ROBOTICS EDUCATION • ASK MR ROBOTO • GOBOT ROVER

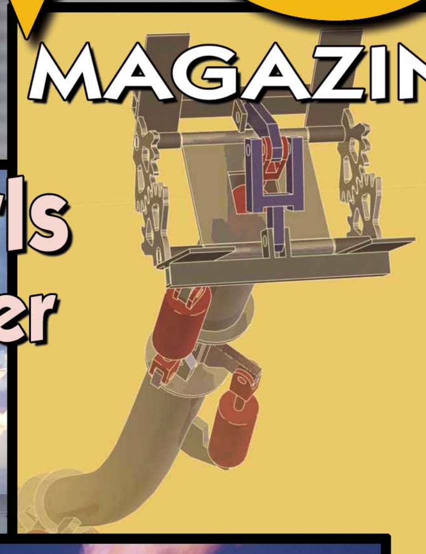
SERVO MAGAZINE

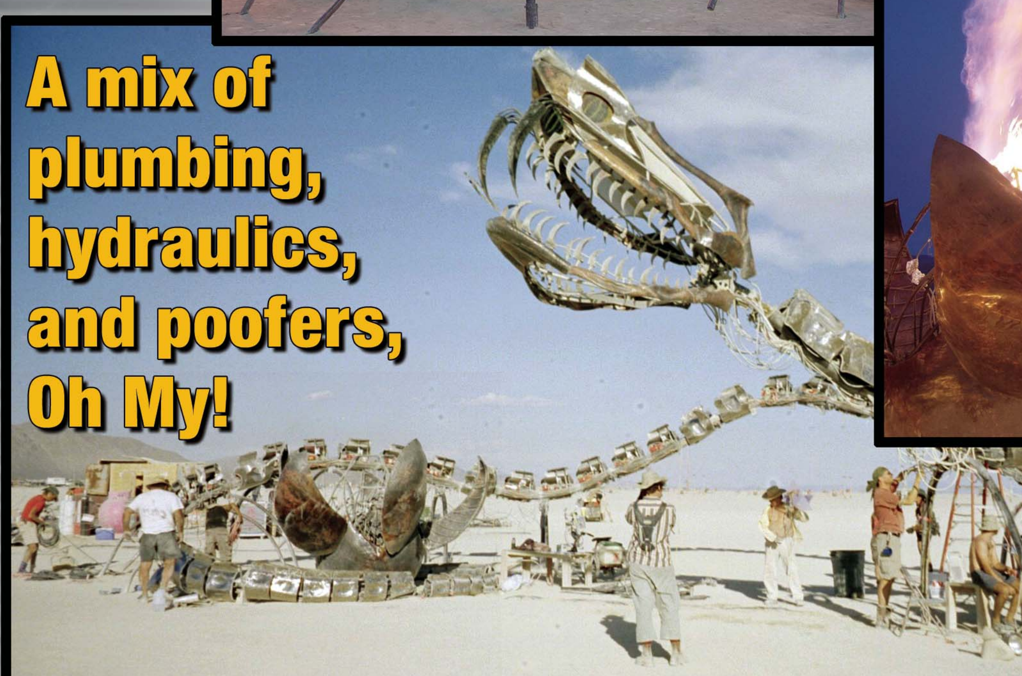
FOR THE ROBOT EXPERIMENTER

www.servomagazine.com

FEBRUARY 2007

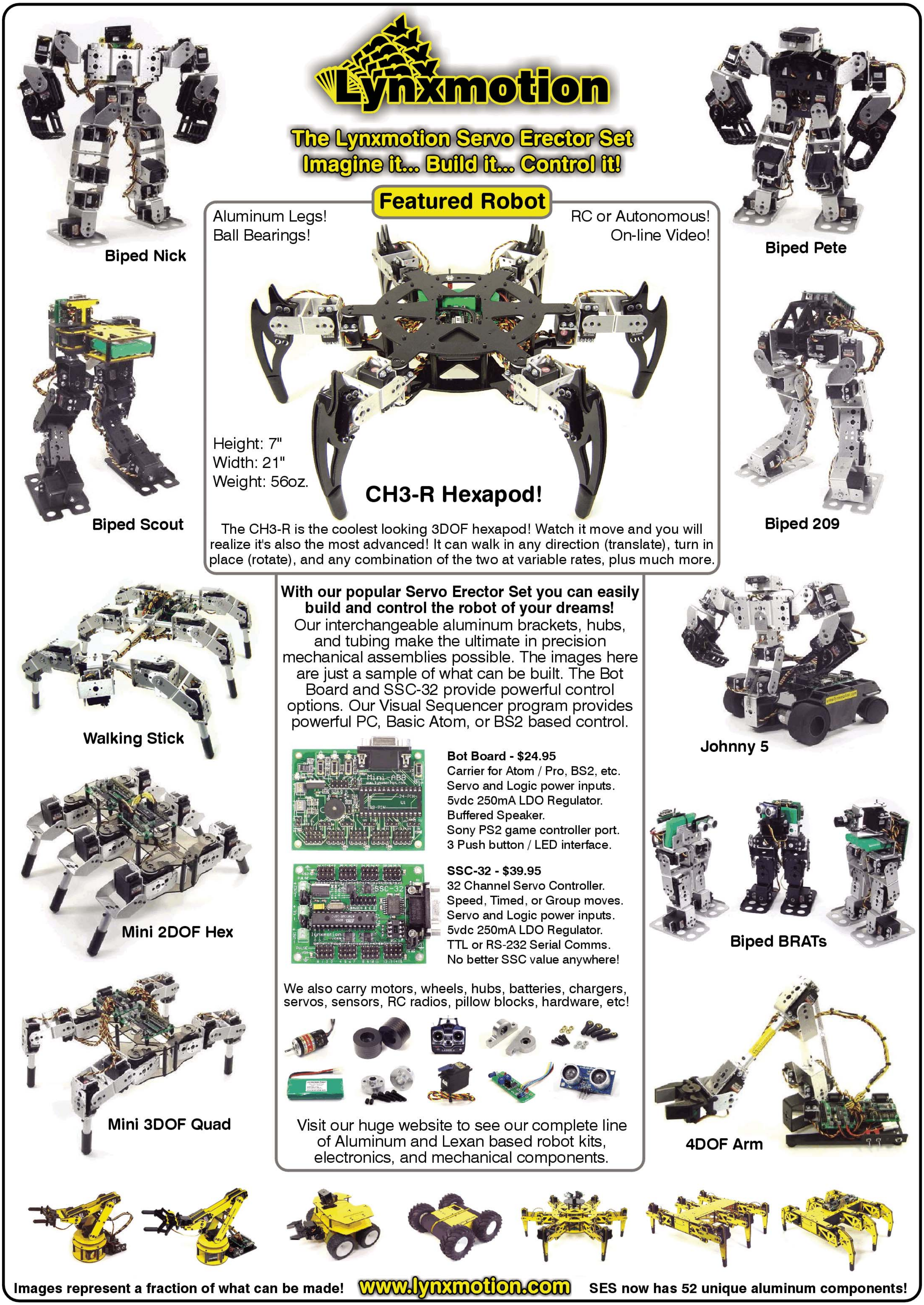
The Flaming Lotus Girls and The Serpent Mother





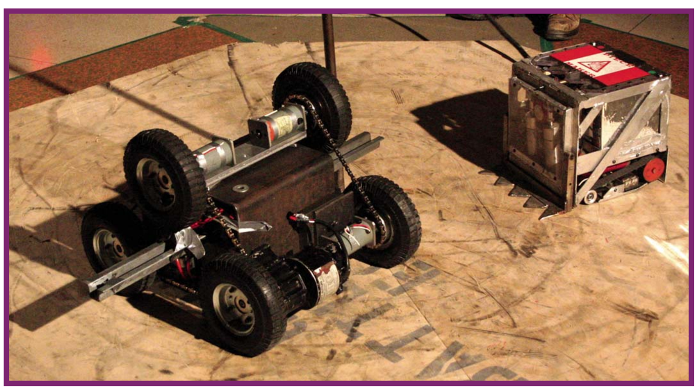
A mix of plumbing, hydraulics, and poofers, Oh My!

- Robot Simulation: AI Behaviors
- Low Power Robot Communications
- The Unmanned Little Bird Project


