and R9. That would also require recalculating the voltage dividers R10 and R11.

Relay K1 is a five volt relay whereas K2 is a nine volt relay. Any good DPDT relay will do. The important specification to consider is the 'must operate voltage' needs to be less than the desired cut-off voltages of four or eight volts. For the 4.8 volt pack, the primary load resistor is 25 ohms; R14 and R6 in parallel. For the 9.6 volt pack, the load is R12 at 50 ohms.

Construction

Only standard tools and a volt-amp meter are required for this project. The circuits are built on pre-punched circuit boards because it saves having to drill holes since the layout is not critical.

Start with the clock circuit; put a voltage across it and make sure the clock runs. Build the reference circuit and the voltage divider and test. Get that right and add the rest of the circuit. The package is shown in Figure 4. (Note that the wooden box started life as a candy box.) The clocks are cemented onto the lid.

The choice of connectors for the batteries and the optional charger are dictated by the form of the battery packs. I took mine out of an old, obsolete computer. The schematic shows V+ and V- monitoring points are just pieces of wire to connect a voltmeter. The current monitoring points are small pin sets and shorting plugs, again taken from an old computer.

Test and Calibration

To test the circuits, I made up some battery packs with alkaline batteries from a dollar store. Do not connect the charger, however, because charging an alkaline battery leads to disaster. Monitor the voltage, then at four volts or eight volts, respectively, the circuit should disconnect itself. Note that when you disconnect the battery and check the open circuit voltage, you may find that it has gone up above the four or eight volts. Don't worry, this is quite normal. If you restart the test, the circuit will disconnect again in a few minutes.

Calibration is very simple. Look again at Figure 1 and note that the longest time of discharge is in a virtually flat part of the curve. Measure the current at the nominal voltage of 4.8 or 9.6 volts. There is more current flowing in the first few minutes when the voltage is between 5.1 and 4.8 volts, and less current flowing in the latter part of the curve when the voltage suddenly drops to four volts. This evens things out so the current measured in the flat part of the curve is a good average. Make a note of the current for each circuit because you'll need that information during normal operation.

PARTS LIST

ITEM	DESCRIPTION		SUPPLIER
Z1-Z2	Shunt regulator	TL431/432	Texas Instruments
IC1-IC2	Comparator	LM311N	National Semiconductor
K1	Relay 5V DPDT	RY-5W-K	Fujitsu
K2	Relay 9V DPDT	RY-9W-K	Fujitsu
LED1, LED2	1.6 volt LED		
S1	Switch, momentary ON		
S2	Switch, momentary ON		
Clocks	1.5 volt	Dollar store	
R1, R2	Resistor 1.1K	1/4 watt	
R3	Resistor 10K	1/4 watt	
R4	Resistor 8.2K	1/4 watt	
R5	Resistor 330	1/4 watt	
R6, R12, R14	Resistor	50 ohm, 10 watt	
R7	Resistor 680	1/4 watt	
R8, R9	Resistor 1.1K	1/4 watt	
R10	Resistor 2.2K	1/4 watt	
R11	Resistor 1.0K	1/4 watt	
R13	N/A		
Circuit boards			Radio Shack (276-170)
Enclosure	See text		
Connectors	Match to battery packs and/or chargers		

Operation

To operate the tester, start by setting the clocks to a convenient time, say 12 o'clock. Press the start button and let it run. When the clocks stop running, the test is finished. Note the elapsed time. As an example, let's assume the time is two hours, 27 minutes. The average current measured during the calibration is 240 mA. We calculate 2.45 hours X 240 mA = 588 mAh. Allowing for a safety margin, this pack would be good for two hours of operation.

If you are using the optional charger to immediately recharge the batteries, I suggest you connect the charger via a timer. Remove the automatic turn-on and start the timer manually. Typically, a pack may need 15 hours to charge. Add to this the expected time to discharge, and some margin, then set the timer to turn off the power.

Variations

The load or current demand on the battery pack can be varied by changing the load resistors. Remove the load resistors and measure the current at the nominal voltage. It is a simple matter to calculate an appropriate load resistor to bring the total load to the desired value. This system is intended to test only NiCad or NiMH type batteries. I have no experience with Li-Poly types and do not know what the discharge characteristics are.

Conclusion

This unit is simple, inexpensive, independent of external power, measures time, has a memory, does not contain a microprocessor, and stops when the battery has been discharged, satisfying all the original objectives. Now it will be convenient to test battery packs to determine the available capacity. Of course, comments or questions are always welcome. Email me at paul.weijers@videotron.ca SV

Robot Simulation
FOR EVERYONE

by John Blankenship with Samuel Mishal

There are numerous robot simulators available, but most are lacking in one way or another. Some are very sophisticated and therefore difficult to use. Others might be easy to use, but lack the features you want. If you do get lucky and find the perfect program for your needs it could be expensive, especially if a school or robot club wants a copy for every student or member.

This article will provide a very general overview of how simulations can be written, but more importantly, it will show how simulations can be a valuable tool for learning how to program and for developing robotic algorithms. You will also discover how you or your school or club can get a totally free robot simulation language that is both powerful and easy to use.

Making the Case for Simulation

In order to determine what features we want in a simulator, we need to make a wish-list of all the features we might want in a robot. Notice I said in a robot, not in a robot simulation. After all, if the simulation is truly accurate, we should be able to think of it as a real robot not only in how we program it, but how it makes us feel.

Anyone that has participated in building a robot knows the rush that comes from getting a robot to solve some problem on its own, even if that problem is something simple like following a line or avoiding obstacles as it wanders around a room. A good simulation language should give us that same rush; the same sense of accomplishment that we get when programming a real robot.

During my years as a college professor of computer hardware and software, I had my students write and use many robot simulations. As you might expect, working with simulations allowed my students to quickly develop and test algorithms for robot motion and problem solving. The simulations also helped them decide what sensors were needed and how they should be placed before they started actual construction. What you might not expect is that both using and writing simulation software seemed to help my students learn as much or more about programming as they did in their standard programming classes.

Programming classes — especially those for beginners — can be boring and ineffective because the examples used in the classroom often fail to be relevant or exciting. When simulations are involved in the teaching process though, students are writing programs with clearly understandable purposes. Furthermore, because the programs are graphical in nature, students get immediate visual feedback to help them grasp the logic of programming principles.

Simulations are also a fantastic tool for robot clubs. In my work with clubs, I found many members spend most of their time building the hardware platforms that provide the robot's movement. Little time is spent on the sensory capabilities of the robot because novice robot builders generally don't appreciate the need for sensors, let alone understand how to use them properly. And using sensors means learning how to program algorithms that will control the robot's behavior.

The software that controls a robot is only effective if the robot is capable of interacting with its environment. This brings us back to our initial objective. We need to decide what sensors we would like our robot to have. Just a few short years ago, robot enthusiasts needed a reasonable knowledge of electronics just to build a simple infrared system so that a robot could sense objects without physically touching them. Fortunately, today's robotic enthusiasts can equip their robot with a wide variety of sensors without having a background in electronics.

If you examine the ads in any issue of *SERVO*, you will find electronic compasses that provide heading information, GPS devices that can specify the robot's position, and infrared systems that can detect objects in the robot's path or even lines on the floor. There are ultrasonic and infrared systems that can measure distances to objects and digital cameras that can identify and track specified colors.

Perhaps our sensor wish-list should include the ability to determine how much charge is left on the battery. We might also want our robot to have the ability to know when it sees a flashing beacon. One use for such a sensor, for example, would be to place the beacon over a charging station to help the robot find it before its battery finally dies.

Even though we generally don't want our robots to bump into things like furniture and people, we might want our robot to have some physical switches placed around its perimeter. The switches would provide a failsafe means of detecting objects that might be missed by the more sophisticated infrared or ultrasonic systems.

A robot with such capabilities would be the dream of many hobbyists. With so many sensory options, it is easy to imagine dozens, if not hundreds, of challenging situations for the robot to attempt to solve. You could, for example, program the robot to follow a line or wall, wander about in a room while avoiding obstacles, find a path to a goal, solve a maze, or even cover an area as if it was vacuuming the floor or mowing the lawn. Figuring how to use sensors such as these to make a robot able to react intelligently to its environment is what robotics is truly about.

A robot with all these capabilities is certainly attainable today with off-the-shelf technologies. Unfortunately, even with the low cost of technology today, a robot with all of these sensors would still cost more than most hobbyist can afford. Eventually, manufacturers will be able to offer a complete experimental platform at reasonable prices for any situation.

Imagine how motivated students would be to learn programming if schools could provide a robot for each station in a computer laboratory. Imagine how much more effective this concept could be if the school could provide a robot for each and every student to take home. There are many barriers that keep schools from providing robots to all their students, though.

Once you have software that can read pixel data from the screen it could, for example, "look" in front of the robot, moving forward one pixel at a time until it found some color other than the background color of the screen. When simulating a camera, the software could report the color of the pixel found. When simulating an ultrasonic ranging device, the software would report how many pixels were read before an object was found.

In order to simulate a bumper or infrared perimeter detection system, the software would look at pixels at strategic positions around the edges of the robot to see if any are not the background color. In this case, we don't care about the exact color of the pixels or how far away they are. We only care if an object is present or not so that the software can report that a sensor has been triggered by a collision with an obstacle.

Even though the basic principles of creating a simulation system are relatively easy to understand, the details of actually implementing it can be very complex, especially if you want the system to accurately reflect the real world. My initial attempt at developing a simulation system involved creating a collection of subroutines written in the C language. These routines could animate the robot's movements, and read pixel data and interpret it to create the illusion of sensors.

I used the routines when teaching college-level programming classes for the C language and found them to be

> "Anyone that has participated in building a robot knows the rush that comes from getting a robot to solve some problem on its own ... A good simulation language should give us that same rush."

Cost is certainly a factor, but there are many others. Transporting the robot between classes or to and from school would be difficult because of the bulk or weight even without considering the possibilities of damaging the sensors.

One obvious solution, of course, is to provide a simulated robot with all of these sensory capabilities. Creating a simulated robot that moves around on your computer screen is a relatively easy task. Each time the robot moves, it needs to be erased at its current location and redrawn at the new position. Things get a little more complicated if there are other objects on the screen, but if you want to try your hand at animation, you shouldn't have much trouble finding many resources that explain how to do it.

Finding information on how to simulate all the sensors on our wish-list is not nearly so easy, but the techniques are not all that complicated. One basic requirement of the language you use to write a simulator is its ability to read pixel data from the screen. With this capability, the software can use trigonometry to calculate the coordinates of positions around the robot and examine the colors of those pixels to "see" if there are walls or objects in the robot's vicinity. The actual mathematics is interesting and could be the topic for an entire article.

extremely valuable for both motivating students, as well as teaching programming principles. In fact, the results were so successful that I often wished I could implement the system in a simple language such as Basic so that it could be used to teach programming principles to a much wider range of people.

RobotBASIC is Born

When I mentioned my goals to a good friend of mine, Samuel Mishal, he told me of a Basic-like interpreter that he had written as part of another project. After only a couple weeks of collaboration, we were able to incorporate my original routines into his interpreter and add numerous other features and enhancements, as well. Initial tests of the software have shown that it is powerful enough to excite professional programmers and yet simple enough to be used by sixth graders, especially if you don't tell them about all the features of the language. We named our system RobotBASIC.

It has been our experience that users of all age levels and abilities quickly start to relate to the simulated robot as if it were real. Young users find its "video game feel" exciting and seem to learn about logic and programming with minimal effort, especially if they are introduced to the system properly. Experienced programmers love that the simulation reacts so much like the real thing and that they can test programming ideas much faster than they could on a real robot. The simulated robot can be used to develop algorithms for solving all the real-world situations mentioned earlier.

Some robot enthusiasts reported that they got as much satisfaction from programming the simulation as they did programming physical robots with similar capabilities. They also loved how the debugging features of the language can provide insights as to how the sensors are reacting with the environment, which is often very difficult to determine with a real robot.

Sam and I feel so strongly about the value of this tool, that we have decided to allow it to be used by individuals, schools, clubs, and other organizations totally free of charge. The only restriction we place on its use is that it cannot be sold, only given away. You can get your free copy of the software along with numerous demo programs from the *SERVO* website (**www.servomagazine.com**). You can also find out more about RobotBASIC by visiting our website at **http://robotbasic.web1000.com**. The program runs on Windows-based systems (Windows 95 and later).

Sample Applications

In order to demonstrate the simulator's functionality, let's look at how we can use it to develop an algorithm to allow a robot to follow a line on the floor. In order to be concise, these examples are meant to be informational and are not necessarily complete. Fully functional examples are provided in the zipped file when you download the program.

In real life, my students often created a line for their robots to follow by using black electrical tape on white poster board. Photo-transistors were used to create sensors that could detect the absence or presence of the line beneath the robot based on the amount of light being reflected. Each student group chose the number of line sensors their robot would have based on the algorithm they planned to use. I have had students complete the task using only one sensor, or as many as five. The spacing of the sensors is important too, as it needs to be appropriately based on the width of the line.

In RobotBASIC, there are three line sensors, one of which is mounted directly in front of the robot. The other two are spaced 10 degrees left and right of the center sensor. With three sensors, you have many choices for how you want to attack the problem. You could, for example, use only the middle sensor and have the robot constantly swing left and right as it attempts to keep the sensor on the line. Such an algorithm can work, but the robot follows the line with a vibrating, fidgeting sort of motion that is far from efficient.

With a little thought, you might decide that a better solution can be implemented by using the two outside sensors. In this case, the robot should try to keep the line between the two sensors. It can do this by simply turning a little to the right every time the sensor on the right detects the line and turning left when the left sensor is triggered. Let's see how we can implement this simple idea in RobotBASIC.

First, we need to draw a line on the screen for the robot to follow. We also need to create and place the robot on the screen. Look at the code in Figure 1. The graphical commands that draw the line are placed in a subroutine to enhance the readability of the program.

```
// this is a comment   First we draw the line
gosub DrawLine
// place the robot at the beginning of the line
// and face it left 90 degrees
rLocate  200, 71, -90
end


DrawLine:       //The colon indicates this is a label
  linewidth 4
  setcolor Green
  rLineColor Green      // this tells the simulator that
                        // Green areas are not objects
  gotoxy 170,71
  lineto 160,72
  lineto 145,80
  lineto 140,90
  lineto 130,100
  lineto 125,110
  lineto 120,140
  lineto 130,200
  lineto 140,250
  lineto 130,270
  lineto 145,300
  lineto 200,350
  lineto 300,325
  lineto 450,375
  lineto 450,450
  lineto 600,450
  lineto 600,400
  lineto 650,200
return
```

FIGURE 1. This code draws a line on the screen and places the robot at its start.

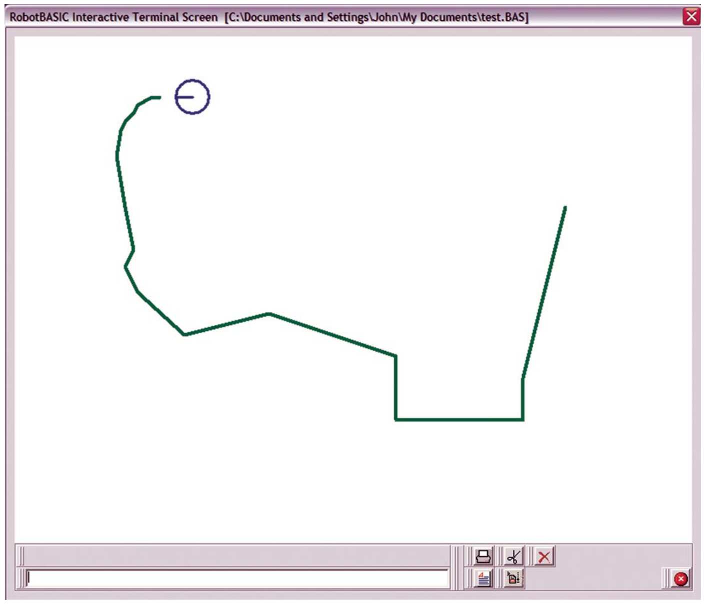
The graphical commands are self-explanatory so I will not discuss them here. RobotBASIC comes with extensive help files and example programs to assist in learning the language syntax. The only item in Figure 1 that might need further explanation is the rLineColor command. RobotBASIC will normally issue an error if the robot's body encounters a color on the screen (collision with some obstacle). Since the robot must be able to move over a line, we must tell the system the color of a line so that it can differentiate it from obstacles.

All of the robot commands and functions in RobotBASIC start with the letter 'r.' The command rLocate x,y,heading creates the robot and places it on the screen at the specified location and heading. The robot's default radius is 20 pixels, so this program places the center of the robot 30 pixels to the right of the start of the line. This means the front edge of the robot is still 10 pixels from the line. Figure 2 shows the output screen when the program is run.

There are only two commands that are used to move the robot on the screen. You can turn the robot either left (negative numbers) or right using rTurn degrees. The robot can be moved straight ahead (or backwards if num is negative) a specified number of pixels using rForward num. RobotBASIC will issue an error if the robot collides with a wall or an object on the screen.

The three line sensors mentioned earlier are all read with the single function rSense( ). This function returns an integer where the three least significant bits represent the sensor data with the least significant being the right-most sensor. We can interrogate the status of a specific sensor by using a binary AND operator as shown in the examples of Figure 3.

FIGURE 2. The robot is ready to approach the line.



EXAMPLE	ACTION
`if rSense( ) & 1`	`true if right sensor sees the line`
`if rSense( ) = 4`	`true if only the left sensor sees the line`
`if rSense( ) & 6`	`true if left OR middle sensor sees the line a = rSense( )`
`if (a = 2)`	`true if only the middle sensor sees the line`
`if rSense( )`	`true if any sensor sees the line`
`if rSense( ) & 7`	`true if any sensor sees the line`

FIGURE 3. The rSense( ) function reads the three line sensors.

Figure 4 shows a RobotBASIC program that can follow a line using the simple algorithm described earlier. The program simply turns right or left, depending on where it sees the line. The RobotBASIC commands while and wend (while-end) create a loop that, in this case, continues forever (or until the user stops the program).

```
rForward 10   // move the robot over to the line
while true
  if rSense() & 1 then rTurn 1
  if rSense() & 4 then rTurn -1
  rForward 1
wend
```

FIGURE 4. This program will follow a simple line.

One problem with the program in Figure 4 is that the robot does not know when it reaches the end of the line. Figure 5 shows how we can solve this problem. Notice this program uses both single line if-then statements and multiline if-else-endif statements. It constantly checks to see if any of the sensors are seeing the line. Since it is possible that the line could be between two of the sensors, the program will continue trying to follow the line until the line is not seen 50 times in a row.

```
c=0
rForward 10     // move the robot over to the line
while c<50      // exit loop if line not seen for 50
tries
  if rSense() & 1 then rTurn 1
  if rSense() & 4 then rTurn -1
  rForward 1
  if rSense( )  // if any sensor sees the line
     c = 0      // start the counter over
  else
     c = c + 1 // inc the counter if no line seen
  endif
wend
```

FIGURE 5. This program knows when the end of the line has been reached.

A major problem with the algorithm used to follow the line in Figures 4 and 5 is that it will not work if the line turns sharply. This can easily be seen if you test this algorithm with RobotBASIC. The robot will do just fine if the line is relatively straight, but it will fail when the line turns sharply. In order to solve this problem, we need to understand exactly why it is happening.

The robot fails to follow the line when the line turns faster than the robot is turning — in this case, more than about a 45 degree change because the robot is turning about one pixel left and right for each pixel that it moves forward. When this happens, the robot moves past the turn and will not see the line on any of the sensors. It will continue moving forward, but it will have already lost the line.

There are relatively straightforward approaches to solving this problem. We could, for example, make sure the robot stays on the line by ensuring that it does not move forward past a turn in the line. It could do this by turning more until it is safe to move forward.

An alternative solution would be to let the robot move past the turn in the line, but give it a way to find its way back to the line. Either of these two algorithms could provide a possible solution for the robot, but they do it in a very different manner and that will be reflected in the behavior we see as the robot attempts to follow a line.

In the first case, the robot will appear to slow down when it gets to sharp turns because it should turn more than it is going forward. In the second algorithm, the robot should constantly move forward, but try to make its way back to the line after it has lost the line in a sharp curve. You might think that the second algorithm is much better. After all, it should allow the robot to reach the end of the line more quickly if we can implement it properly.

Imagine for a minute though, that the robot in question is in the form of a car that is driving down a road and not just following a line on the screen. The second algorithm would indeed let the car take a shorter path to the end of the road, but it does so by letting the car take short-cuts by driving off the road when the road makes a sharp turn and then getting back on the road a little further on.

It is important to realize that neither of these algorithms is necessarily better than the other. Each one would have advantages and disadvantages for different situations and different environments. One of the great things about using a simulation is that it allows you to test and improve a variety of algorithms very quickly.

Figures 6 and 7 show some RobotBASIC code fragments for implementing both of the above algorithms. If you run the programs, you see that the robot behaves exactly as predicted. For simplicity, both of these examples have no checks to see when the end of the line has been reached.

```
while true
  rForward(1)
  while rSense() & 1
    rTurn(1)
  wend
  while rSense() & 4
    rturn(-1)
  wend
wend
```

FIGURE 6. This program will keep the robot on the line even when it turns sharply.

Let's compare Figure 4 and Figure 6. The program in Figure 4 lost the line in a fast turn because the robot only checked one time (with an if statement) to see if it needed to turn. In Figure 6, the robot uses a while-wend loop to continue to turn as much as it needs to before moving forward.

```
while true
   if rSense() & 1
      rTurn 1
// remember which direction we WERE turning
      r = 1
   endif
  if rSense() & 4
      rTurn -1
// remember which direction we WERE turning
      r = -1
  endif
// since we don't care if we loose the line,
// move forward twice
  rForward 2

// if we lost the line make a BIG turn back
// towards it
  if rSense() = 0  then rTurn 3*r
wend
```

FIGURE 7. This program will let the robot find the line after it has lost it in a turn.

The code in Figure 7 is slightly more complicated than the previous examples. Remember, in this situation, the robot should try to find the line again after losing it in a sharp turn. Once the robot has lost the line, it cannot use the line sensor to determine which way to turn. To solve this problem, we need a way for the robot to remember which way it was turning the last time it saw the line. This will normally be the direction the robot should turn if it has lost the line. Each time the robot makes a normal turn (the if decisions in Figure 7), the program uses the variable r to remember which direction the robot was turning. Then, later in the program, if none of the sensors are on (indicating we probably have lost the line), the robot will be able to head back toward the line.

In this example — especially since we don't care if we lose the line — the robot moves forward two pixels at a time. If we have not lost the line, it will turn only one degree at a time. If the line has been lost, however, then three-degree turns are used to help it get back on course.

Closing Arguments

Notice that these choices for how much to move or turn are somewhat arbitrary. With a little experimentation, you could determine the optimum numbers for your situation. This reminds us of the advantage of using a simulation. With RobotBASIC, you can change the numbers and see how the robot responds very quickly. If you were using a real robot, you might well have to edit the code, re-compile, download to the robot, etc., every time you make a change, making the testing process very time-consuming.

Each time you try new ideas on a real robot, you have to make sure its batteries are charged and take care that it does not scuff up your furniture or damage itself by falling down a flight of stairs – all things you don't have to worry about if you use a simulation.

The RobotBASIC simulation also has the advantage of providing debugging tools that allow you to better assess why the robot might not be responding to its environment exactly the way you predicted. Hopefully by now, you are starting to see that there are many advantages to using a simulation.

Wrap Up

I encourage you to download your free copy of RobotBASIC and run and examine the included demo programs. Study the help files and start writing your own code. If you are in a robot club, I should point out a feature of RobotBASIC that allows you to easily create competitions among members. RobotBASIC uses a point system that keeps track of the number of times that sensors are used and how many movements are made by the robot. This allows your club or school to create a contest to see who can complete a specified task with the smallest number of points.

Let us know if you enjoy using RobotBASIC and what topics you would like to see addressed. **SV**

PERSONALITY *and* AUTONOMOUS ROBOTS

by Bryan Bergeron

An autonomous planetary rover detects a potentially perilous discontinuity in the path directly between it and the nearest recharging station. Should it accelerate and attempt to bridge the gap or turn around and search for a safer — albeit more circuitous — route to station?

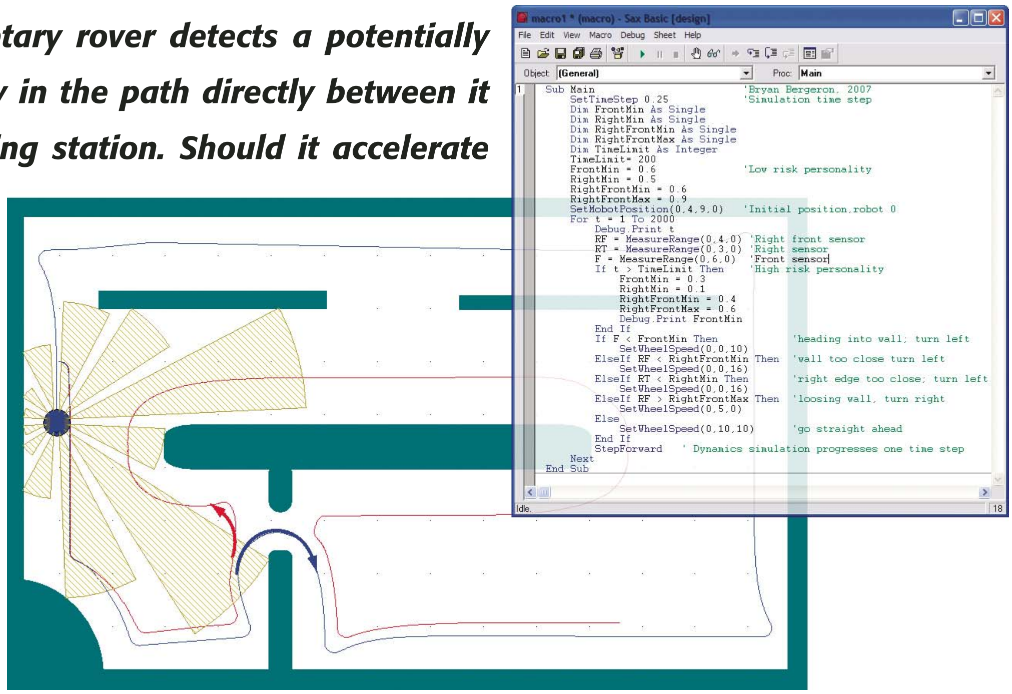


FIGURE 1. Risk-averse (red trace) and risk-tolerant (blue trace) robot behavior. The Basic code that defines the behavior is overlaid in the upper right.

Although a definitive answer requires much more information, if the robot's battery is fully charged and there are no other time constraints, a reasonable course of action is to search for an alternative route. Such risk-averse behavior is inappropriate if the rover has only minutes of battery power remaining, and if failure to reach the station in time results in permanent robot shutdown and mission failure. In this case, despite the risk of damage, the robot should attempt the direct path to the recharging station.

Enabling an autonomous robot to exhibit behaviors such as context-appropriate risk aversion can be achieved by creating a table of links between every possible scenario and the corresponding behavior response. However, if the robot designer fails to foresee every scenario, then the robot may act unpredictably when faced with an undefined challenge.

A more robust approach is to provide the robot with a personality — that is, a default set of behaviors. The particular mix of behaviors displayed by the robot should reflect the robot's internal state, such as battery charge, and factors in the environment, including obstacles and threats. In anthropomorphic terms, the autonomous robot should be able to modulate its personality to emphasize behavior traits such as passivity, aggressiveness, and risk aversion.

This article explores the concept of robot personality using an inexpensive robot simulation environment. Continue reading to discover how personality can enhance the effectiveness of your autonomous robot, whether it's a combat robot, carpet rover explorer, or aerial vehicle.

PERSONALITY

We commonly ascribe personalities to others, their pets, and other animals as a form of shorthand for describing the collection of behaviors that they exhibit. These behaviors may be instinctive or learned by interacting with and adapting to the environment.

Although there are specific personality nomenclatures used by clinical and human resource professionals, ordinary terms such as dominant, submissive, sneaky, confrontational, energetic, passive, gregarious, and loner are commonly used to describe personality types.