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

SERVO MAGAZINE

Features & Projects



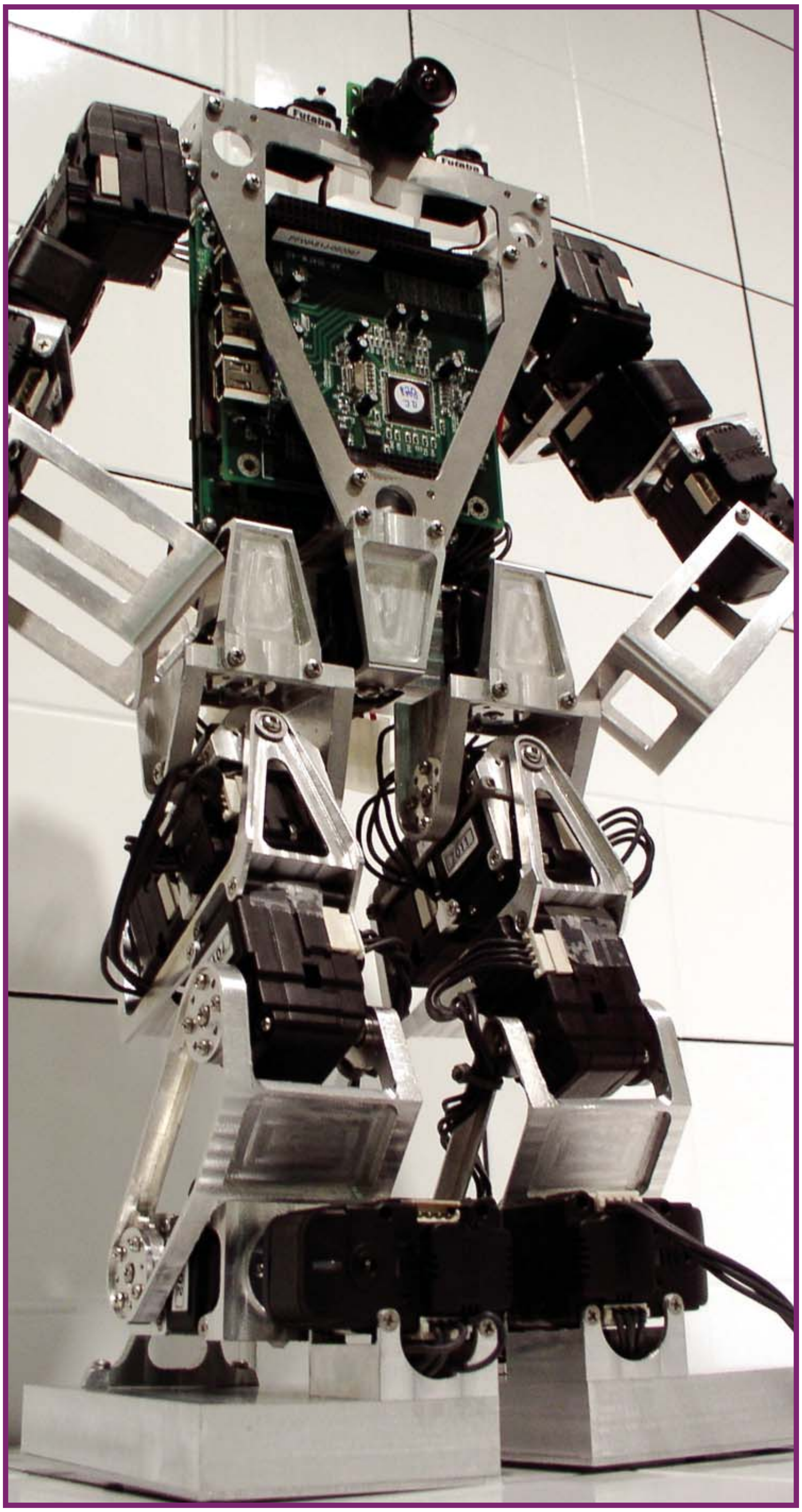
SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine,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

02.2007
VOL. 5 NO. 2

Columns

Departments

DARwIn 2.0 Page 35



See You at RoboCup 2007!

Mind / Iron

by Tom Carroll

Mind and Iron. Hmmm. What do these words mean to the average person? What do they mean to us who experiment with robots? I see them as a simplistic way of describing our way of tackling any technological project. We take our minds and develop something that is tangible but not necessarily made of iron. Some of you may remember the TV series and the movie entitled "Iron Giant" in which a boy makes friends with a giant alien robot that the government wants to destroy; a typical plot line for all 'B' movies. Well, rest assured that few — if any — robots are made of iron.

Outside the combat robot arena, few robots are even made of steel or similar. That is not the point here. Robots are envisioned as strong and invincible, just as we envision iron. This magazine is not about making powerful robots but rather introducing people to the fascinating science of robotics and assisting those who have been into robotics for a while. We want to put our minds to use to create something that is useful and enjoyable that we can see and touch. Robots combine so many fields of engineering and science that their creation can satisfy those who enjoy the mechanical aspects, computer science, sensors, vision, speech, electronics, RF technology, and many more fields.

In my *SERVO* column "Then and Now," I write about aspects of robotics that existed in the past and compare them to what is available today. Those of us who were first enthralled by our computers that could only take our inputs from a keyboard, process them, and present them upon a screen or printer are now overjoyed to see our inputs applied to motion. Simple programs and routines stored in our machines can cause them to have "minds of their own" as they roam about at will. We are so lucky these days to have inexpensive technology available to us for our robot projects that would have cost thousands of dollars just two decades ago. Cheap $1 microcontrollers can serve as the 'mind' of our 'iron' friends. Surplus gearmotors drive our machines. Sophisticated, yet affordable, vision sensors give our machines the ability to see intelligently. They can talk and listen with today's inexpensive speech recognition and synthesis boards.

I am frequently asked just how does one "get into" robotics? I usually answer back, "just what do you want to do with a robot or learn about?" I always recommend that people go to the Internet or a library and read about the subject. In that way, they can narrow down just what it is that interests them the most. Some may want to explore underwater, remotely-operated or autonomous vehicles. Others look with interest at the many types of combat, sumo, and maze robot contests that are in existence and envision their robot winning competitions. Others just want a robot platform upon which they can experiment with different types of appendages or sensor suites. Still other people just want to build a robot that does something that no other machine can do. Reading and studying about robots and their capabilities is the best way to delve into this new science.

Another entry method to experimental robotics is to find others who have the same interest. Next month's "Then and Now" column covers the subject of robotics organizations from a historical aspect. Get on the Internet and find a robotics group near you. If you're lucky enough to locate a more established group, you'll find members who have undoubtedly run across the same problem or have the answers to your many questions. People love to share their knowledge, especially those in experimental robotics. I have been involved

Mind/Iron Continued

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

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

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

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

CONTRIBUTING EDITORS
Jeff Eckert, Tom Carroll, Gordon McComb, David Geer, Pete Miles, Kevin Berry, Chris Harriman, Bryan Bergeron, Karl Muecke, Patrick Cox, Dennis Hong, R. Steven Rainwater, Paul Pawelski, Robin Hewitt, Steve Nelson, Jim Miller, Peter Best, Simone Jones, Martin Koch, James Isom

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

BIO-FEEDBACK

Dear SERVO:

In the Jan 07 issue, Pete Miles gave an excellent, detailed answer to the question about the differences between wired and wireless Playstation gamepads. This is one of the reasons why your magazine is so useful from cover to cover. Thanks again, and in the spirit of giving something back, I offer some more details on the subject.

Some PS gamepads (including the PSone and Dual Shock 2) use a protocol in which both sides write a bit while the clock is low and then read the bit from the other side after the clock goes high. With such gamepads, the time between the clock low and high signals must be at least 6 µsec. Here is the assembler code for the critical timing section that can be used on a PIC for such gamepads:

```
bcf   PS2_CLOCK              ; clear CLOCK
btfsc  rwByteCmd, 0          ; send CMD bit
bsf   PS2_CMD
btfss  rwByteCmd, 0
bcf   PS2_CMD
op                           ; wait 2 µsec @ 4 MHz
nop
bsf   PS2_CLOCK              ; set CLOCK
movlw 0x80                   ; read DATA
btfsc        PS2_DATA
iorwf        rwByteData, f
```

Other PS gamepads (including most wireless models) use a protocol in which both sides write a bit before the clock is low and then read the bit from the other side after the clock goes low and before the clock goes high. With such gamepads, the time between the clock low and high signals must be no more than 5 µsec. Otherwise, the gamepad will time out. Here's the PIC code required for these gamepads:

```
btfsc  rwByteCmd, 0          ; send CMD
bsf   PS2_CMD
btfss  rwByteCmd, 0
bcf   PS2_CMD
bcf   PS2_CLOCK              ; clear CLOCK
movlw 0x80                   ; read DATA
btfsc        PS2_DATA
iorwf        rwByteData, f
   nop                       ; wait 1 µsec @ 4 MHz
bsf   PS2_CLOCK              ; set CLOCK
```

The timing change is the primary reason why Basic programs running on slower PICs can't handle the newer wireless gamepads. This is a good example of where a little bit of old assembler code comes to the rescue.

Frank Pittelli, Ph.D.
CheapControlSystems.com

Dear SERVO:

Compliments to Dave Calkins on his fine article about the Trinity College Home Robot Fire Fighting Contest. In my opinion, this is the best all-around robot contest, encompassing different levels of skill, rules that evolve yet remain consistent from year-to-year, and an associated symposium. I would like to correct one error in the article. The Firefighting contest was the brain child of Jake Mendelssohn. Also, the first contest was held at the Science Center of Connecticut in West Hartford before moving to Trinity College in 1995. Jake not only originated the contest but was its master of ceremonies for the first 10 years. I know Jake and Dr. Ahlgren personally and their enthusiasm and hard work is what makes this contest successful. Congratulations to them both.

John Piccirillo, Ph.D.
University of Alabama in Huntsville

with robotics more years than I can remember, but I will always find someone at my own Seattle Robotics Society meetings that has a lot more knowledge than me about a specific subject. Another good thing about group meetings is swap meets where members exchange un-needed robot parts with others.

Once you know what aspect of robotics you are most interested in and have met a group of fellow roboticists, you should expand your personal library with a few good reference books, such as Gordon McComb's *Robot Builder's Sourcebook* and others you'll find here in *SERVO*. You may want to go the route of Parallax's Boe-Bot series to learn PIC programming or even the LEGO Mindstorms or VEX robot kits. For those who want to "cut their own metal" and bypass kits, there are many books available to assist you, such as a book I co-wrote a few years back with fellow *SERVO* columnist, Pete Miles, entitled *Build Your Own Combat Robot*, but there are a lot of newer titles also available that will aid you in your own designs.

In many books, you might find extensive lists of required tools that the author has compiled that he or she feels is necessary to build robots. Quite frankly, some of the best robots that I have seen were built entirely with basic hand tools. Don't go out and buy a bunch of tools without really understanding just why it is you need them. If you can't use a hand hacksaw or saber saw to cut a piece of metal, go to a friend who has a shear to cut it. The idea is to build something, not to have the best workshop in town.

There is no better feeling than when you first hit the power switch on your new creation and see it come to life. (Well, actually standing by your spouse and holding your new born child comes first, but seeing your "iron" "do its thing" is a great feeling.) Actually, one of mine came to life and just as quickly dove off the workbench and killed itself, but that was my stupidity.

The bottom line is: Just do it! Whether you are a new reader of *SERVO* and are deciding on just what to build or a long-time robot builder who is mulling over a new robot project idea, the only thing stopping you is YOU. It will not matter at all what your creation looks like or what it will or won't do. You cannot learn and improve if you don't put your "mind to the iron." SV

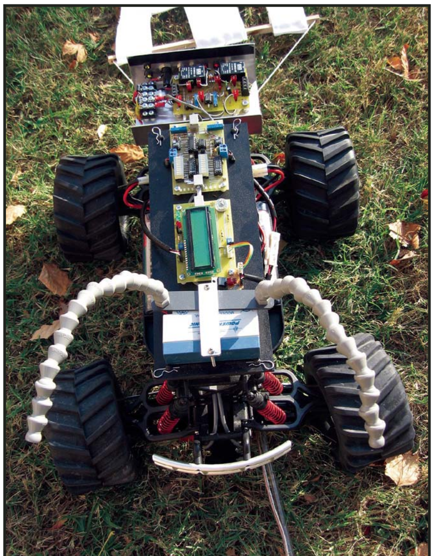
Robytes

by Jeff Eckert

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

— Jeff Eckert

Sticking It to Ticks



The Tick Rover automatically wipes out ticks and other outdoor pests. Photo courtesy of VMI.

If you are experiencing a fever and headache, feel lethargic, and have a stiff neck and muscle pains, it could just be a reaction to that late-night snack at Taco Bell. But if you also have detected a red lesion up to three inches in diameter somewhere on your skin, you quite possibly could have Lyme disease — acquired via the bite of a tick. The nasty little creatures can also give you Rocky Mountain spotted fever, tick typhus, and other diseases. Fortunately, it turns out that ticks usually inhabit only a 15-foot wide boundary between cultivated lawns and woods (the "ecotone"), so the solution is to wipe them out in that area. Based on that approach, students and faculty of the Virginia Military Institute (**www.vmi.edu**) and Old Dominion University (**www.odu.edu**) came up with a way to protect your turf: the Robot Sentinel (a.k.a., Tick Rover) robotic tick killing system.

To make the thing work, you route a flexible, perforated tube around the ecotone. The tube emits a chemoattractant (e.g., CO_2), which draws ticks into the tube's path. Then the wheeled robot follows the tube while collecting and exposing ticks to permethrin (a common insecticide). Inside the tube is a signal wire that the bot follows using inductive sensors. After every lap, it returns to a shed to be recharged, cleaned, and UV sterilized.

Apparently, after three months of treatment, the ticks' life cycle will be broken, and the area will be free of them for years. The patent-pending machine can be adapted to kill off termites, cockroaches, aphids, and others. The co-advisers on the project were James Squire and David Livingston of VMI and Daniel Sonenshine from Old Dominion. For more photos and videos, visit **http://academics.vmi.edu/ee_js/Research/Tick_Rover/Field_Test2/Tick_Rover_2.htm**.

Bots Harmful to Mental Health?

If not for a clinical psychologist from the Massachusetts Institute of Technology (MIT; **www.mit.edu**), I never would have suspected it. But Sherry Turkle, the Abby Rockefeller Mauzé Professor of the Social Studies of Science and Technology and director of the MIT Initiative on Technology and Self has serious concerns about "the implications of increasingly personal interactions between robots and humans," and acknowledges that some of her research has given her "the chills," and noted that she is "struggling to find an open voice." (Apparently, she is winning that struggle, having delivered the bad news at a lecture on "What Questions Do 'Sociable Robots' Pose for Science, Technology, and Society?")

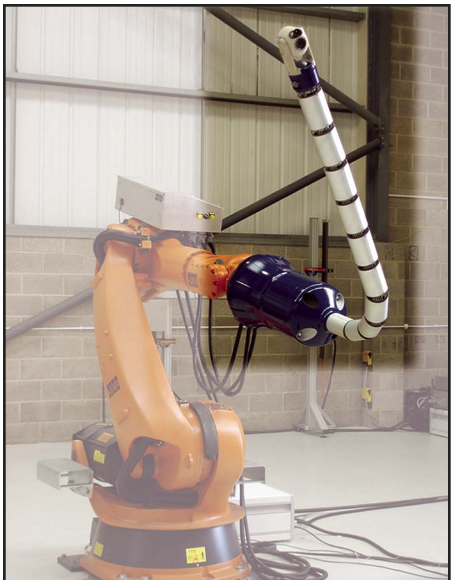
You see, Turkle is looking at bots not as machines but as "evocative objects" and "relational artifacts." The chilly thing is that children and adults seem to be forming bonds with Furbies, Aibos, and other robo pets to the point at which we are taking care of them rather than vice versa. Turkle was alarmed, for example, to discover that a local nursing home had bought 25 "My Real Baby" dolls for the residents because of their soothing effect. The soothing response was based on a sham, she believes, asking "What can something that does not have a life cycle know about your death, or about your pain?"

Oooooookay ...

Bad dog (left) and good dog (right). Good dog by Chance Agrella, courtesy of freerangestock.com.

Snakebot Designed for Airbus Work

Back in the industrial world, OC Robotics (**www.ocrobotics.com**) has been around since 1997, originally providing servo mechanisms for HP and Rolls-Royce. The company now focuses on snake-arm robots, and it was approached by Airbus UK and KUKA Robot Group to help develop a tool that can operate inside rib bays and other confined aircraft



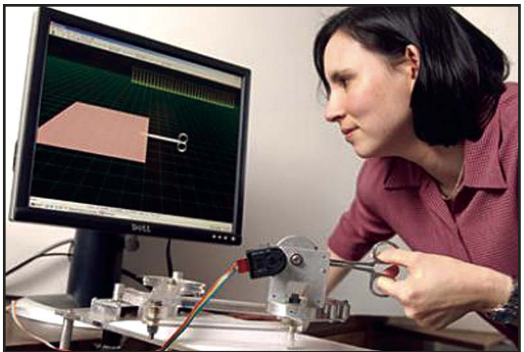
Attached to a KUKA industrial robot, this snake arm is designed for assembly and inspection tasks within aircraft wings. Photo courtesy of OC Robotics.

structures. The result is the snake arm shown here, which uses the KUKA unit as a delivery tool.

The robot is equipped with a wrist and tool interface to allow attachment of a variety of different tools designed by OC Robotics. Initial tests show the arm is flexible enough to deliver the required tools to areas of the wing box that were previously inaccessible to automation, to perform tasks such as inspection, final sealant application, and swagging. In the future, the OC Robotics Extender product family will be adaptable to other industrial robot models.

But Good for Physical Health

Everything is a trade-off, they say, and as we substitute robot hands for human ones in surgical procedures, we gain precision but lose the sense of touch. Well, maybe not, given the efforts of mechanical engineer Allison Okamura, who is a participant in the National Science Foundation (NSF) Engineering Research Center for Computer-Integrated Surgical Systems and Technology, based at Johns Hopkins University (**www.jhu.edu**). With funding from the National Institutes of Health and the NSF, she has established a collaboration with Intuitive Surgical Inc., maker of the da Vinci robotic system widely used for heart and prostate operations.



Allison Okamura demonstrates her lab's scissor-based surgical simulator. Photo by Will Kirk, courtesy of Johns Hopkins University.

"The surgeons have asked for this kind of feedback," says Okamura. "So we're using our understanding of haptic technology to try to give surgeons back the sense of touch that they lose when they use robotic medical tools." For example, the da Vinci system can tie sutures, but the operator gets no feedback as to how hard the thread is being pulled, which can result in breaks. The researchers want the human operator to be able to feel some resistance to sense when too much force is applied. At this point, they have not even established the optimal method to achieve that.

One approach would be to attach force sensors to the robotic tools that would convey how much force is being applied. Another is to create mathematical computer models that represent the tool's movements and then send feedback to the operator's hands.

In the meantime, the team has developed an interim system that — in the case of the suture — uses a colored circle to follow an image of the robotic tool on a display, indicating how much force is being applied. The operator will be cued with a red light (too much force), a yellow light (caution), and a green one (right amount). Research continues. SV

GEERHEAD

by David Geer

Contact the author at geercom@alltel.net

The Unmanned Little Bird Project

Optionally Manned Helicopter Ramps Up Unmanned Air Vehicle (UAV) Development and Testing

Those familiar with Little Bird may know it as a small, two-person helicopter. This is the very reason it was selected as the platform for the unmanned — actually optionally-manned — helicopter project known as Unmanned Little Bird or the ULB.

Because all testing can be done with a human operator on board, test duration is increased and test location can be anywhere, speeding the development process.

Due to concerns around the safety of the Little Bird hardware investment and without a human operator on board, researchers would test one functionality such as hovering and then land the vehicle. They could then work with the data collected before risking the hardware again.

With a human operator on board who could take over the controls at any time, the risk to the hardware was greatly decreased and researchers could safely test many kinds of functionality in one flight before going back to the lab to make use of the data.

UAVs must test in unpopulated areas like the desert to avoid the risk of the uncertain technology falling on people. With a human operator available in case of incident, tests can be conducted anywhere, saving the time and costs associated with acquiring and getting to a qualifying test location.

As a result of these factors, testing that would have taken weeks took days. Because researchers have developed a standardized means of deploying optionally manned capabilities to manned vehicles, they can easily be adapted to any other manned vehicle for fast development of unmanned capabilities.

Little Bird up close on the ground before media and other invited guests.

Little Bird, Big Robot

Autonomy is the primary robotic characteristic of the ULB. Boeing researchers are pushing autonomous unmanned vehicle control to higher levels with each iteration of the helicopter.

"A lot of current unmanned vehicles have pilots on the ground that fly the aircraft. It's like an elaborate radio control configuration. These guys are still doing the stick and throttle, even for a couple of the helicopter UAVs, as well," says Dino Cerchie, program manager for the ULB Demonstrator and A/MH-6X Little Bird programs, Advanced Rotorcraft Systems, Boeing.

Cerchie's team started with a manned vehicle and made it optionally manned, using a combination flight mission planning/waypoint control methodology to guide the aircraft. "It's a little bit next-generation as far as autonomous behavior," adds Cerchie. While you don't have to have someone fly the ULB, you can have a human operator for testing or other purposes and it is big enough to carry human and other cargo.

Specs, Support Systems, and Hardware

The ULB has flown as long as three uninterrupted hours at a speed of up to 125 knots. The Rolls-Royce engine — a turbine aviation engine — uses aviation fuel, all standard, to accomplish this.

The ULB has a 27.5 foot diameter rotor, is nine feet tall, seven feet wide, and a little over 20 feet long.