Personalities may be context specific, in that the dominant person or leader of one group may be submissive or subordinate in another. This behavioral inhibition or attenuation of a behavior trait as a function of external or internal factors is beneficial in complex collaborative tasks that demand an unambiguous chain of command and authority.

ROBOT PERSONALITY

Whereas a person's personality is

evident by observing their words and gestures, determining a robot's personality — if there is one — is more problematic. In a few rare cases, such as the anthropomorphic Kismet robot [1], facial expressions and body posturing modeled after humans can provide a key to robot personality — or at least the personality that the robot is programmed to mimic.

The overall form and function of a robot can also be used to infer something about the robot's personality. For example, the Machine Interface Personality Profile [2] categorizes a machine's personality along four axes: submissive-dominant; unconscientious-conscientious; emotionally stable-emotionally unstable; and closed-open. Acknowledging the myriad determinants of robot personality, for the purpose of our discussion, personality is defined in terms of gross, observable movements.

There are limits on the degree to which a robot can be provided with anything resembling a real personality. Instantiating an autonomous robot with a human personality — such as the paranoid, suicidally depressed robot Marvin in *The Hitchhiker's Guide to the Universe* — is the stuff of science fiction. That said, biomimetics does have a role in suggesting robot personalities that should result in behaviors that are predictable, effective, and consistent with the understanding of the robot handler.

Animal behavior concepts relevant to robot personality range from complex, stereotypic behavior displayed by predators hunting in packs and by males competing for territory and mates, to simple, non-specific behaviors such as general aggressiveness and inquisitiveness. Behaviors that occur within the context of interspecies interaction, such as predation, parasitism, and competition, also provide a potential mapping for effective robot personalities. For example, competitiveness may be a more effective personality trait than inquisitiveness when time is limited.

An advantage of using established categories of human and even animal behaviors and relationships as the basis for robot personality is the availability of validated mathematical models of personality. Modeling and simulation of personality is used in the fields of clinical psychiatry and psychology to explore complex mental disorders such as schizophrenia, uncontrolled anger and aggression, and depression. Neural networks are often used to model complex clinical behaviors, in part because medical researchers are more familiar with connectionist theories than with complex mathematical modeling. Animal behavior models are also used as the basis for robot personality. For example, the pheromone model supports personalities based on the chemical marker system used by insects.

As described by Braitenberg in *Vehicles: Experiments in Synthetic Psychology*, seemingly intelligent, complex behaviors as abstract as fear and love can be synthesized from simple sensor-motor connections [3]. A modern application of Braitenberg's theories is to arbitrate behavior based on competing behavior groups. The downside of using arbitration and other relatively simple approaches to define robot personality is that they often don't scale well when the environment contains other friendly or hostile robots. For example, in predator-prey pursuit, if every robot in a swarm of autonomous predator robots exhibits the same pursuit behavior, they will probably physically interfere with each other as they converge on the prey.

ROBOT SIMULATIONS

To illustrate the practical application robot personality, let's define several elementary models of robot personality and examine the models in robot simulations. The vehicle for this exploration is MobotSim, a 2D simulation of differential drive mobile robots. The downloadable program is inexpensive at $30, and there is a 30-day/100 uses free trial period. MobotSim provides a graphical IDE supported by the Basic language. For a detailed discussion of MobotSim, see the article on AI Simulation in the February '07 issue of *SERVO*.

The first simulation run, shown in Figure 1, illustrates risk-averse and risk-tolerant robot behaviors. The simulated robot — which appears as a blue disc in the figure — is configured as a right wall follower. The blue and red traces show paths taken by the robot when exhibiting risk-tolerant and risk-averse behaviors, respectively. The red trace shows how the risk-averse robot avoids the relatively narrow passage to the chamber on the right. The blue trace illustrates how the risk-tolerant robot negotiated the passage and moved directly into the chamber.

In this example, risk tolerance and aversion is manifested in the minimum acceptable distance between the simulated ultrasound sensors and the wall or other obstacle. The sensor ranges for risk-averse and risk-tolerant personalities are shown in Figure 2. For example, the minimum right sensor-obstacle distance is 0.5 meters for the risk-averse robot, compared with 0.1 meters for the risk-tolerant robot. A robot that more closely follows a wall is more susceptible to being jammed in a narrow passage and of hitting an irregular wall.

The code listing in Figure 1, which is available with the graphical layout on the *SERVO* website (**www.servomagazine.com**), also illustrates a simple model of time-dependent personality dynamics. The robot initially displays risk-averse behavior, but after 200 timesteps, the robot shifts to risk-tolerant behavior, as in the following pseudocode:

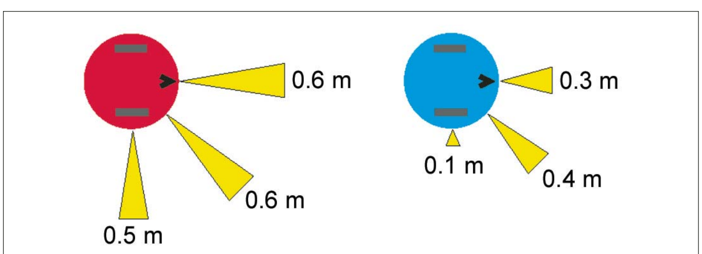


FIGURE 2. Ultrasound sensor ranges for risk-averse (red) and risk-tolerant (blue) simulated robots.

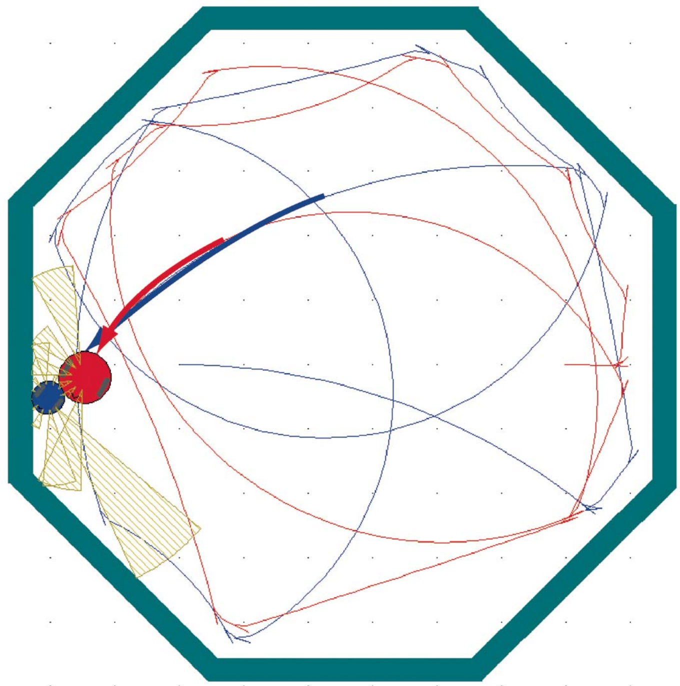


FIGURE 3. Combat robots with aggressive (red) and passive personalities (blue). The larger red robot has the smaller blue robot pinned to a wall.

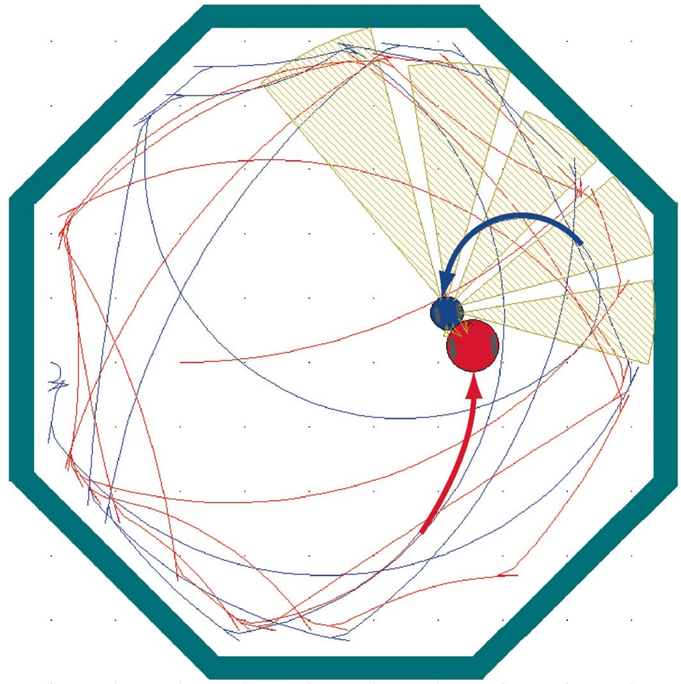


FIGURE 4. The blue robot reverts to aggressive behavior when in close proximity to the larger red robot, given adequate maneuvering room.

```
TimeLimit= 200
FrontMinimum = 0.6                    'Distance limits
RightMinimum = 0.5
RightFrontMinimum = 0.6
For t = 1 To 2000
      RF = MeasureRange(right_front)   'Right front sensor
      RT = MeasureRange(right)         'Right sensor
      F = MeasureRange(front)          'Front sensor
      If t > TimeLimit Then
             FrontMinimum = 0.3
             RightMinimum = 0.1
             RightFrontMinimum = 0.4
      End If
Next
```

In addition to timesteps, the shift in personality could be keyed to battery temperature or voltage, time remaining in a competition, ambient light level, or other combination of internal and external parameters.

A second simulation run, shown in Figure 3, illustrates size-dependent aggressive and passive robot personalities in a combat scenario. The model underlying this simulation modulates the degree of aggressiveness exhibited by each robot as a function of their proximity to each other and their relative sizes.

At the beginning of the simulation, the two robots start near the center of the octagon and seek out each other. When the two robots are in close proximity, the larger robot continues pursuit while the smaller robot retreats. The personality of the larger robot is such that it escalates hostilities and continues until the two robots collide or the larger robot jams the small robot into a wall, as in Figure 3. As shown by the red and blue traces in the figure, when the two robots are within several lengths of each other, the blue robot suddenly cuts to the opposite wall. However, the larger red robot also alters its trajectory and successfully overtakes the smaller blue robot.

The model underlying the behavior of the larger robot is appropriate for an aggressive, expendable combat robot. Conversely, the personality of the smaller robot is more suitable for a non-expendable robot that retreats at the first sign of danger and fights only when there is no other option. For example, Figure 4 shows the behavior of the smaller robot when it is cornered. If the robot has sufficient maneuvering room, it displays aggressive behavior and attacks the larger robot. As you can see by following the traces of each robot, had the blue robot not turned to attack the red robot, the blue robot would have been jammed in the upper right corner of the octagon.

Figure 5 shows the simulation run of two robots of equal size. The robots spend considerably more time chasing and evading each other, as shown by the larger number of traces. In this example, the blue robot tracks the red robot and jams it against the octagon wall. Because robot size can be set interactively from within the graphical environment, the same code (which is available on the *SERVO* site) is used for the simulations shown in Figures 3, 4, and 5.

The triggers used to modulate aggression can be extended to include time, arbitrary quantities such as robot color, or

RESOURCES

MobotSim by MobotSoft — **www.mobotsoft.com**

Models of behavior and behavior disorders — **www.cnbc.cmu.edu/disordermodels/**

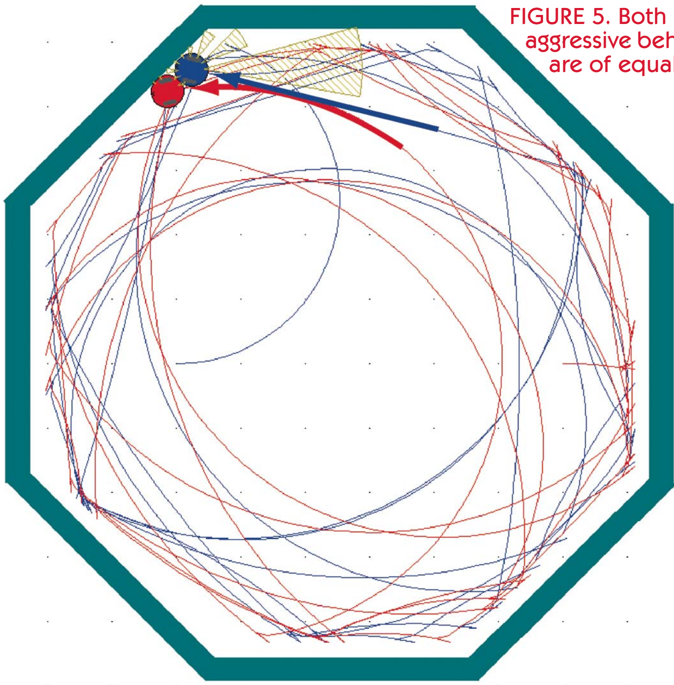


FIGURE 5. Both robots exhibit aggressive behavior when they are of equal size.

indicators of relative fitness, such as internal battery voltage or the speed of the opposing robot. The personality models defined here are simplistic in that the availability of data is presumed. In practice, determining the size and color of the opposing robot, for example, would require a video source and modest image processing.

FROM HERE

Thinking of robot behavior in terms of personality provides roboticists with the same economy of behavior labels that personality profiles provide psychologists. Although the simple behaviors discussed here may be useful for your robotics project, you should extend the models to explore the full potential of robot personalities. For example, consider adding more behaviors, such as those associated with manic-depression, to the models of risk tolerance and aggression. Properly configured and triggered, the resultant, seemingly erratic robot behavior should be difficult for hostile robots to interpret. In a battle, a robot may more easily defeat or avoid a robot whose behaviors are easily interpreted. As such, robot behavior that is easily understood by the controller but less so to hostile autonomous or manually controlled robots can be advantageous in a confrontation.

Robots with user-definable personalities or behavior profiles that can work cooperatively with humans have vast potential in the medical, consumer, and entertainment industries. For example, a team of surgical robots capable of working cooperatively with human surgeons could enable a hospital to handle the otherwise overwhelming surge in surgical load following a major natural disaster or military action. During a robot-assisted procedure, the human surgeon could specify the aggressiveness of the surgical procedure. Alternatively, robots connected to a hospital information system could determine the most effective mode of operation autonomously, based on the patient's diagnosis, patient backlog, and probability of patient survival.