Primary support systems include the flight control computer on board the craft and any ground systems and computers that communicate with it. A ground station operator monitors the aircraft and can modify the flight path of the vehicle, changing the direction of the sensors, as well.

The flight control system uses an Embedded GPS/INS Navigation System (EGI) for state position data and an air data computer for speed and altitude data. The flight control computer uses the combined information to command the actuators that fly the craft.

One of the ULB's eyes is an MX15EORI web cam sensor — a high-end camera. While testing the vehicle, researchers have applied a variety of sensors for various purposes. "We look at [different ways of effectively] putting an eye in the sky," says Cerchie.

All ULB software on the ground and in the air is custom programmed; Boeing developed proprietary code written in C. Some software monitors and uplinks commands to the ULB and other software controls input/output data (I/O) to the craft. Still other software controls the rules that guide aircraft flight and other activity.

Under development are systems and software for the improvement of Intelligence, Surveillance, and Reconnaissance (ISR) missions. The craft is used to relay communications for ground-to-air communications signals or air-to-air signals. At a point where the signal may begin to drop off before it could get to its final destination, the craft could intercept and resend the signal on to its intended recipient.

Communications protocols include the Ethernet interface on the ground facing side and on the aircraft. The craft also uses MIL-STD-5053 for data transfer around the various equipment on the craft that need to know what's happening.

The craft is also being developed for autonomous carriage of payloads such as supplies to various locations so they can be delivered without risk to a human operator.

The craft's engines include generic, commercial, off-the-shelf actuators that are developed further in-house and, of course, the Rolls Royce 250-C30 engine for flight.

Demonstration Objectives

The ULB demonstration objectives include target identification, precision re-supply, and communication relay as previously mentioned.

In target identification demonstration missions, the ULB receives input from a ground-based system through the ground station about a potential target. When this data is transferred to the craft, it automatically slues its camera toward the target position.

ULB then flies to the position, zooms in, makes a positive ID of the target, and collects better coordinates on its location, which it sends to other ground- or air-based systems. This method cuts 20-30 minutes off the time it takes to identify targets.

In precision re-supply, the ULB carries a payload to a particular location. Because it will be able to do this unmanned in actual practice (as it does in testing), lives will be saved, as no human operator will be required on board.

In communication relay, the ULB acts as a relay point in a larger wireless network. "The helicopter itself," explains Cerchie, "is just to get a capability at a given location. Unique to a helicopter is that it doesn't have to move to be airborne. It can be fixed in space. We are effectively a low flying satellite or low flying cell phone network tower."

As such, it can enable very high bandwidth communications with streaming video without interruptions in communications or the network that might otherwise arise due to distance or location issues.

In the military, most equipment and systems are integrated with other systems. Integration is therefore a top priority. Airborne and ground systems are integrated so that the ULB can send and receive information from other types of air and ground-based systems and platforms. The ULB project works on the kinds of interface technologies that make it possible to move that information around faster.

A very nice ULB angle shot, rotors turning more visibly.

THE 50-FOOT DROP

"When we first started developing and flight-testing the aircraft," says Dino Cerchie, program manager for the ULB Demonstrator, "since it is [flown] manned or unmanned, we could start the flight in manned mode and transition it to unmanned mode. We started doing that. We started playing [with ULB's] waypoint course. We had one waypoint course that left the plant and one that returned to the plant.

We never combined the two until one flight where the test pilot and I were on the craft checking to see if this was going to work. We figured out very quickly [that] there was a 50-foot difference [drop] between the outbound waypoint course and the inbound waypoint course. We were sitting out there hovering at about 500 feet and the aircraft just plummeted about 50 feet and caught itself in a hover and came home." At no time during this test scenario did the men take over the controls.

SEARCH AND RESCUE

Today's deployed UAVs can perform the search function of search and rescue. Yet, these tiny craft are too small for the formidable task of air-lifting soldiers out of harm's way. A larger and usually manned craft must enter the danger zone to get them out.

Each ULB can carry at least two passengers without requiring that a human operator also be put at risk. It can thus perform search and rescue. The soldiers rescued would not need any flight training in order to be saved.

Little Bird demonstrator, a pre-cursor to the current model.

A large view of the ULB in-flight.

Advances

The current ULB configuration is the A/MH-6X, the next generation after the ULB demonstrator. Whereas the first craft was a proof-of-concept with a single control channel, the "6X" has more payload capacity, more range, and comes with controls that are more redundant.

When asked, ULB researchers stated that they are, in fact, working on various methods for avoiding detection for future models. In the larger picture, every aircraft program has a growth path. "If it didn't, I'd be out of a job," quips Cerchie.

The team definitely has many other improvements in mind. For example, they are working with Rolls-Royce on a new main rotor system with a higher performance engine and on extending the fuel capability to lengthen the vehicle's range and payload capacity.

Most importantly, there is the autonomy. "What we have done is laid in the basic core control of the aircraft," says Cerchie. "A lot of the autonomous behavior you keep hearing about, as that develops in the next decade or so, those kinds of features can be added on to the core control so that software actually starts talking to the core controller software. We've developed it so that it allows growth as autonomous behavior ability grows." So, new autonomous features will be scalable and standardized to the existing system.

Having a Ball!

According to Cerchie, the people working on ULB are enjoying themselves. The design approach they took from the beginning put them on a path where they can have a safety pilot in the vehicle, allowing them to take more risks throughout the development process. Because the pilot can take over at any time if there is an autonomous system error, the craft can be held intact for future testing, avoiding replacement costs.

"This has allowed us to progress at a much more rapid pace than other UAV programs," says Cerchie; "even though this program is only a couple years old, we are abreast if not ahead of most other UAV programs and that [standing] is only going to [improve] with time."

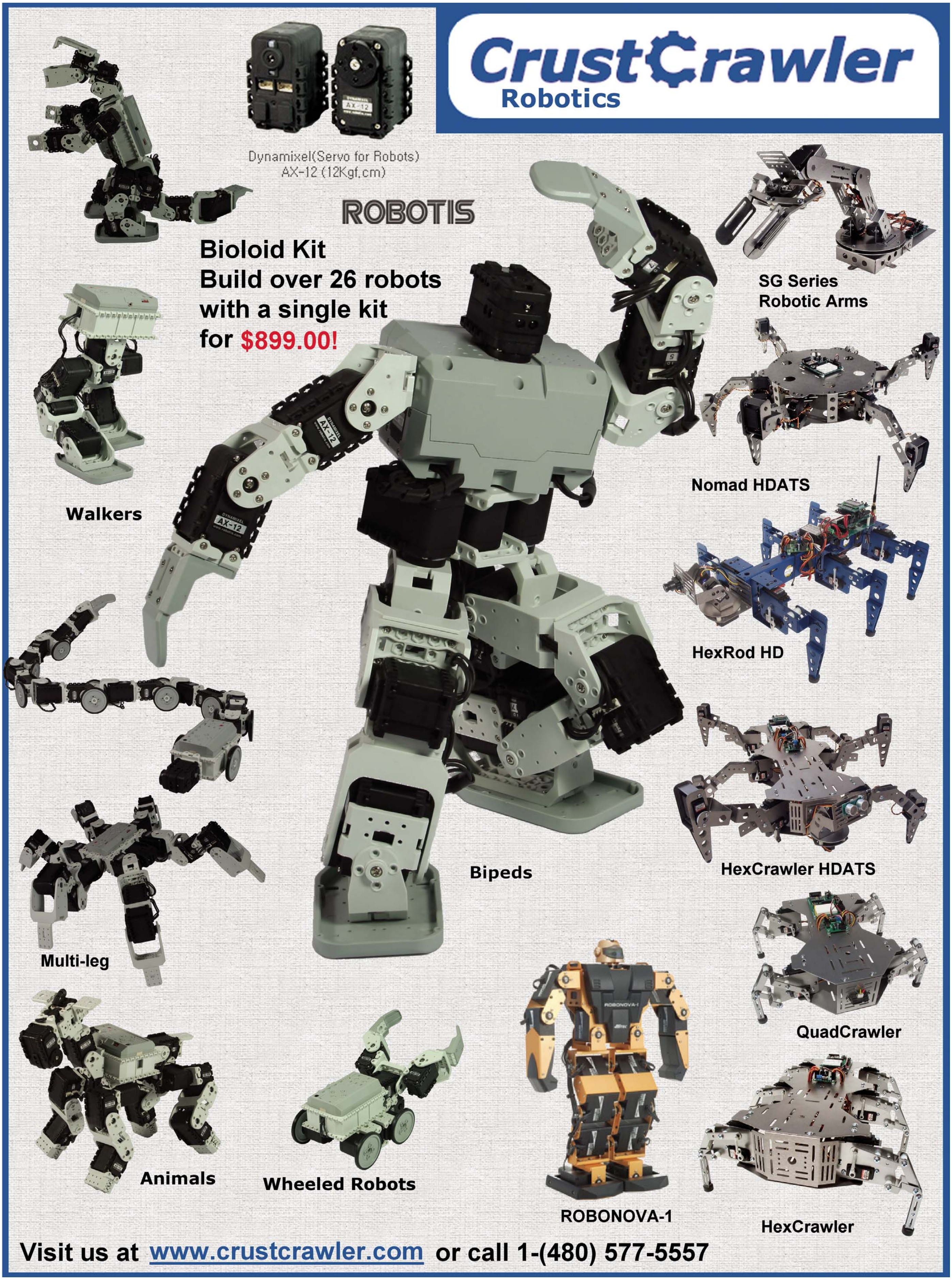
Live Demos of the ULB

While demos are held on military bases and by invitation only, I'm sure some of our readers know how to be invited. Plans are to put the ULB in production soon, perhaps as early as 2008. SV

RESOURCES

Boeing
www.boeing.com

Rolls-Royce aircraft engine
www.rolls-royce.com/civil_aerospace/downloads/helicopters/m250c30_c30p.pdf


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

ASK MR. ROBOTO

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?

by Pete Miles

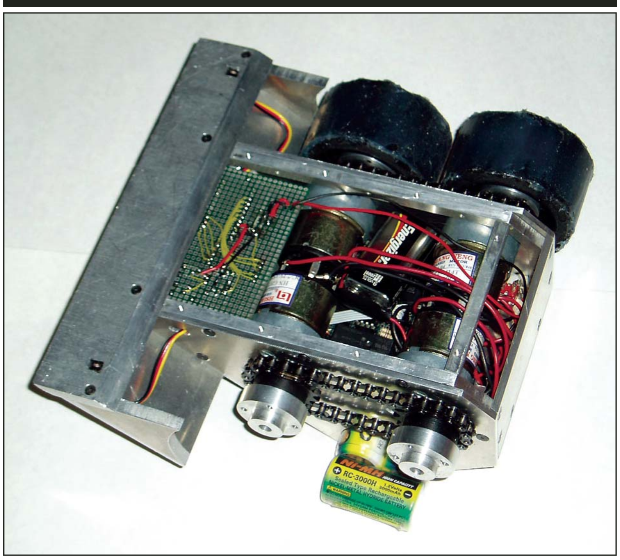
Q. I am in the process of designing a 3 kg sumo robot for the upcoming RoboGames in San Francisco, CA. I am planning to make a four wheel drive robot and I would like your opinion on which drive configuration is better: using four separate motors to drive each wheel, or one motor driving two wheels on one side of the robot and another motor driving the other two wheels.

— Dave Malony

A. First off, the key to robot sumo is making sure that your robot works all the time and will not drive off the ring by itself. This is the number one problem I have seen with all sumo robots regardless of their weight division. It is quite frustrating to see your robot lose because the batteries go dead or you forgot to plug them in. Or, your microcontroller could reset, a wheel could fall off, or the edge sensor might not see the edge of the ring and drive off it on its own. If you can prevent these mishaps, then you will have an above-average robot.

Both of the configurations that you suggest will work just fine in your robot, and many competitors use both of them very effectively. Now, assuming the overall torque is the same for both configurations, then I would say the four motor configuration is the weaker of the two designs. The reason for this is that if the front (or rear) of the robot is lifted up slightly, then one set of wheels will no longer be in solid contact with the ground. This will cut the available pushing torque from four motors down to two motors, and the robot becomes less effective at pushing.

Figure 1. Bottom view of a 3 kg sumo robot showing four motors to drive four wheels and on each side, the wheels are driven together through #25 sprockets and chain.



One option that you didn't mention is using four motors *and* gearing the wheels on each side of the robot together. This will give you the best of both worlds. Then, when the front of your robot is lifted up in the air, the motors driving the wheels that are not in contact with the ground will transfer all their torque to the rear wheels, so your robot doesn't lose any of its pushing torque. Figure 1 shows a photo of the bottom of one of my 3 kg sumo robots that uses four gear motors from Lynxmotion (**www.lynxmotion.com**) and a set of #25 plastic sprockets and steel drive chain from Small Parts (**www.smallparts.com**) to connect the wheels together.

In many cases, using two larger motors instead of using four smaller motors will generate more overall torque for pushing, especially when using cordless drill motors. But the larger motors will have a much higher current draw from the batteries. Depending on how easy it is to recharge or replace the batteries during a tournament, higher current drawing motors could be a disadvantage.

If the combined torque for a four motor robot is similar to the combined torque of a two motor robot, I would go with the four motor robot, and gear the wheels together. If the overall torque of a two motor robot is greater than a four motor robot, then go with the two wheeled robot and gear the wheels together.

Q. Do you know of any companies that sell higher torque servos than the 330 oz-in torque Hitec HSR-5995TG servo?

— Kris Santos

A. I would suggest that you take a look at the Tonegawa Seiko servos sold by CK Design Technology, Inc. (**www.ckdesigntech.com**), and Vantec (**www.vantec.com**). Though I haven't personally used them, they do have some impressive specs. Table 1 shows a set of specifications for the PS-050 and the SPS-105 servos. As you will notice, they have considerably higher stall torques than the HSR-5995TG servo, but they are also larger in size. Both of these servos are mounted in aluminum cases for added strength, and they use the standard R/C pulse with signals to control their position. The PS-050 servo uses a standard Futaba spline for servo horns. The PS-150 servo uses a special 33 mm (1.3") diameter hub that is mounted on a 12 mm (0.47") diameter keyed shaft. This hub is used instead of the standard servo horns for mounting linkages and hardware.

When you go to their websites, you might think that these servos are expensive. Yes they do cost more than HSR-5995TG servos, but with the cost per torque ratio, these servos are rather inexpensive when compared to other R/C servos. If you get any of these servos for your application, please write a short article about what you did with them. I am sure all the readers of *SERVO Magazine* would be very interested in reading about your experiences with these monster servos.

Model	Model PS-050	Model PS-150
Voltage	4.8V-8.4V	12 ±2 VDC
Idle Current	8 mA @ 6.0V	35 mA @ 12.0V
Stall Current	4.5 A @ 6.0V	9 A @ 12.0V
Stall Torque	907 oz/in @ 6.0V	5280 oz/in @ 12.0V
Speed	0.29 sec/60°	0.6 sec/60°
Travel	±60°	±45°, ±90°, ±180° options
Size	3.94" x 1.73" x 3.66"	5.19" x 2.15" x 4.66"
Weight	10 oz.	27.5 oz.

Table 1. Specifications of Tonegawa Seiko Servos.

Q. What is the difference between speed control and torque control on an electric motor?

— Jim Derrik

A. Well, that depends on your point of view. Technically speaking, torque control is another way of saying speed control since you can't push torque into a motor. Motor torque is simply a reaction force that the motor generates when it is trying to maintain its design speed for a given applied voltage.

When an external force is applied to a motor's drive shaft, the motor's electrical current draw will automatically increase due to the laws of physics (electricity and magnetism). As a result, the motor's speed will decrease. Equation 1 shows a simplified equation that shows how the motor's speed is as a function of the applied voltage to the motor, V_{in}, and the amount of current, I, the motor is drawing. K_v is a constant that is specific to the motor's design, and, R, in the motor's internal resistance. As you can see in this equation, as the current draw increases (for a fixed battery supply, V_{in}), the motor's speed decreases.

Equation 2 shows how the current draw, I, is a function of the motor's torque. The motor's torque constant, K_t, is also specific to the particular motor design. A small amount of current, I_o, is needed for the motor to overcome its internal frictional losses. As this equation shows, as the motor's torque increases, so does the current draw.

EQUATION 1:

$$RPM = K_v(V_{in} - IR)$$

EQUATION 2:

$$I = I_o + \frac{Torque}{K_t}$$

A speed controller is used to vary the applied voltage to the motor to change the motor's shaft speed. From Equation 1, higher voltages, V_{in}, will result in higher motor speed. In an open loop system, the speed is assumed to be directly proportional to the applied voltage and it assumes that the changes in motor torque are negligible. In a radio control system, operators will increase the voltage in response and applied torques to obtain the speed that they want.

In a closed loop speed controller, there will be some sort of sensor that measures the actual motor speed, such as an encoder. A microcontroller is used to measure the actual speed (encoders) and compare the results to the desired speed, and will adjust the applied voltage, V_{in}, so that the actual speed will be the same as the desired speed. Changes in externally applied torques from the environment will change the motor's actual speed, which a closed loop speed controller will compensate for.

In essence, a torque controller is used to move a motor to a desired position or distance, within a certain time period or at a certain speed to overcome some external resistance or force (like gravity). In many applications, the important part is that the motor actually makes the move.

The implementation of a torque controller is essentially the same as a speed controller. Some sort of sensor is used to determine if the motor (or robot) is moving according to a desired plan. A microcontroller is used to measure the actual motion of the robot (or motor) and compare the results to the desired set points. It will then adjust the applied voltage, Vin, so that the motor is responding properly.

As you can see, a speed controller and a torque controller are the same type of device. They both take an input signal and then adjust the output voltage to drive a motor. In most cases, the input signal comes from the same source — either as a commanded position (and/or velocity) or a voltage level. Closed loop systems can take inputs from different types of sensors, such as encoders, current meters, accelerometers, and then a microcontroller is used to generate the output signals to drive the motors.

Keep in mind that torque and speed are two different things, and with a motor, they are coupled together. They both affect each other. You can have low or high torques at low speeds, and you can have high or low torques at high speeds. Motor torque is generated when the motor is moving slower than what it should be for a given applied voltage, and to increase the torque, the applied voltage is increased. A lot of people look at Equation 2 by itself and think that torque can be a function of current draw. This is incorrect since the applied voltage and speed are also part of the model.

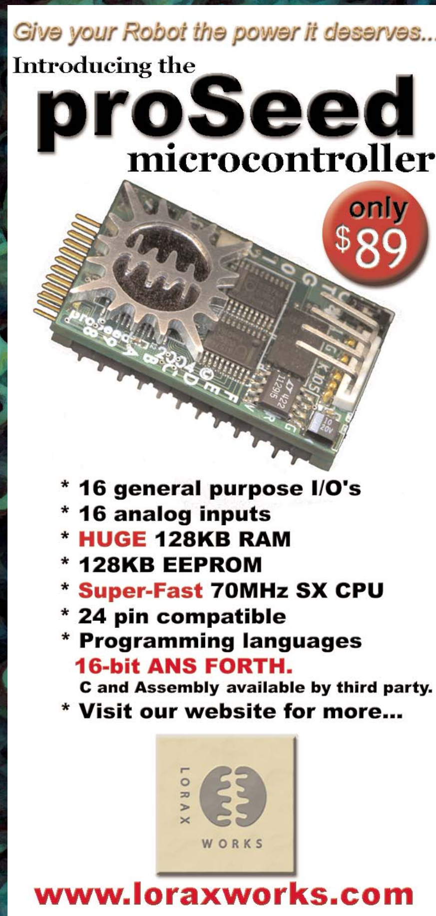
Q. Is wood a good material for robot bodies? The reason I ask is that I don't see any robots made out of wood.

— Bob Idassy
Pittsburgh, PA

A. Wood makes a great material for making robots. I am pretty sure that most robot builders have used wood at one time or another in their robots. Wood is fairly strong, lightweight, and inexpensive. It can be easily obtained from hardware stores, woodworking stores, craft stores, and hobby stores. Inexpensive tools can be used to create very complex geometries, and wooden robots are easy to repair if they get damaged.