In working with robot personalities, consider the difference between individual and group or swarm behaviors. Following examples in biology, there are two basic personality strategies used to confer advantages to a group. The first is to enable individual robots to exhibit different personalities at different times, as in the examples above. The second method is to assign one personality to a segment of the swarm and another personality to the remaining robots, as in the military and ant colonies. You can explore the relative merit of each approach by defining small swarms within MobotSim or other robot simulation application. **SV**

REFERENCES

[1] C. Breazeal, *Designing Sociable Robots*. 2002: MIT Press.

[2] B. Bergeron, *The Eternal E-Customer: How Emotionally Intelligent Interfaces Can Create Long-Lasting Customer Relationships*. 2001, New York: McGraw-Hill.

[3] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. 1984, Cambridge: MIT Press.

MACHINE ART

I'm a 27-year-old bot-building maniac! I seem to be awestruck, hypnotized actually by automation. I love the construction and design aspects of robotics. I would love to buy all the sweet bots and kits available today, but the truth is, I can't afford it. However, I have come to the conclusion that nothing beats building something with your own brain and hands.

by John Square

I encourage everyone out there to build his or her own robots. I build all mine on my coffee table with basic hand tools. You don't need a huge machine shop. (Although, that wouldn't be too bad to have). You're only limited by your imagination and — of course — your budget. I buy aluminum stock and sheet metal at the big home centers. They have a nice selection at moderate prices. Shop around though as they definitely have different selections. So, in the spirit of home brewing, here's a simple project for all you cash strapped builders.

This project is a collapsible leg covering for my homemade walker bot (Figure 1). As the leg moves back and forth, the covering opens and retracts with a smooth motion. I really wanted a six-legged walker when they first came out. However, I didn't have the $800 to dish out. So I built my own metal monster. I really tried to take the boxy-ness out of it. I feel my leg coverings give it a more insect-like look. Since these coverings are made for my homemade bot, you may need to adjust the size of your coverings to fit the legs on your bot.

The covering is made out of seven strips of varying lengths of sheet metal. Every strip has two holes drilled for mounting. The strips are bent into an arc and attached to a threaded rod. Each strip has two tabs, bent in opposite directions (one tab bent up, and the other tab bent down). When opening these tabs, allow

FIGURE 1. This homemade walker was built for less than $40! (Not including the electronics, though, which were a bit more.)

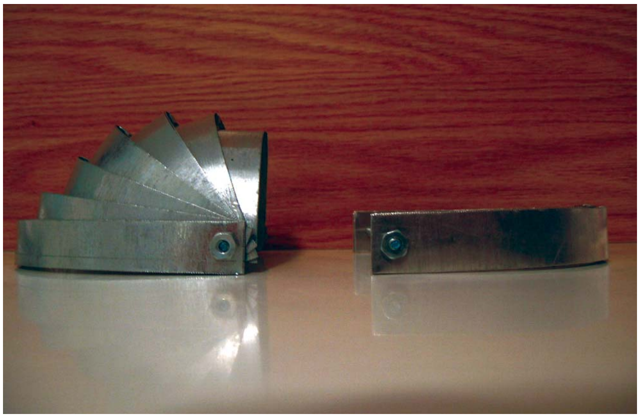


FIGURE 2. The covering opens with an accordion-like action. When closed, it appears to be a single strip.

the first segment to pull on the second, the second segment to pull on the third, and so on. Pulling on the first segment, the covering opens with an accordion-like action. When closed, it appears to be a single strip (Figure 2). You simply mount one side of the covering to the leg and the other side to the body.

The materials required for this project can be found at any hardware store or home center. The parts listed are for one leg covering. I had to build seven altogether for my walker. So you might want to buy a bigger piece of sheet metal and several threaded rods if you're building that many. I used a piece of heating duct I got for free on a job site.

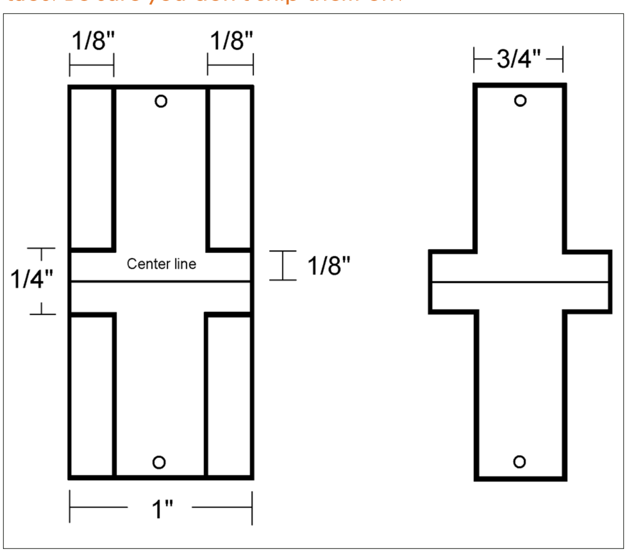


FIGURE 4. Lay out the lines to form the tabs. Be sure you don't snip them off!

Don't be afraid of doing a little cutting. A heating duct is a good source for material and is fairly inexpensive. (Free metal is the best friend to this builder!) So keep your eyes open for freebies. You could also use any gauge as long as you can cut and bend it. Since it's going on the walker, a thinner, lighter, and more flexible piece works best.

Construction

To start, we'll lay out the strips and drill holes. It's a good idea to drill all your holes before you cut out the strips. It makes everything easier. Turn your sheet metal so it's 7" wide and 9" tall. Lay out seven lines one inch apart on your metal. These seven 9" strips will be the individual segments of the covering.

To allow the unit to be collapsible, each segment has to be longer than the previous strip. Starting from left to right, we'll mark the lengths of each strip. Mark a length of 5-1/2" for the first segment. This will be the first and shortest strip. Mark a length of six inches for the second. Continue with half-inch increments ending with a strip that's nine inches long. To lay out the drill holes, make a line a 1/4" up from the bottom. Now simply divide each strip in half (half an inch) and mark the drill holes. Mark the top holes on the strips the same way. Be sure you measure down from your line, not up. All drill hole locations should be the same on every strip (see Figure 3). Drill your holes and cut out your strips.

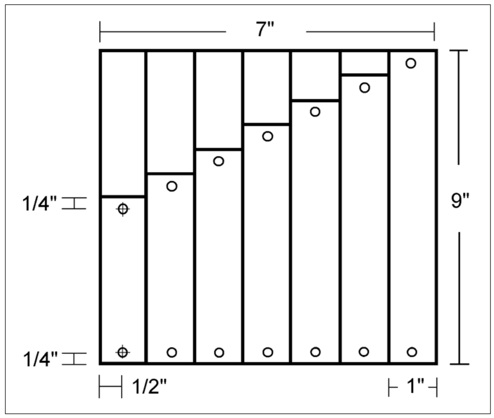


FIGURE 3. Starting with a 5-1/2 inch strip, each strip is a 1/2 inch longer than the strip previous. Ending with the longest nine-inch strip.

Now that we have seven strips of varying lengths, we need to make some tabs. Start with the shortest strip. To keep it simple, the tabs will need to be perpendicular and centered on the edge. We'll have to scribe two lines down the length and 1/8" from both edges. These lines will be the new edge after you remove the material to reveal the tabs. Now divide your 5-1/2" strip in half. Mark a line perfectly centered at 2-3/4". Now, simply measure a 1/8" in both directions on the edges from the centerline, marking the edges of the tabs. Mark the lines to form

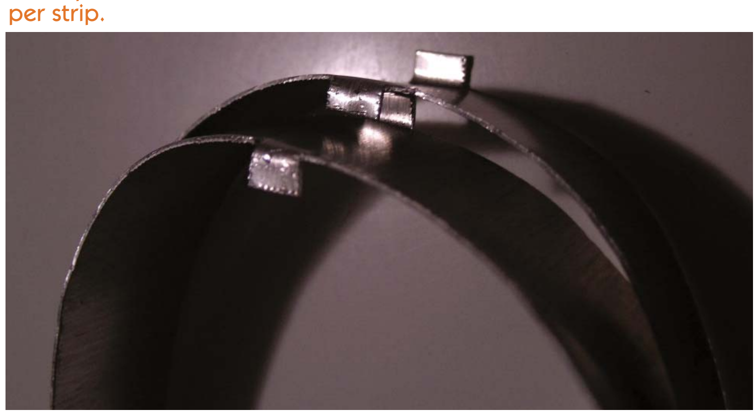


FIGURE 5. Bend the tabs one up and one down per strip.

the 1/4" x 1/8" tab (Figure 4).

Let's do another strip to confirm the process. Take the second strip which is six inches long. Mark two lines an 1/8" from both edges. Measure three inches to the center, dividing the strip and mark a line. Measure a 1/8" from the center in both directions on the edge, creating the tabs. Cut away the material so you end up with a strip that will be 3/4" wide. Continue this process with the remaining strips. Lay out the cut lines, divide the piece in half, mark the tabs, and remove the material. And remember, neatness counts!

Assembly

Finally we can put it all together. Take your 6-32 threaded rod. Twist a nut on about 3-1/2" down. Starting with the longest strip, slide one end down to the nut. Continue with each strip in order, ending with the smallest. Bend the smallest strip into an arc and thread it on the bolt. Continue in order bending and attaching the strips, ending with the longest strip. Attach a nut on top. Now, starting with the smallest strip, bend the tab up perpendicular to the edge. Bend the leading edge down on the second strip, then bend it's opposite tab up (Figure 5). Repeat this process for the remaining tabs. That's it! You're all done!

You will find that you can adjust the size of the covering, too. Using the 12" bolt allows you to adjust the arc span. As the span increases, the clearance between each strip will drop.

This project is easy to scale up or down, depending on your application. As a bonus, the 59¢ bolt can be used for at least two more coverings! I guarantee you'll have one sweet looking walker in the end.

Expansion

You will find all sorts of uses for these coverings. You could use them as shoulder or elbow coverings on your latest automaton. You could build really small ones for eyelids. The uses are only limited by your imagination. I just started fabricating a retractable face shield for my robot Exeter. Check out Figure 6. Driven by a servo, the shield opens and retracts over his eyes. When it opens all the way down, the shield is backlit by using blue LEDs. It gives the impression that Exeter is reading some sort of display behind it. When it retracts, it appears to be one strip. Now go build something sweet! **SV**

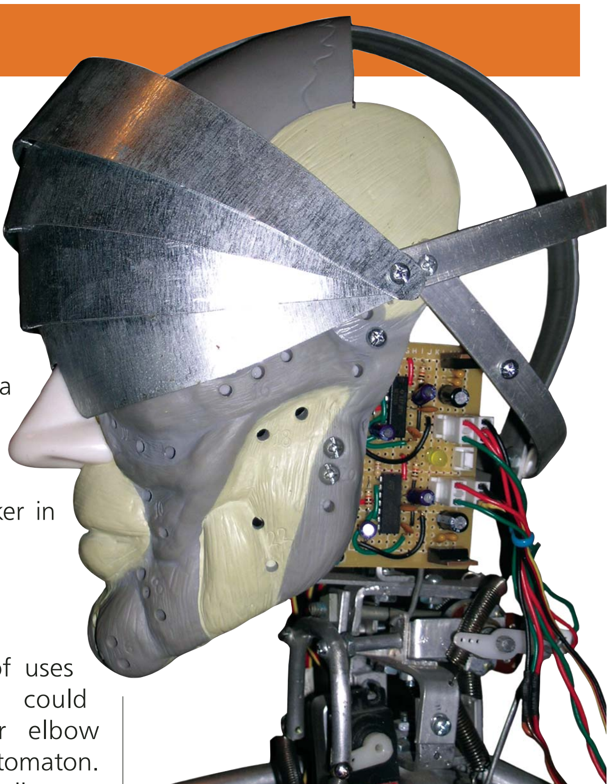


FIGURE 6. My automaton Exeter sporting his new retractable face shield.

PARTS LIST
for One Leg Covering

MATERIALS

Qty	
1	7x9 inch piece of sheet metal
1	6-32 12 inch threaded rod
2	Nuts

TOOLS

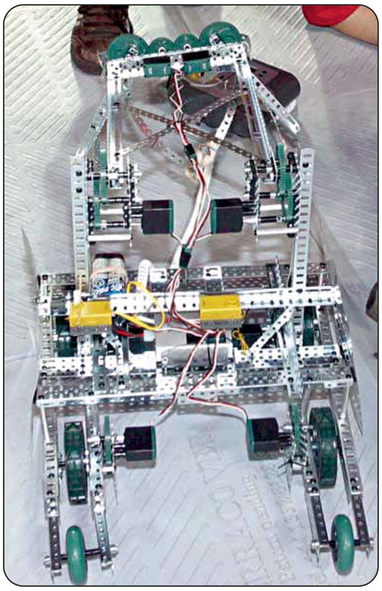
- Power drill
- 9/64" metal drill bit
- Tin snips
- Needle nose pliers
- Ruler
- Pencil, marker

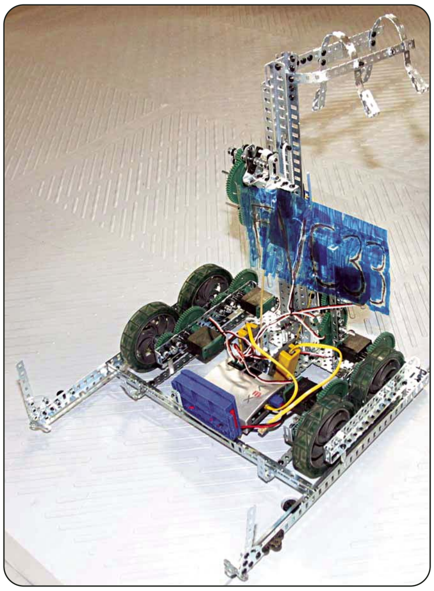
CONTACT THE AUTHOR

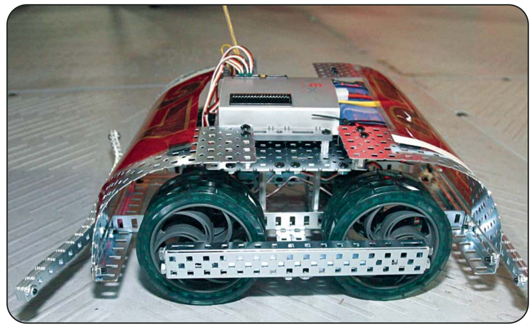
John Square can be reached at www.Squareman456@yahoo.com