For smaller robots — like desktop robots — I would recommend the model airplane plywood found at most hobby stores. It is a little more expensive, but it is very strong. For larger robots, regular plywood works well. Marine grade plywood is very strong, but very expensive when compared to other types of plywood. For structural parts, consider hardwoods like oak or maple. Regular pine is great for prototyping since it is cheap and easy to shape and fasten/glue to other parts. The softer woods should be avoided in areas where high stresses can occur since they can split when shocked or overloaded.

Another thing to consider is that wood is often used in combination with other materials. I have seen a lot of robots that have wooden base plates, wooden shells, and wooden mounting brackets. It is very versatile, and it is often overlooked as a good building material. Remember, all the great sailing vessels in the world havebeen made out of wood, and high performance racing boats are made out of wood. Airplanes have been made out of wood; high performance model airplanes are still made out of wood. Bridges are made out of wood and most of our homes are made out of wood. With the low cost and availability of modern materials, people are forgetting that wood is an excellent building material. So, a robot can be made out of wood, and they make good looking robots. **SV**



Robotics Showcase

NEW PRODUCTS

ACCESSORIES

CH Multi Functional Panel

CH PProducts and Ergodex announce the CH Multi Function Panel (MFP). The revolutionary MFP enables flight simmers to design their own cockpits, either duplicating any existing aircraft from a Cessna 172 to F-16 Fighting Falcon to a Boeing 777, or designing their own novel cockpit. Each MFP comes with 25 keys which can be positioned, removed, and re-positioned to any location on the panel's "active surface," making each MFP completely customizable. Each of the 25 keys is movable, programmable, wireless, and has no batteries. By using MFPs to emulate avionics panels, communications panels, or GPS panels, flight simmers get added realism in their desktop or home-built cockpits. Through the placement and programming of keys on the MFP, any avionics panel can be emulated.

The CH Control Manager™, optimized for flight simulation and other entertainment software, is combined in the MFP with Ergodex' technology, allowing up to 16 panels to be used on a single PC. The MFP also includes a removable clear tray that allows users to create custom key templates to be placed underneath the tray, aiding in the identification of key placement and adding realism. Sample templates — such as the one shown in the photo — will be available at no charge on the CH website, as well as the CH Hangar website.

The MFP is designed for any Windows application, including — but not limited to — PC Gaming and Flight Simulation. The MFP has the same functionality as a keyboard, with some major advantages. Each key can be removed and re-positioned wherever the user wants on the MFP tray. Each key is held in place with a re-usable, inexhaustible adhesive, sometimes called "Molecular Velcro," which allows re-positioning the keys over and over again, without losing any "stickiness." The MFP can be seen on the CH Hangar website and ordered at **www.chproducts.com/shop/usb.html#23**.

With CH's Control Manager — which supports Windows 98, ME, 2K, XP, and XP 64 bit — a flight sim enthusiast can add up to 16 CH controllers, including multiple MFPs. Software such as Flight Sim, Combat Sim, and so on, sees the controllers as one device. With Control Manager, games which are limited to supporting only one controller can be used with many controllers.

"This product is our first thrust in OEM channel for bringing Ergodex technology to customers in new markets. Working with CH Products to bring the Ergodex Engine to flight simmers and other gaming enthusiasts is an exciting new avenue for our technology" said Scott Rix, CTO of Ergodex. "New features and new implementations of the Ergodex technology are coming not only in products from Ergodex, but also in OEM product lines, such as CH's MFP, and in the form of licensed technology in new instrumentation both within and outside the computer industry."

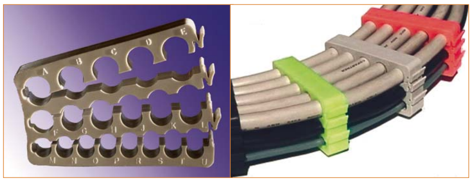
The MFP allows you to program each key in any way you want, depending on the game. Each key can perform as any combination of the following: keystrokes, joystick buttons, mouse buttons, joystick axis, and mouse axis. Additional trays and keys may be purchased so that the customer can set up one tray for one game, another tray for another game, a third tray for Windows applications (such as Photoshop or Word), and so on. The MFP includes one CH Panel, one clear key tray, one set of keys (keys 1-25), and CH Control Manager

Software version 5.0.

For further information, please contact:

CH Products or Ergodex — Website: **www.chproducts.com** or **www.ergodex.com**

The Ultimate Cord Organizer Clip



Delta 9 Products has introduced the Ultimate Cord Organizer Clip. Rick Nelson — DNP Product Manger — explains, "our product provides an innovative way for the professional to organize and track cables, cords, and wires between electronic devices."

Each slot in the Ultimate Cord Organizer Clip has a letter assigned to it and retains the cables and cords when open. The Ultimate Cord Organizer sorts by size and type, but also allows you to channel, isolate, and track cords, cables, and wires.

The Ultimate Cord Clip is available in four standard colors: black, gray, neon orange, and neon green. Custom colors are also available. The Clip is sold in five packs ($9.95), 10 packs ($18.90), 20 packs ($35.90), and 60 pack ($102.35) quantities.

For further information, please contact:

Delta 9 Products — Tel: **530 • 333 • 2014** Website: **www.ultimatecordorganizer.com**

SENSORS

USB Motion Detection System

Kadtronix introduces the USB Motion Detector (UMD) system. Costing as as low as $91 in single-unit quantity, the system is comprised of the following elements: USB motion detector, USB wiring interface, and Windows software.

This product was developed for applications needing to perform motion-triggered actions under PC control. Combined with a Windows PC or laptop, the UMD becomes the basis of a smart motion-activated system. Automatically send email, run applications, play .wav sounds, set digital outputs, play multimedia presentations, and more.

Utilizing existing Windows drivers, no custom drivers are needed. Included with the UMD is the USB Digital I/O Commander "Digio" software. This software allows you to define and customize system parameters for your specific application. Using Digio, configure the system for any of these applications and more:

- Security systems
- Industrial control
- Robotics
- Kiosks
- PowerPoint presentations
- Research

Digio can be configured for use in motion-activated slideshows and videos, making it well-suited for use in computer-based kiosks. Run motion-activated multimedia presentations in shopping malls, airports, convenience stores, etc.

The USB interface features 16 configurable I/O signals and six-foot length of cable with attached USB connector. (May be used with a USB extender for distances up to 100 feet or more.)

This do-it-yourself (DIY) system provides a low-cost option for those with knowledge of low-voltage wiring and assembly. (Motion detector and AC power adapter sold separately.) Also available as a complete turn-key system including motion detector, USB interface, and Digio software, all pre-wired and ready for installation.

For further information, please contact:

Delahoussaye Consulting — Tel: **321 • 757 • 9280** Email: **info@kadtronix.com** Website: **www.kadtronix.com**

EVENTS CALENDAR

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

Most of the robot action is happening in India this month. Two university-level, nationwide competitions are being held simultaneously. The Robotix event is being held in West Bengal and includes events with intriguing names like Softandroid. Meanwhile, down in Tiruchirappalli — or Trichy as it's commonly called — the National Institute of Technology is holding the annual Pragyan technical festival. The robot portion of Pragyan is known as RoboVigyan and includes events named Trailblazer and EyeRobot.

Closer to home, you can find the APEC Micromouse Contest happening later this month in Anaheim, CA. As I say every year, if you have a chance to see this one, don't miss it. These are some of the quickest little robots you'll ever see. Watching them find their way through a complex maze is an impressive sight.

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

February

1-4 **Robotix**
IIT Khargpur, West Bengal, India
A national-level competition. Events include Fastrack Manual, Fastrack Auto, and Softandroid.
http://gymkhana.iitkgp.ac.in/robotix

1-4 **Pragyan**
National Institute of Technology, Trichy, India
Events include TrailBlazer and EyeRobot.
www.pragyan.org

26 **APEC Micromouse Contest**
Anaheim, CA
One of the best-known micromouse competitions in the United States. Expect to see some very advanced and fast micromouse robots.
www.apec-conf.org

March

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

9-10 **AMD Jerry Sanders Creative Design Contest**
University of Illinois at Urbana-Champaign, IL
Check the website for the details of this year's contest.
http://dc.cen.uiuc.edu

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

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

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

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

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

31 **Penn State Abington Mini Grand Challenge**
Penn State Abington, Abington, PA

Autonomous outdoor ground robots must navigate on and off-road around the campus, avoiding obstacles.
www.ecsel.psu.edu/~avanzato/robots/contests/outdoor/contest05.htm

April

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

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

12 **BattleBotsIQ**
Location TBA
Student-built, radio-controlled vehicles destroy each other for the educational value.
www.battlebotsiq.com

12-14 **FIRST Robotics Competition**
Atlanta, GA
National Championship for the regional FIRST winners.
www.usfirst.org

14-15 **Trinity College Fire-Fighting Home Robot Contest**
Trinity College, Hartford, CT
The well-known championship event for fire-fighting robots.
www.trincoll.edu/events/robot

20 **Carnegie Mellon Mobot Races**
CMU, Pittsburgh, PA
The traditional Mobot slalom and MoboJoust events.
www.cs.cmu.edu/~mobot

COMBAT ZONE

Featured This Month

PARTICIPATION

Frequency Control and Bot Blocking — A Real-Life Near Miss

● by Kevin Berry

One of the keys to a safe combat event is maintaining good frequency control. Another is proper pit safety. This is preached regularly by all event organizers — including myself — yet I fell into the "unsafe participant" trap at the recent Daytona Area Robot Tournament.

Usually, I keep each of my bots on a separate frequency. Since they usually don't run at the same time, this was mainly because I bought crystals in pairs (Tx and Rx) so I never had duplicates. This event, however, I found myself with an extra Rx crystal, so I thought I'd make life easier by running my antweight and beetleweight on the same frequency, eliminating the need for transmitter frequency changes. I never realized this would lead to a near miss in the pits!

For the whole event, I followed my practice of blocking my bots up off their wheels in the pits. I felt a little silly with this, since they were all insect class pushy bots, but I had my son with me, so following rules is a life lesson, as well as an event requirement. Late in the day, my beetle was practically destroyed in the arena. I took it back to the pits, tried to switch it on, and verified it was one

Even after massive combat damage, John Henry got a second wind in the pits and scored a last hit as a runaway bot.

dead duck. I left it sitting on the table, and went over to the arena to watch my son fight the antweight.

Suddenly, a fellow competitor appeared, holding the supposedly dead bot, now fully alive and kicking. Somehow, a loose connection had re-connected, and when we powered up the transmitter, it took off, striking him in the arm. While this was a no-harm incident, it sure shook me up. If I'd verified the power switch was off — even on a dead bot — or blocked up the wheels, this would have been no big deal. Had it been a bot with an active weapon, serious injury could have occurred.

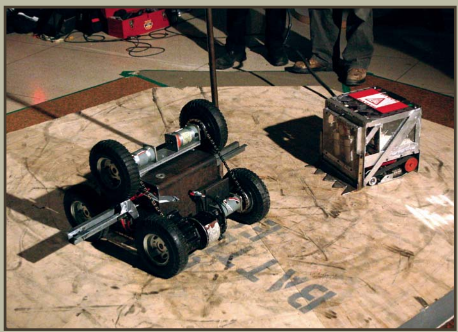
Lesson learned for Kevin: Rules are meant to be followed, even on dead bots. Just like many people are killed with "unloaded" guns, dead bots can be dangerous too! **SV**

ROCK 'EM, SOCK 'EM ROBOTS! STRONG MEN DANCER PAINTER ROBOTS! SUMO!!

● by Simone Jones

The Ontario College of Art & Design (OCAD) has been running the SUMO Robot Challenge on an annual basis since 1992. The event is the brainchild of Norman T. White, a former professor who was instrumental in the development of the electronics, robotics, and mechanics curriculum at OCAD. The event has attracted wide participation from artists, designers, engineers, and laymen inventors. The strength of the event is derived from its celebration of invention, ingenuity, humor, and good-natured competition. OCAD has developed a partnership with the Ontario Science Centre with the intention of broadening the appeal of the event to students and teachers from local high schools, as well as at-risk youth.

A key element of the event is to excite students about the possibilities that can be derived from working with technology. All of the robots in the competition are designed and built by the competitors. Since budgets are an issue for many of the competitors, you frequently see machines built from surplus and salvaged parts. This is encouraged and supported by the event organizers because it reinforces a creative approach to the art and design process rather than focusing on robotic machines that come "ready-made" and "out-of-the-box."

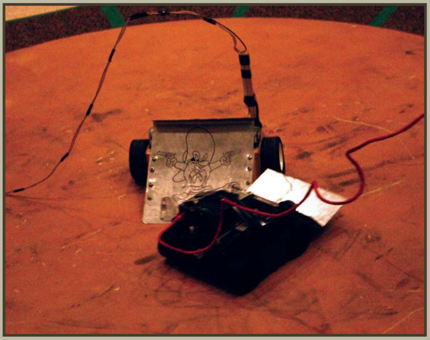


Furious George and Piece Maker — Clever Class.

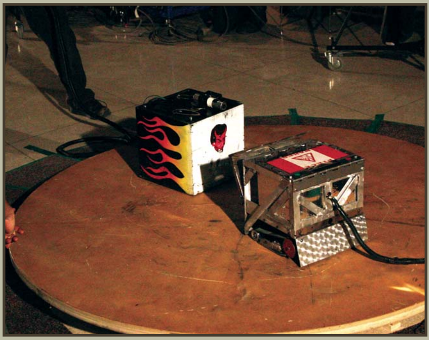
In fact, the local electronics surplus store, Active Surplus, is a regular supporter of the event. Luckily, Active Surplus is located just five minutes from the OCAD campus so competitors can run down the street to pick up spare parts between bouts!

The event has brought awareness of OCAD's Integrated Media

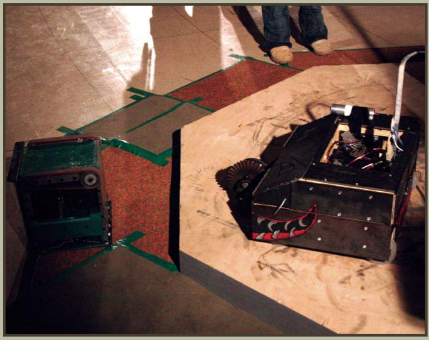
Lightweight competition.



Piece Maker and Dougy B — Classic Class.



The Creature and Baby Hulk — Clever Class.



Program out to the larger community. We have a healthy rivalry with Ryerson University and we have alumni and non-students who compete regularly — returning year after year to test their machines and their updated designs.

Notable competitors include Steve Hazard, Jimmy Green, Doug Back, Stephanie and Katherine Gavrylec, and Brian and Patrick Stuurman. It should also be noted that since 1992, Ray McLeary (the "Amazing Ray") has been our MC and Duane Moulder has been our Referee. Their volunteerism and commitment to the event is tremendously appreciated.

The current classes of competition for the event are: SUMO Classic, SUMO Clever, SUMO Lightweight, SUMO Autonomous, SUMO Lightweight Autonomous, Dancer/Painter, and Tug of War. This year's event will be held in the OCAD Auditorium, 100 McCaul Street, Toronto, Canada on March 3, 2007. To get more information about the event, got to **www.student.ocad.on.ca/~sumo/**. This year's event is being organized by Simone Jones, Assistant Dean in the Faculty of Art. SV

All photos are courtesy of Robert Sherwin.

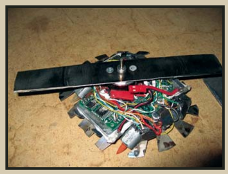
EVENTS

RESULTS — November 14 - December 17

Daytona Area Robot Tournament was held on 11/18/2006 in Daytona Beach, FL. Results are as follows:

• *Antweights* — 1st: "Babe The Blue Bot," box, Legendary Robotics; 2nd: "Ant from Hell," spinner, V; 3rd: "Ultimate Ultimatum," spinner, Overvolted Robotics.

• *Beetleweights* — 1st: "Ron," wedge/saw, Overvolted Robotics



(Currently ranked #1); 2nd:"Nuclear Kitten," spinner, Test Bot; 3rd: "Upper Cut," spinner, Logicom.

• *Fairyweights* — 1st: ""Doodle Bug," box, Ninja; 2nd: "Skeeter From Hell," wedge, V; 3rd: "Strike Terror," wedge, V.

• *Mantisweights* — 1st: "Mantis From Hell," wedge, V; 2nd: "Tom," kludge, Diamond Back.

Robot Rebellion 6.4, The Robot Shoot-Out was held on 12/2/06 at Mike's Hobby Shop in Carrollton, TX. One pound Antweights were fought. Results are as follows:

• 1st:"Dark Pounder," vertical blade spinner, Dark Forces; 2nd: "Micro 44," horizontal blade spinner, Dark Forces; 3rd: "Number 1," Wedge/Clamp bot, A&W.

Pennbots was held on 12/16/06 at Yellow Breeches Middle School in Boiling Springs, PA. Results were not available at press time. SV

TECHNICAL KNOWLEDGE

Radio Mod — Spring Loading the Left-hand Sticks

● by Kevin Berry

There are two kinds of bot drivers in the world: single stick and tank drive. Me, being almost 50 years old and having grown up driving old-fashioned farm equipment, I'm a tank drive kind of guy. My kids, being of the video game generation, are fine driving a bot with just one stick. So we've settled on a scheme that for our pusher bots, we use both radio sticks in pure tank drive fashion. For spinners, we use the right-hand stick mixed for drive and the left-hand for weapon control.

Many radios come with the right-hand stick spring loaded both ways (front-to-back and left-to-right), but the left stick spring loaded only left-to-right. The front-to-back movement of this stick is lightly ratcheted, since it was originally designed as a throttle for boats and planes. This sure makes it hard to drive tank style, and for weapon use can be downright dangerous, since it requires careful positioning to turn the spinny thing off. So, I like to modify the left stick on all my radios to add the spring-loaded function.

This isn't too tough of a job, although it does depend on the manufacturer.

Legal note: Modifying the electronics of radio control gear is only to be done by an FCC licensed technician. This is only a mechanical mod, and as long as you don't move or change any wires or electrical components, it's okay.

FIGURE 1

FIGURE 2

Figure 1 shows a typical four-channel, FM radio. While cases and displays vary, the mechanism connected to the sticks is very similar in most radios. Figure 2 shows most of the tools needed, along with a peek at the parts we're going after. First step is to remove the battery and crystal, mainly to get them out of harm's way. Then remove the back of the radio. The Quattro has six screws; your transmitter may vary. However, they usually all come out from the back. Resist the temptation to remove any screws from the front unless you are sure it's needed to get the back off. Figure 3 shows the back removed, with a peek at the softy, creamy insides of the transmitter.

I digress for a moment into the world of robotic hacker philosophy. According to Berry's Law (which I just self-named), there are two kinds of jobs in the world: easy ones and hard ones. We are about to find out which kind your radio requires. Locate the little metal tension spring on the front-to-back mechanism of the left-hand stick. It should look something like the one on Figure 4. Remove the screw holding it on, and take off the spring and the screw. If you have fully restocked your karma lately, there will already be a lever arm and coil spring in place. The stick will move front to back and spring itself to center. In this case, you are done. Sometimes you can put the tension spring back in place, but upside down, just as a handy way to save it for the future. Put the back on the radio, reinstall the battery and crystal, and stop reading this article.

FIGURE 3

Due to a few missed payments on my karma, I got the "hard" job. Figure 4 shows the Empty Gaping Hole where the manufacturer chose to save a few cents and not install the needed parts. Using fine needle-nosed pliers, carefully unhook the spring from the one installed lever, and lift the lever and spring out. If you can't easily unhook the bottom of the coil spring from the nub deep inside the mechanism, just leave it in place. Figure 5 shows a typical example of the lever and spring. Now, bot builder extraordinare, you get to craft a copy of the lever. I use 1/8" polycarbonate (e.g., Lexan™) but any similar material will serve. I outlined the existing lever onto the paper covering of the poly, then cut it out with a coping saw.