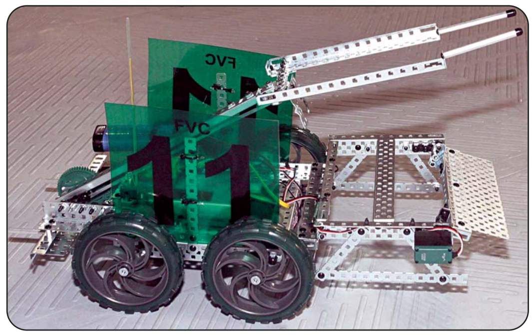
MENAGERIE

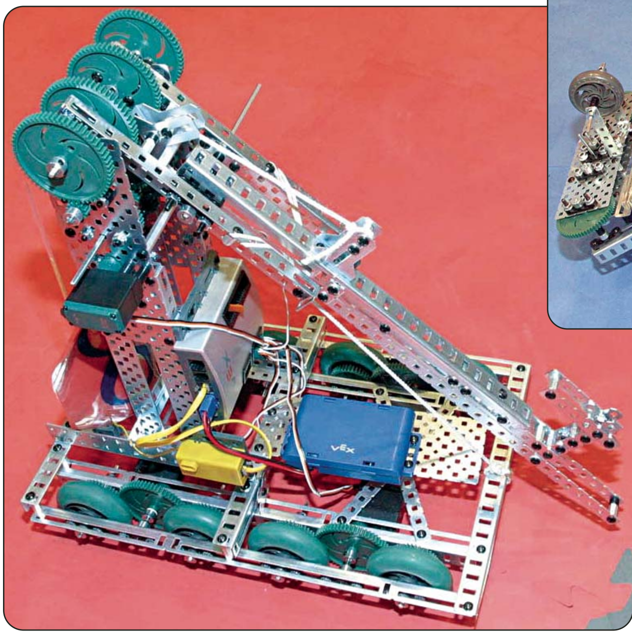
VIVACIOUS VEX VARIATIONS

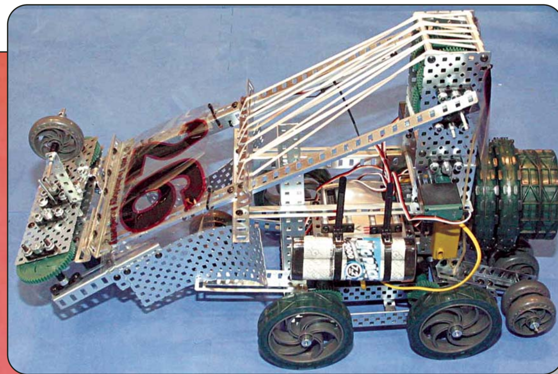


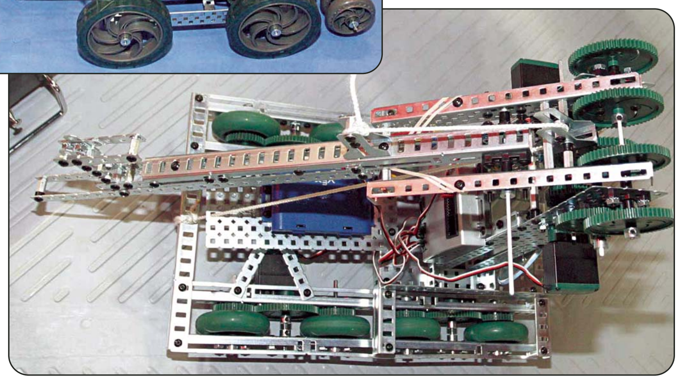






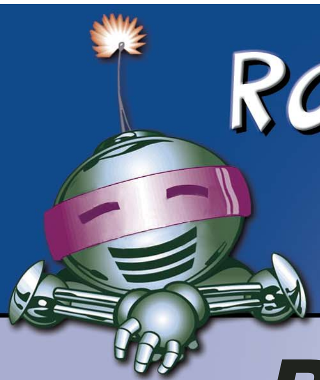






MORE VIVACIOUS VEX VARIATIONS

ROBOTICS RESOURCES



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

BY GORDON MCCOMB

Plastics for Your Robot Creations, Redux

Like clockwork, every few days someone on the various Internet discussion forums asks a question about plastic — you know, that ubiquitous stuff we're all supposed to disdain, but in secret we love because it's so easy to use! While plastic isn't the perfect choice in every robot building situation, more often than not is offers the best mix of cost, ease of use, and durability.

We last covered plastics here in April '03, and it's time we gave it a second, more in-depth look. We'll see what kinds of plastics are best for robotics, and where to find the stuff on the Internet and in your home town.

Why Plastic?

Most robots are constructed using a chassis of some type; either an open framework or a series of decks or bases that together form an expandable platform. The materials used for the chassis are typically wood, metal, or plastic. Of these three, plastic tends to be among the least expensive, the lightest in weight, and the easiest to drill and cut. Of course — and as we'll see — not all plastics are the same, and some are excessively expensive, heavy, and require specialty tools. As a practical matter, we'll ignore these and concentrate on the more commonly used plastics for construction.

Ease-of-work — how well you can drill and cut the material — is an important and often overlooked aspect of materials selection. Metal makes for an excellent robot body, but for most people it's a pain to work with. Even the simple act of drilling a hole may require specialty drill bits, cutting oil (when cutting steel and thicker aluminum), and lots of effort.

On the other hand, there are several kinds of readily-available plastics that are strong and rigid enough for use as a robot chassis, while being easier to work with. A few types — such as expanded PVC (see below) — can even be cut and drilled using hand tools. Granted, plastic will almost never offer the same structural strength as metal, but for the typical small desktop robot, the tradeoff between metal and plastic is inconsequential.

While plastic has a lot going for it, it's not always the perfect choice in every situation. Obviously, when you need durability nothing beats a good hunk of metal. And while many types of plastic are available locally and on the Internet, as a construction material, there are more sources for both wood and metal. Finally, because most plastics are made from oil or oil-based products, fluctuations in the price of oil can impact the cost of plastic. As the cost of oil increases, so does the cost of plastic.

Best Plastics for Robotics

There are literally hundreds of types of plastics, but the following handful are among those that are the most affordable and easiest to get. Most of these plastics are available in sheet form to build robot bases, but many also come in pipe, tube, and rod shapes. I'll list them in approximate order of usefulness in amateur robot construction.

• *PVC* — or polyvinyl chloride — is an extremely versatile plastic best known as the material used in fresh water plumbing and outdoor plastic patio furniture. The raw PVC plastic is clear; it's typically processed with white pigment, but it can be made into most any color you want. The main advantage of PVC is that it is extremely easy to cut and drill, and almost impervious to breakage.

Besides plumbing fixtures and pipes, PVC is supplied in film, sheet, rod, tubing, even nut and bolt form. For robotics, a particularly useful form is called "expanded" or foamed." During manufacture, the PVC plastic is bulked up using an inert gas. This makes the sheet thicker and lighter, yet it's still a hard, rigid material. Expanded PVC — which goes by different trade names such as Sintra and Komatex — is ideal for building small robot bodies. It's become my personal plastic of choice due to its low cost, ease-of-work, and light weight.

• *ABS* (acrylonitrile butadiene styrene) is another plastic commonly found around the home. The most common application for ABS is sewer and waste-water plumbing pipes. Despite its shiny black appearance in plumbing material, ABS is really a glossy, translucent plastic that can take on just about any color and texture. It is tough and hard, yet relatively easy to cut and drill.

You'll find ABS in rods, sheets, and pipes, and in the plastic used to construct LEGO pieces. In sheet form, you'll usually find it colored dark gray or black; the black sheet may have a textured side in order to hide scratches and blemishes. ABS sheet is often used as a protective covering. ABS is a denser, heavier plastic than expanded PVC. It is ideal when you need more structural support than what PVC offers.

You'll probably want an electric drill and saw when working with ABS. Make sure the tools are sharp. Unlike some other plastics (like acrylic), drilling ABS is not as prone to cracking or splintering.

• *Polyethylene* is lightweight and translucent, and is often used to make flexible tubing. It also comes in rod, film, sheet, and pipe form. The material can be reformed with an application of low heat, and when in tube form, can be cut with a knife.

Among its many forms are "high" and "low" density variations, referred to as HDPE and LDPE. I like the HDPE materials, as they are fairly easy to drill and cut, yet the plastic is extremely durable. You probably already have some HDPE in your house — it's commonly used as the material for plastic kitchen cutting boards. A similar plastic — polypropylene — is an alternative when you need a material that is harder and more resistant to heat.

• *Polycarbonate* plastic is known for its durability. Its physical strength makes it a popular choice as replacement pane glass — it's sometimes called "unbreakable" glass, but it's not glass and is breakable. Polycarbonate is one of the easier plastics to find; most home improvement stores carry a selection, some pre-cut into smaller sizes. Common trade names for polycarbonate include Lexan and Makrolon, but there are many others.

While polycarbonate is very durable, it's also hard to cut and drill. As a robot body material, polycarbonate is typically machined to shape using a carbon dioxide laser. The laser beam literally burns through the plastic, leaving a scored edge that can be left as-is or sanded down. (ABS may also be laser cut, but PVC is almost always cut using a mechanical router or a water jet, as burning through PVC releases toxic and corrosive gases.)

As a window glazing material, polycarbonate is typically water-clear, but it's available in most any color, including translucent and opaque.

• *Polystyrene* is a mainstay in the toy industry, and is the plastic most often used when molding plastic toys. Although often labeled as "high-impact" plastic, polystyrene is brittle and susceptible to damage by low heat and sunlight. It is available in rods, sheets, and foam board.

Polystyrene is hard to cut and drill without cracking and breaking, but it is lightweight. If it's thin enough, it's a perfect plastic for vacuum forming. With vacuum forming, the plastic is heated until it's soft, then placed over a 3D mold. A vacuum suction pulls the plastic tight around the mold, and is then allowed to cool. You can find plans to build a small vacuum forming machine on the Internet (try Google and Yahoo! searches), and ready-made rigs are sold for the hobbyist RC modeler market. You can often find polystyrene in thin sheet form in hobby and modeling stores.

Note that while ABS has a styrene component, polystyrene and ABS are two different types of plastics.

• *Acrylic* is a popular choice for building robots out of plastic, but it's not an ideal choice. This plastic can be easily scratched, but if the scratches aren't too deep, they can be rubbed out. Acrylic is somewhat tough to cut without cracking, and requires careful drilling. Otherwise, the plastic can crack as the bit punches through to the other side.

You must also exercise care when using acrylic around electronics. Nearly all plastics (unless they're specially treated) generate static electricity. With its smooth surface and chemical makeup, acrylic is among the most prodigious of static generators. If used near electronics, you may want to lay down a thin sheet of aluminum foil (but be careful of shorts), or use a textured or treated plastic that is not as prone to static buildup.

Acrylic comes in sheets — clear or colored — but is also available in extruded tubing, rods, and the coating in pour-on plastic laminate. Like polycarbonate, acrylic sheets can often be found in home improvement stores. Common trade names include Plexiglas, Lucite, and Perspex.

Identifying Tossed Plastic

If you're into scrounging, you may find the plastic you're looking for in the trash. Many plastics used in consumer products are identified with an ID symbol. Table 1 shows what the ID symbols mean. (Even if you don't find what you're looking for, be sure to take these recyclable plastics out of the garbage, and put them into the recycling bin!)

Finding Plastics Locally

For polycarbonate and acrylic plastics, your local home improvement store is a good — even if limited — first stop. Most sell acrylic or polycarbonate plastic in sheets, in various sizes, in thicknesses starting at about

TABLE 1

Symbol #	Plastic Type
1	PETE — Polyethylene Terephthalate
2	HDPE — High Density Polyethylene
3	PVC — Polyvinyl Chloride
4	LDPE — Low Density Polyethylene
5	PP — Polypropylene
6	PS — Polystyrene
7	A combination of plastics or none of the above.

1/8". The smaller sheets cost more per square inch, but unless you're building a big robot, you don't need much.

Specialty plastics retailers and fabricators offer a broader variety, including PVC, ABS, acetal resin (also known as Delrin, commonly used for machining parts, but expensive), polystyrene, and even others. If you purchase sheet plastics from a specialty retailer, you may have to buy an entire sheet, which is four by eight feet. But most will cut it for you.

If you don't need that much material, most retailers have a "junk bin" of excess plastic pieces; scrounge through it to see what you can find. If you're not sure what kind of plastic you've found, be sure to ask.

Finding Plastics Online

For the widest selection possible, turn to online and mail order sales. The following sources offer all types of plastics, in sizes from small scraps to large four by eight foot sheets. Some also maintain a retail store front where — if you live nearby — you can select and pick up your plastics at the front counter. Bear in mind that some specialty plastic dealers are wholesalers to the trade, and expect quantity orders. However, many of these dealers also will sell (and sometimes give away) scrap pieces.

Advanced Plastics, Inc.
www.advanced-plastics.com

Wholesale distributor of a variety of fiberglass reinforced plastic products. Local outlets in Tennessee, AL, and St. Louis, MO.

Advantage Distribution
www.advantagedistribution.com

Advantage Distribution carries plastic products, including acrylic sheet, rigid expanded PVC (Sintra), corrugated plastic. Also sells Bienfang Foamboard, Pillocore Foamcore, and Ultra Board (polystyrene foam core).

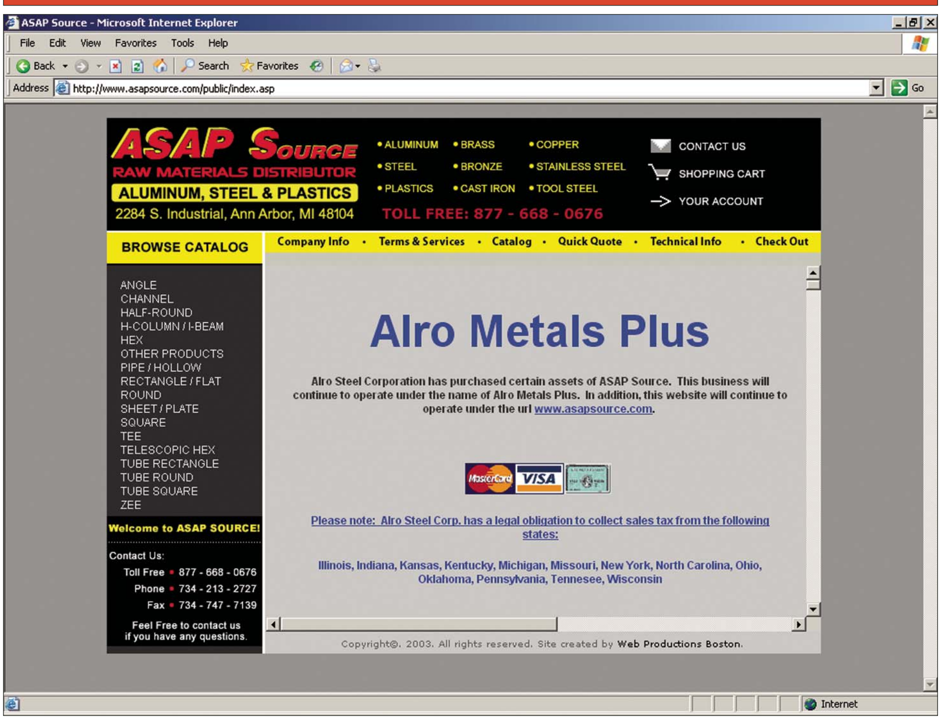
ASAP Source
www.asapsource.com

Acetal and acrylic plastic pipes, rods, sheets, and plate.

Aspects, Inc.
www.aspectsinc.com

Acrylic dome hemispheres of several different sizes. Intended to keep nasty squirrels from eating the birdfeed, but also useful for robots.

ASAP Source offers plastics and other building materials.



Cal Plastics and Metals
www.calplasticsandmetals.com

Plastics include ABS, acetal, acrylic, nylon, polycarbonate, and PVC.

GoldenWest Manufacturing
www.goldenwestmfg.com

Machineable plastic, foam board, and Butter-Board — a lightweight plastic block that is non-abrasive and very easy to machine or work with hand tools.

Laird Plastics
www.lairdplastics.com

Plastic sheets, rods, tubes, films, and related plastic products.

Multi-Craft Plastics, Inc.
www.multicraftplastics.com

Plastics of all kinds, including sheet, tube, rod, and extruded profiles, plus CNC and laser cutting. Retail stores in Portland and Eugene, OR.

McMaster-Carr Supply Company
www.mcmaster.com

Huge inventory of industrial supply, including various types, sizes, and colors of pre-cut ABS and PVC plastic sheets.

Plastic Products, Inc.
www.plastic-products.com

Sheets, rods, tubes, profiles, shapes, slabs, and "massive blocks," and more. Check the Steals and Deals page.

Plastruct, Inc.
www.plastruct.com

Leader in plastic scale model parts, sold through hobby stores. Of their product, their structural shapes, tubing, sheet, and patterned sheet materials are of keen interest to robot builders.

Public Missiles, Ltd.
www.publicmissiles.com

For robotics parts, look for their polymer tubing (strong but light), wrapped phenolic tubing, and two-part expanding foam.

R & J Sign Supply
www.rjsign.com

Of particular interest to us robot constructors is Scooterboard, a light-weight — yet strong — substrate that can be used to build machine bodies and other parts. Also sells Alumalite (aluminum over foam), corrugated plastic, and PVC foam board.

Ridout Plastics
www.ecomplastics.com

Industrial plastic distributor and fabricator. Products include acrylics, ABS, and PVC, in sheet, rod, and other forms. Located in San Diego, CA, but will ship.

Small Parts, Inc.
www.smallparts.com

In addition to small tools, gears, and other specialty components, Small Parts also offers a variety of raw plastic materials in rod, sheet, or tube form.

Specialty Resources Company
www.aliendecor.com

Clear plastic hemispheres of various sizes: five inches to 96 inches, with colors and hex patterns available.

TAP Plastics
www.tapplastics.com

Fiberglass, plastic, and signage products. Plastics retailer, with stores in Northern California, Oregon, and Washington.

Special Sources for Expanded PVC

You can purchase pre-cut sheets of rigid expanded PVC from the following specialty online retailers.

Budget Robotics
www.budgetrobotics.com

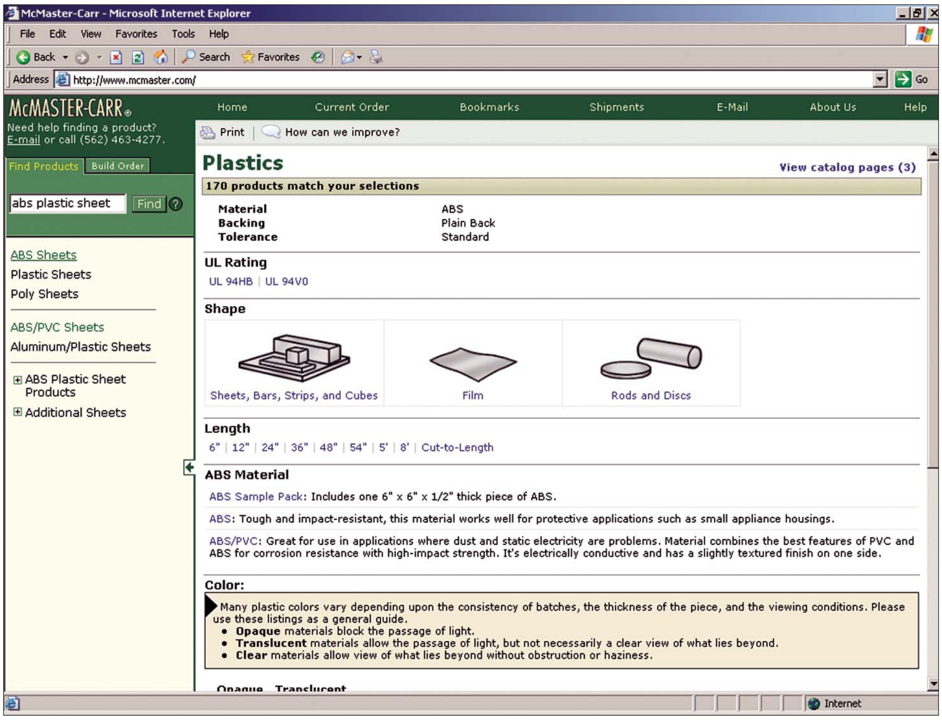
Budget Robotics has 12" x 12" rigid expanded PVC and acrylic sheets.

Lynxmotion
www.lynxmotion.com

8" x 12" rigid expanded PVC or polycarbonate (Lexan) plastic sheets.

Solarbotics
www.solarbotics.com

Features 12" x 8" rigid expanded PVC sheets. **SV**

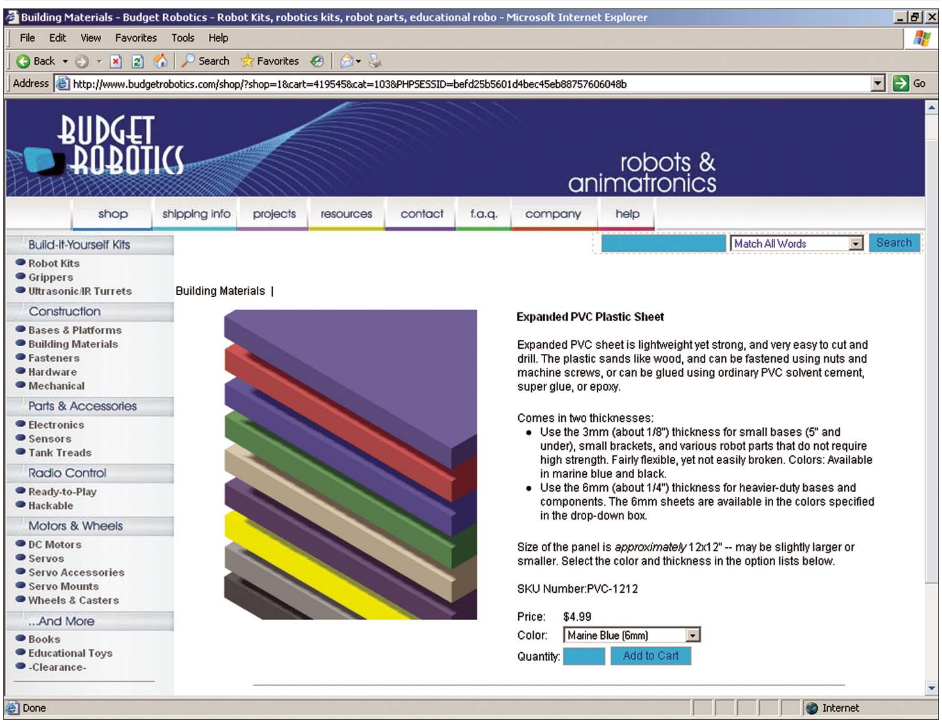


You can find these and other plastics products on McMaster-Carr Internet pages by using the site's search feature.

CONTACT THE AUTHOR

Gordon McComb can be reached via email at robots@robotoid.com.

Budget Robotics (my company) offers pre-cut expanded PVC in different thicknesses and colors.



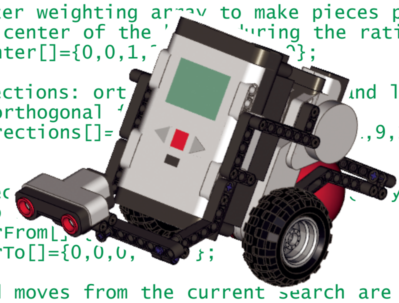
```
// castling bonuses
B8 castleRates[]={-40,-35,-30,0,5};

//center weighting ar  y to make pieces prefer
//the center of the    ring the rating routine
B8 center[]={0,0,1,                 };

//directions: ort                  and left/right
from orthogonal
B8 directions[]=                   ,9,10,-10,1,-
1};

//direc                           y really for
bishop
B8 dirFrom[]
B8 dirTo[]={0,0,0,                 };

//Good moves from the current search are stored in
this array
//so we can recognize them while searching and make
sure they are tested first
```



LESSONS FROM THE LABORATORY

A bi-monthly column for kids!

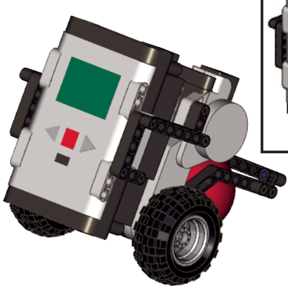
Sensors for the Runt

by James Isom

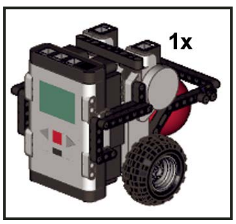
This month, we feature a few additional sensor attachments for the Runt chassis.

TOUCH SENSOR

STEP 1:

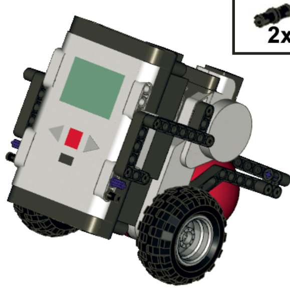


Parts:



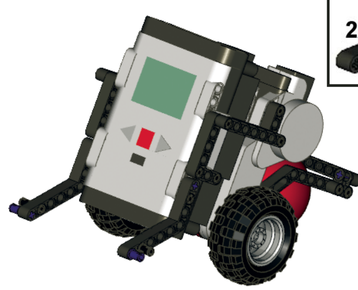
STEP 2:

Parts:



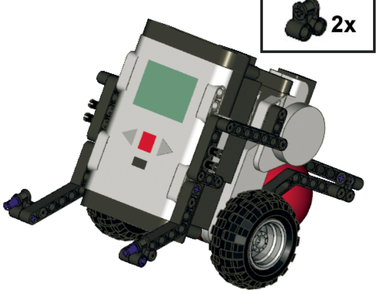
STEP 3:

Parts:



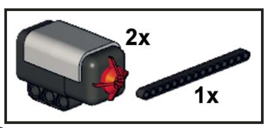
STEP 4:

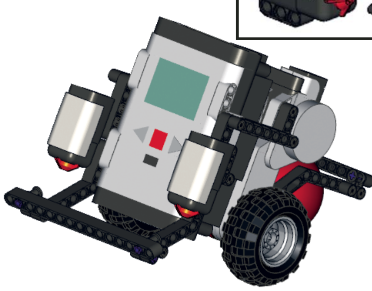
Parts:



STEP 5:

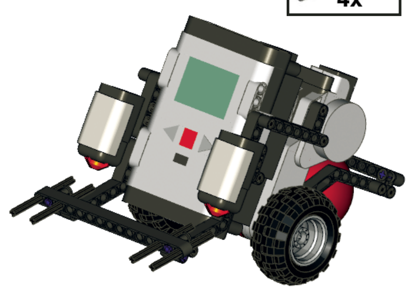
Parts:





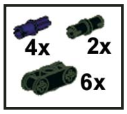
STEP 6:

Parts:



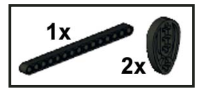
STEP 7:

Parts:



STEP 8:

Parts:



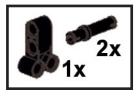
LIGHT SENSOR

STEP 1:

Parts:

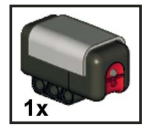
STEP 2:

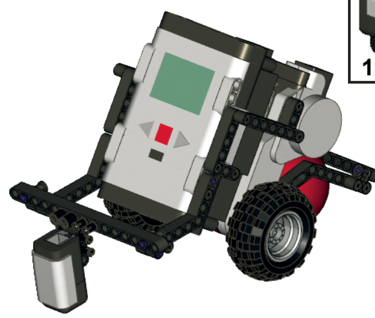
Parts:



STEP 3:

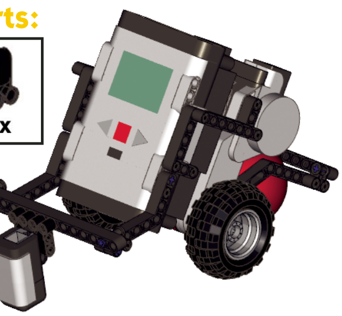
Parts:





STEP 4:

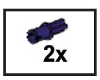
Parts:



ULTRASONIC SENSOR

STEP 1:

Parts:

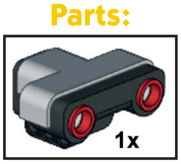


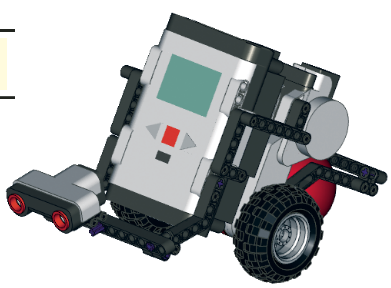
STEP 2:

Parts:

STEP 3:

Parts:

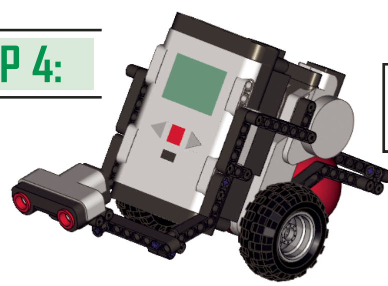




Sample programs for these sensor attachments will be available at **www.legoedwest.com** in the downloads area. **SV**

STEP 4:

Parts:



Mind Candy For Today's Roboticist

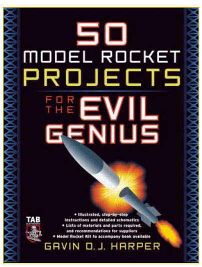
The *SERVO* Store

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

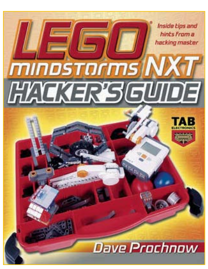
by Gavin D. J. Harper



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



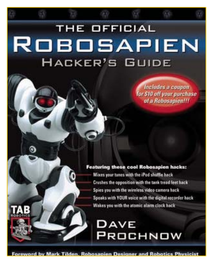
Here is an awesome next-generation collection of LEGO MINDSTORMS projects that enables you to build and program a real working robot in just 30 minutes! New technologies and expanded sensor capabilities make it easier than ever to add a level of sophistication to robotic and architectural creations.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

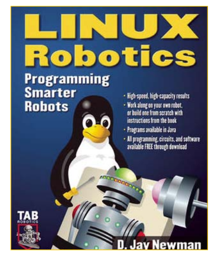
The fourth edition of this invention-inspiring engineering resource covers the past, present, and future of mechanisms and mechanical devices. You'll find drawings and descriptions of more than 2,000 components that have proven themselves over time and can be incorporated into the very latest mechanical, 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. **$89.95**

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. **$24.95 – Buy 2 or more at $19.95 each!**

Linux Robotics

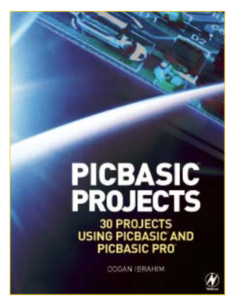
by D. Jay Newman



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



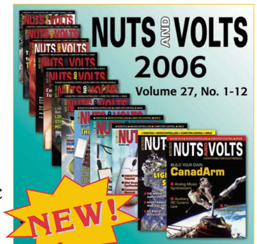
Covering the PIC BASIC and PIC BASIC PRO compilers, *PIC Basic Projects* provides an easy-to-use toolkit for developing applications with PIC BASIC. Numerous simple projects give clear and concrete examples of how PIC BASIC can be used to develop electronics applications, while larger and more advanced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

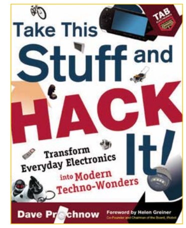
Intelligent Sensor Design Using the Microchip dsPIC

by Creed Huddleston

Unlike many embedded systems books that confine themselves strictly to firmware and software, this book also delves into the supporting electronic hardware, providing the reader with a complete understanding of the issues involved when interfacing to specific types of sensors and offering insight into the real-world problems designers will face. Meaningful software examples are implemented in both C and assembly language, and the source code is included on the accompanying CD. **$59.95**

Take This Stuff and Hack It!

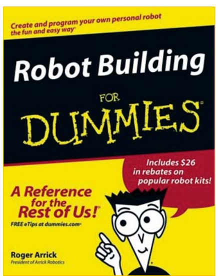
by Dave Prochnow



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**

Robot Building for Dummies

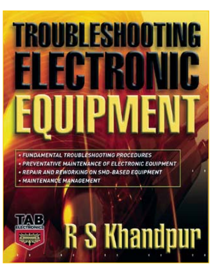
by Roger Arrick / Nancy Stevenson



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.

NEW! The SERVO store now offers DVDs!

Forbidden Planet: 50th Anniversary Edition

Director: Fred M. Wilcox
Run Time: 98 minutes

A starship crew goes to investigate the silence of a planet's colony only to find two survivors and a deadly secret that one of them has. When Adams and his crew are sent to investigate the silence from a planet inhabited by scientists, he finds all but two have died. Dr. Morbius and his daughter Altaira have somehow survived a hideous monster which roams the planet. Unknown to Adams, Morbius has made a discovery.

Actors: Walter Pidgeon, Anne Francis, Leslie Nielsen, Warren Stevens, Jack Kelly. **$26.95**

The Day the Earth Stood Still

Director: Robert Wise
Run Time: 92 minutes

An alien (Klaatu) with his mighty robot (Gort) land their spacecraft on Cold War-era Earth just after the end of World War II. They bring an important message to the planet that Klaatu wishes to tell to representatives of all nations. However, communication turns out to be difficult, so, after learning something about the natives, Klaatu decides on an alternative approach.

Actors: Michael Rennie, Patricia Neal, Hugh Marlowe, Sam Jaffe, Billy Gray. **$14.95**

From HomoSapien to RoboSapien

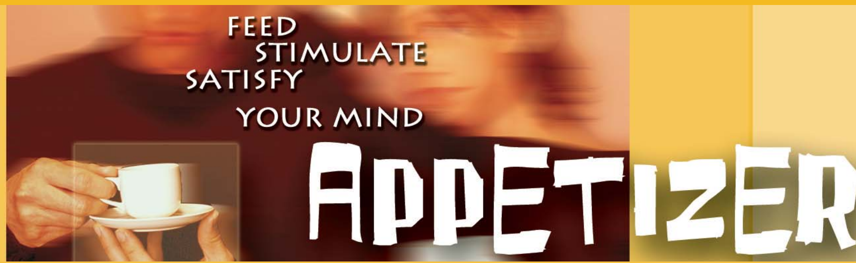
SUMMER SPECIAL $14.59!

SERVO Magazine T-Shirts

For men and women

Before R2D2 there was R1D1

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% preshrunk cotton.



by Paul Pawelski

The Cost of Mentorship

Last year, my wife and I volunteered at a regional competition for FIRST. My wife is a psychologist and is only mildly interested in what makes technology work. However, watching robots as big as she is going by and talking with the very excited students who were involved in making them convinced her that FIRST was a good thing and that we should get more involved.

An opportunity came up at work to talk with my division's head of engineering about sponsoring a team at a local high school. He liked the idea and asked me to make it happen. That is how my rookie year as a mentor also became my first year as the lead mentor for a team.

Our team started out at the beginning of the school year with secure funding, three teachers, around 20 students, and almost as many engineers willing to mentor.

The biggest problem we faced in the beginning was the school itself. Teaching is a hard job. A good teacher puts in around 60 hours a week during the school year. He or she has to put up with school boards that are scared by lawyers and insurance companies. Regulations and procedures which assume that students, mentors, and even teachers cannot be trusted are the norm. During the build season, nearly every day one of the teachers could be heard saying "no, it's not worth my job" to some idea that seemed logical to the student or mentor who suggested it. For mentors who come from an environment where risk proportional to potential gain is a way of life, a high school truly is a foreign environment.

The intent of FIRST is to make it appealing to more students than just the classic geeks. This means that the team has to compete for the students' time with other extracurricular activities such as football, soccer, swimming, wrestling, cross-country, jobs, church, parties, and dating. That was our second problem.

Our third problem was that mentors have lives, too. It is typical for an engineer to work 45 to 50 hours a week. Most of the mentors also had families who wanted to see them at least once in awhile.

Our fourth problem was that the engineering mentors thought of the students like the college interns they worked with in their jobs. However, four more years of maturity and having to actually pay for your education makes college interns much more focused and capable than typical high school students.

Despite all of this, we did get a bot to the regional competition. We finished 23rd out of 46 entries and we received the "Rookie Inspiration Award." Not bad for a first year.

Most articles about FIRST talk about the excitement, the energy, the feeling of accomplishment that everyone involved gets when their team's bot rolls out of its starting position in its first match. They talk about watching the light go on inside a student when they finally "get it" and the growth in the abilities of the students over the course of the season.

All of that does happen. However, it doesn't happen easily and it doesn't happen for free. FIRST is not about building a bot in six weeks, spending three days at a regional competition and, if you are lucky, three days at the finals. FIRST is about a year-round commitment to building students. It is another job. You will spend less time at home. Your personal projects will not get done. Your yard will no longer be the pride of the neighborhood. There are many things you will give up in order to mentor a team.

It Is Still Worth It

Last Thursday, our team had an awards reception. After the reception, the mother of one of the girls on the team came up to my wife and gave her a big hug. She told my wife how happy she was that someone actually cares about her daughter and how much her daughter has changed in the last six months because of my wife and the team.

That girl is a freshman. If she stays with it, she will have three more years with FIRST. Three more years of doing things she thought she couldn't do. Three more years traveling down the road from "drama queen" to confident, capable young woman. She is not the only one. To some extent every student, teacher, and mentor on the team has changed.

Dean Kamen says he wants to get FIRST into every high school. That will not happen. Places like Mobridge, SD or Ainsworth, NE, which have no significant industry, will not be able to find the mentors or the money. However, there are over a thousand schools currently which do have FIRST teams and there are thousands more in locations where there are the resources necessary to start teams. Thousands of students who come away knowing that they can build, they can design, they can create. They learn that there are more options in their lives than either winning American Idol or saying "would you like fries with that?"

Tomorrow, we have a planning meeting for the team. The mentors will be trying to convince the teachers that they need to allow team activities to occur over the summer and that they need to lighten up on their "positive control" attitudes. The teachers will try to convince the mentors that we spent way too much money and that more formal procedures are needed. The one thing we'll all agree on is we will be back next season. **SV**

Then and NOW

ROBOT COMMUNICATIONS

by Tom Carroll

Robot communications? You might be asking yourself, "What's this guy talking about? Is it how we control a robot? Is he implying verbal communications between two robots? Is it about how we communicate with robots? Is it how they talk to us? Could he mean how they might talk with our homes and computers?" Well, it's all this and more.

Last year, I wrote about how we can have our robots speak and listen to us in the August and September issues of *SERVO*, but there are other ways to communicate other than by the human voice. The first communications for remote control was Tesla's boat that used a coherer as a receiver back over a century ago. The Electrical Exhibition of 1898 at Madison Square Garden in New York featured his 'robot boat,' as he called it, controlled from shore from a 'command post.' It was a spectacle for those times, but was hardly what we might call reliable communications with a robotic device. Spark gap transmitters emitting RF energy from a few Hertz to light frequencies is a bit of overkill.

Robot Radio Control

In the 1950s and 1960s, to most people 'radio control' meant remotely controlling model airplanes. The earliest radio control systems were not proportional at all but were simple 27 MHz signals sent to a single channel receiver on board the aircraft. Figure 1 shows an early R/C single tube receiver from Aristo-Craft. There were no 'servos' of the type we know today but a simple rubber band-powered escapement that powered the rudder. Figure 2 shows the escapement with the hook for attaching a long, twisted rubber band.

It acted much like a stepping relay, but instead of stepping to different electrical contacts, the steps pushed the rudder to one side, then to the middle, then to the other side, and back to the middle, and so on each time the transmitter's button was pressed. This allowed the operator to control the flight path straight, left, or right but not in altitude. There were a few model cars out there at that time that also used a similar method, and many cars were hacked to make a robot of sorts. I managed to ruin a nice toy Jeep Christmas present years ago as a kid only to produce a 'robot' that looked stupid and acted rather poorly. The original Jeep was a lot better.

Later on, 'remote control' meant controlling your garage door opener, TV set, and a host of other things in this modern world, but model airplanes and cars still had the best R/C equipment available for the robot experimenter. Figure 3 shows a slightly newer dual joystick model airplane R/C system courtesy of the R/C Information Page. When TV remote control 'made the scene,' the hand-held controls used an ultrasonic carrier system to convey information to the TV set. Sounds within a home such as wind and other things caused interference so the standard was changed to an IR system that is still in use today.

Robot experimenters have made use of these hand-held remote controls as simple controllers for their robots. Using an IR transistor 'receiver' and an oscilloscope, people could read the codes represented by the different buttons and transferred them to the robot's MCU for control purposes.

Figure 1. Early R/C receiver.

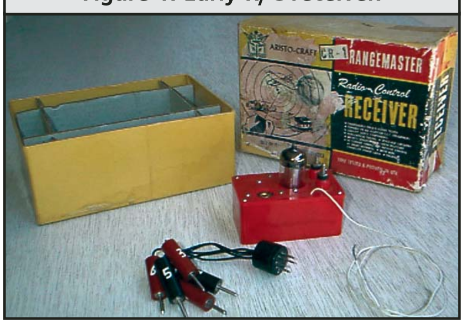


Figure 2. Citizen-Ship Escapement.

Figure 3. Dual joystick transmitter.

Radio Control and the Proportional Servo

Then came proportional R/C equipment and multi-channel radios, and at a proportionally higher cost. (Sorry, I just had to work that in there!) The two dual axis joystick transmitters became the standard for model aircraft and the hand-held trigger units with a 'steering wheel' control on the side for model cars. The first aircraft had a channel for the rudder control and another for elevators. Soon modelers wanted aileron control, trim tabs, engine speed control, landing gear and brake control and many other features so the channels available grew to eight or more. The poor guy on the ground with his cars still had only two or three channels, so the robot builders started using the aircraft systems to control the many functions on their robots. Trouble with the Federal Communications Commission (FCC) started brewing.

The FCC had set aside three original frequency bands for radio-control hobbyist use. The original 27 MHz AM frequency band was designated to be used for surface or air vehicles. This was the initial CB radio band and most of the first systems used the 27.255 MHz notch. Few serious hobbyists use this frequency band any more.

The 72 MHz FM frequency band was and is designated for aircraft only. It is not to be used for robots! The 75 MHz FM frequency band is designated for surface vehicles only, including robots. Fortunately, experimenters can now obtain multi-channel radio systems at 75 MHz, either by re-tuning 72 MHz systems, or buying them straight 'off the shelf' from dealers. Figure 4 shows a top-of-the-line Futaba $2,200 R/C system and Figure 5 is another high-end spread spectrum unit from Spektrum.

The 50.80 to 50.98 MHz band became available for both air and land but you must have a 'Ham' or amateur radio operator's license to use these frequencies. (Note: A Technician's Amateur Radio License could probably be earned by almost all readers of this magazine with maybe five to 10 minutes of study beforehand — there's no more Morse code requirement.)

The 2.4 GHz band recently became available for modeler's use and the spread spectrum technology obtainable at this 'microwave' frequency is great for noise free and reliable communications between a robot and a controller.

With any of these various RF systems, I cannot stress highly enough that you obey these FCC regulations. There have been many hobbyists, professional model makers, and robot operators who have not only had their equipment confiscated but have been heavily fined on top of that for illegally operating R/C equipment. Obey the rules. They are in effect for a good reason.

Figure 4. $2,200 14 channel PCM FM Futaba.

Figure 5. Spektrum DX7.

The R/C Servo — A Robot Experimenter's Dream

Stepping away from the 'communications' aspect of proportional R/C equipment for a moment, I'd like to address the PWM (pulse width modulation) facet of the R/C servo. We've been using PWM for years to drive our motors, but that is a bit different from the PWM that controls our servos. Motor driving PWM relies on the ratio of on to off of the pulses — the more on time for each of the train of pulses, the faster our motor turns. Our R/C servos also use a series of pulses — usually 50 to 60 per second, but it is the width of each pulse that determines the servo's position. The neutral center position of the servo requires a series of 1.5 millisecond pulses whereas 1.0 ms pulses will drive the servo all the way to the left and 2.0 ms pulses will drive it all the way to the right.

Robot experimenters quickly found out that if you removed the feedback pot from the servo's circuit and replaced it with a two resistor bridge or trim pot and then cut off the mechanical stops from the drive system, you could have a bi-directional robot drive motor controlled by a 1.0 to 2.0 ms pulse train. Figure 6 shows a typical servo before and after the pot's removal from the Seattle Robotics Society's website.

There are thousands of links on the Internet to assist you in hacking a servo for continuous rotation. Slight variations from 1.5 ms in either direction would result in faster or slower speeds. A simple 555 timer or microcontroller circuit can easily generate these pulse trains and a vast majority of first-time robot experimenters use servos hacked and connected in this manner.

Robot experimenters have a wide choice of R/C equipment to choose from these days. FM systems, spread spectrum, and pulse code modulation (PCM) are some of the

Figure 6. Hacking a servo.

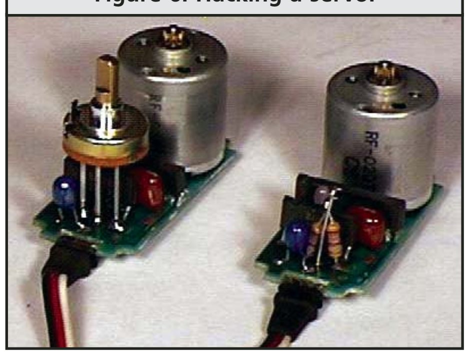


Figure 7. X-10 module.

later technologies with new ones coming out every year. Add-ons such as Vantec's Keykoder allow 12 extra on-off channels to occupy a single channel on the high end Futaba radio, thus expanding the features available for

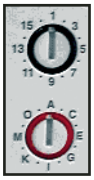


Figure 8. X-10 dials.

communications and control of a complex robot. High power motor drivers are available from Vantec and other manufacturers to take the pulse trains from an R/C receiver and control hundreds to over a thousand amps.

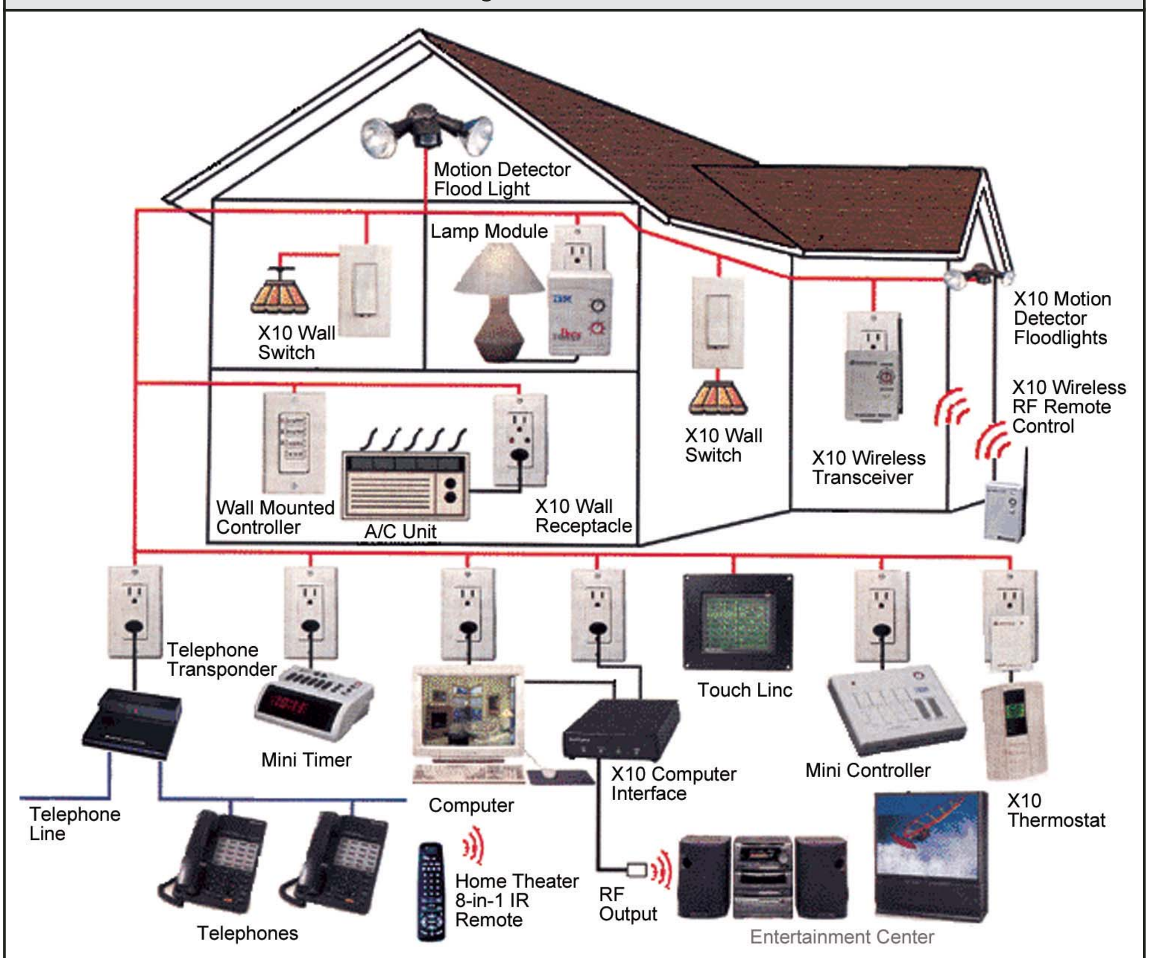
Radio control has always been present in experimental robotics, especially the combat robots and will continue to be; however, the purists still strive for a level of autonomy without a human in the loop.

Home Automation's Link to Robotics

Experimenters not only want to communicate with their robots, many want their robots to communicate with their homes. One of the best advantages of using home automation systems for experimentation with robots is the low cost of the systems. Just as the thousands of model airplane enthusiasts drove the cost of R/C systems down, so did the demand for home automation systems. Robot experimenters jumped on this technology like yellow jackets on a picnic table and devised many ways for two-way communications with their machines. One of the most popular home automation communications protocols is the X-10. We've seen them everywhere for many years but few know just how this system came about.

It was developed in Scotland as a way for hi-fi systems to communicate with different components. A group of people who wanted to develop advanced integrated circuits for the emerging pocket calculator market founded Pico Electronics Ltd. in Glenrothes, Scotland in the 1970s. Their various 'experiments' were labeled X-1, X-2, and so on for the special ICs that they developed for programmable calculators.

Figure 9. X-10 home.



BSR (British Sound Reproduction) makers of well-known record turntables asked Pico to develop an IC for a programmable record changer, 'X-9.' With this success, the also asked for an IC for remote control of all its equipment, or 'X-10.' The first of these devices were named BSR System X-10 and later, X-10 Powerhouse.

You've seen these interfaces in RadioShack and electronic and hardware stores that allow control of lighting and similar household appliances (see Figure 7). There are the duplex receptacle

Figure 10. Z-Wave HomeSeer starter kit.

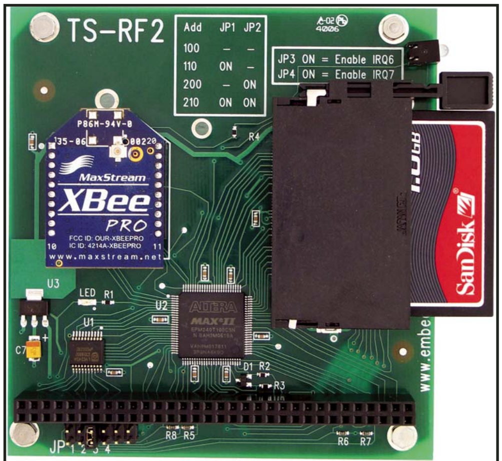


Figure 11. Zigbee radio module.

types that are exchanged with your standard sockets to allow control of lighting plugged into them. On each device are two 16 position rotary switches labeled 'unit' and 'house' that identify each of the units to the control panel or computer (see Figure 8). X-10 and the X-10 PRO transmits information over a house's power lines and the applications for robotics is an interconnection between a computer or home and the robot by a separate RF link to the X-10 controller (see Figure 9).

Binary data is transmitted by sending one ms bursts of 120 kHz data over the home's lines just past the zero crossing of the 60 Hz power sine wave. There is a starting sequence code (two cycles), followed by a house code (four cycles), followed by a key or function code (five cycles) that represents a number of 1 to 16 or on or off. Then an 11-bit code is sent for each command.

The FireCracker was developed in conjunction with the X-10 system as the RF link to and from handheld devices to a computer which, in turn, would control household devices and lighting.

Since your robot is not connected to the 120 VAC power lines — except maybe for charging — an RF link to any communications system is a must. It uses a simple serial interface and was quickly snapped up by robot experimenters as a way for their robot to talk with a computer and through this link, to household items connected to 120 VAC power. X-10 has not kept up with the advances as well as some of the newer home automation manufacturers, but it is still quite popular.

You can view them on the web at **X-10.com**, but I warn you, they will spam you to death with messages stating "this offer will end tonight. Buy now!" and similar spam.

Other Systems Compete with X-10

The Z-Wave standard — created and marketed by Zensys — which also makes the chips and wireless systems for Z-Wave-based 908.4 MHz communications, has brought on board over 150 companies to utilize this standard (see Figure 10). Z-Wave is an RF-based communications standard designed for residential and light commercial control applications. It has also found some popularity with robot systems, as it is strictly RF based rather than the line-carrier to RF standard of X-10.

Zensys advertises its technology: "Z-Wave transforms any stand-alone device into an intelligent networked device that can be controlled and monitored wirelessly. Z-Wave delivers high quality networking at a fraction of the cost of other similar technologies by focusing on narrow bandwidth applications and substituting costly hardware with innovative software solutions." Basic Z-Wave chips and fully functional modules are available for less that $5.

ZigBee is another fairly new communications standard created by the ZigBee Alliance — an industry group that also boasts 150 member companies. ZigBee is based on the IEEE 802.15.4 standard and is incompatible with the Z-Wave systems, and certainly incompatible with X-10 systems. ZigBee is said to be even less expensive for the experimenter or home automation application. Peter Best just had a great two-part article in *SERVO* (Feb-Mar) entitled "Low Powered Robot Communications" that used the IEEE 802.15.4-2003 standard that you should re-read. Figure 11 shows a ZigBee radio module from Technologic Systems that has an external Flash memory slot and uses an external antenna. Both of these systems have lower power consumption than Bluetooth, popular with cell phones; Wi-Fi, popular for laptop 'hot spots;' and wireless USB interconnections — features that are certainly important to small mobile robot devices. They also have the plug and play feature made popular by X-10. Their ease of use makes them a natural for implementation in robot communications.

Smart Labs' Insteon is another home automation protocol that became available in mid-2005 and is said to 'command 40 percent of the market' and is supposed to work with X-10 products. Like X-10, it is both RF and power line carrier communications. A basic Insteon starter kit includes two SigaLinc RFs (the RF links), a ControLinc V2 tabletop controller, and two LampLinc V2 dimmers for lights. SignaLinc RFs are available to improve INSTEON signal strength and network coverage throughout the home. As with all of these systems, there are a wide variety of add-ons to complete a home automation or security system, or a robot control link (after hacking).

These systems and standards mentioned are just the tip of the Iceberg as far as what is available to the robot experimenter. I've covered quite a bit on the older communications standards — especially the X-10 series that started the boom — but it appears in my studies that the ZigBee short distance 2.4 GHz wireless chip sets may be the way to go for robot experimenters.

Searching the Internet for wireless communications will unearth many other applicable sites and standards. If you want to learn more about some of the side technologies of robotics such as RF and wireless communications, I highly recommend our sister publication *Nuts & Volts* as a great source for 'everything for electronics.' The April issue not only had a great article entitled "Control Your World With an X-10 Interface," by Michael Simpson, but also had four ads for wireless and ZigBee wireless modules. *NV* still has a monthly robotics column from which this interest was born — *SERVO Magazine*.

If you need to communicate with your robot by any means such as IR, RF, voice, or whatever, there are resources available at reasonable prices to assist you. I hope this has encouraged you enough to dig deeper and find what is right for your robot to communicate with you, another robot, your household, and its computer or any type of sensor that you can imagine. **SV**

CONTACT THE AUTHOR

Tom Carroll can be reached via email at TWCarroll@aol.com.

Advertiser Index