The result — due to my rather poor craftsmanship skills — is functional but ugly, as shown in Figure 6. I am usually able to find a near-exact match for the spring at a good local hardware store. In this case, the one I found matches the length and diameter of the original exactly, but was a bit stiffer in pull.

This stiffness meant I decided to install the manufacturer's part in the front-to-back position, so it exactly matched the feel of the right-hand stick. I rarely use the left-to-right function on the left stick, so a bit stiffer feeling is fine there. Now, again in accordance with Berry's Law, you will find out just how much of a deficit you are running with "The Powers That Be." If you've been a very, very good bot builder

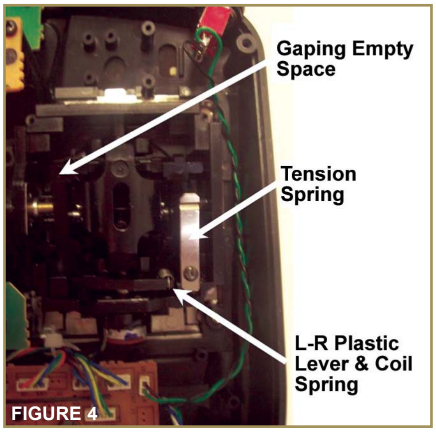
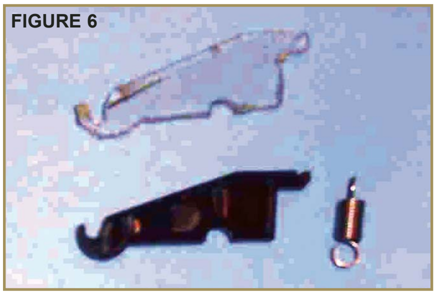


FIGURE 4

FIGURE 5

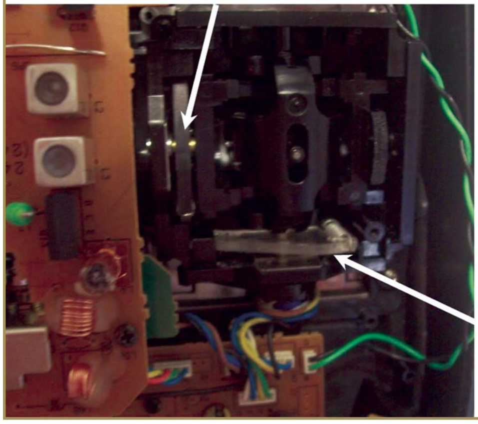
and pit buddy, you'll be able to re-hook the spring on the bottom nub, slip in the lever, and hook the loop on the top of the spring onto the lever. Next, install the second one you've manufactured. Ditto the above discussion on your luck. If everything works out right, things will look like Figure 7.

FIGURE 6



If, however, after several dozen attempts, you just can't get everything installed, or if you have superior mechanical skills and just like to take things apart, you'll have to remove the whole stick assembly from the radio. Things vary much more in this area between manufacturers. On my radio, unless you have a very long, thin screwdriver, it means moving a circuit board aside to access one of the screws holding the assembly in.

FIGURE 7



Warning: When you get the assembly out, it will remain attached by two pairs of wires to the positioning pots. Also, it will probably come apart like a Chinese puzzle. On the good side, with an extra set of hands, installing the levers and springs goes very quickly. Reinstall the assembly and button up the radio.

The first time I did this job it took about two hours, including manufacturing the replacement lever and re-assembling things wrong at least twice. Now, I can do the "flip the lever" style in about 10 minutes, and the "build a new lever" model in under an hour. This leaves me more time to spar with my junior drivers — their single stick against my two — teaching me again the trueness of the standard combat robot motto, "Learn To Friggin' Drive"!

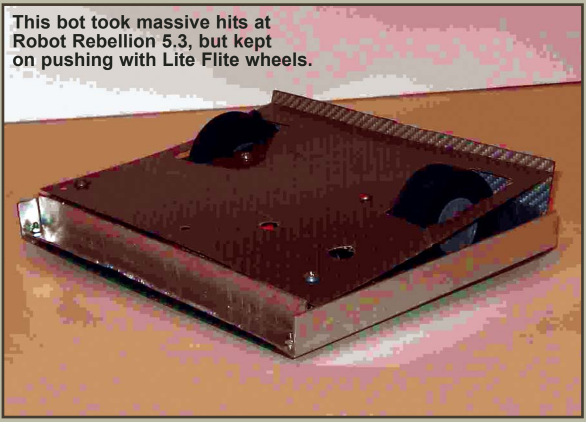
Turns out it doesn't matter how well you've spring-loaded your left-hand stick, if you have no depth perception or good reaction times!. SV

PRODUCT REVIEW — Dave Brown Products Lite Flites

● by Martin Koch

Looking for the perfect wheel for your small robot? Dave Brown Lite Flite wheels might be the best wheels to get your project rolling. I have used Lite Flites on three antweight pusher combat robots competing in four SWARC events. Prices from The RobotMarketplace range from $3.79 for 1-1/2" diameter to $12.99 for 5" diameter wheels. They come in packs of two.

Often used as landing gear wheels for model airplanes, these wheels are perfect for combat robots because of the soft, yet durable, foam construction. It is a very dense material that is also extremely lightweight, which is crucial to small combat robots. Foam may not sound very, well, sound, but the foam these wheels are made of is strong enough to carry them and are becoming some of the most popular wheels in combat robotics. These wheels are not as durable as harder wheels, but absorb much more energy. Also, they can still handle a match even with chunks of foam missing. The foam has good traction, but many builders apply a coating of silicone to the tread for even more.



This bot took massive hits at Robot Rebellion 5.3, but kept on pushing with Lite Flite wheels.

Their durability became evident during a match against a fearsome horizontal spinner at Robot Rebellion 5.3. My pusher bot's exposed Lite Flites were getting cut and shredded, but absorbed the spinners' force. Only when the blade hit the hub I was using to secure the wheels to the shaft did the Lite Flites fly off the robot.

Speaking of hubs, many custom ones are available to secure these wheels to popular motors, such as Banebots and Copal gearmotors.

Dave Brown Products Lite Flites are the perfect wheels for many small robots. Their blend of strength and light weight is a win/win for builders and bots (**www.robotmarketplace.com**). SV

EVENTS

UPCOMING — February and March

CANCELLED!!

ComBots Cup 2007 — This event will take place on 2/9/2007-2/10/2007 in Oakland [illegible] to **www.robogames**[illegible]ther information [illegible]avyweight prize, $[illegible]eweight prize. Venu[illegible]edule dates are tentative a[illegible]e of publication.

Motorama 2007 — This event will take place on 2/16/2007-2/18/2007 in Harrisburg, PA. Go to **www.nerc.us** for further information. 150g-30lb Combat Event. Ants fight in 8' box, Fairies fight in the 5x5 insert. Beetles-Featherweights fight in 16x16 box. All completed forms and entry fees must be received by 1/15/07. This is going to be another awesome event at the Farm Show Complex!

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

ROBOT SIMULATION: *AI Behaviors*

by Bryan Bergeron

Developing a robot with advanced planning or collaborative task capabilities can be a painstakingly slow process that involves long sequences of carefully performed experiments, lots of uncluttered floor space, and an extensive library of algorithms. Since each advance may entail dozens of failed attempts, many roboticists turn to computer-based simulation to save time and money — especially when resources and robots are limited or expensive.

Robot simulation isn't limited to research and development. Most factory robots ship with robot simulations that enable operators and programmers to work out process and coding challenges before potentially ruining a robot or jeopardizing an entire assembly line.

Consider the potential time savings of working with a simulated robot instead of a physical robot in determining the best algorithm for navigating through a given environment. A properly implemented robot simulation may make a hundred virtual runs in the time required for one trial with a real robot. In addition to time savings, relying on a simulation for initial data minimizes the exposure of a real robot to physical damage.

Thanks to advances in simulation technology — including work in the engineering and video game industries — many of the simulation techniques used by NASA to develop multi-million dollar planetary rovers are affordable and readily available to the general robotics community. Furthermore, these simulation techniques apply to the spectrum of robot development and operation activities, from designing high-level AI behaviors, evaluating the physics of the mechanical plant, and optimizing system operations, to evaluating sensor placement and circuit operation.

This article (the first in a series on robot simulation) provides an introduction to simulation technology and examples of how readily-available simulation tools can be used to develop simulated robots that exhibit AI behaviors.

Simulation

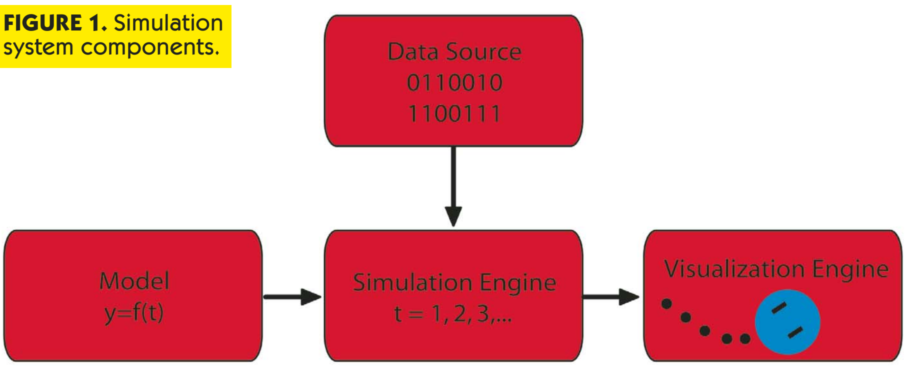
Although simulation is often referred to as a singular activity, it actually involves two separate processes: modeling and simulation. Modeling involves the formulation of mathematical equations, logical descriptions, and algorithms that describe a robot and its interactions with the environment. Simulation is the dynamic evaluation of the model that is triggered by some combination of time, events, and the value of intermediate simulation results.

Components

The components of a typical robot simulation include one or more models, a simulation engine, a data source, and a visualization engine, as shown in Figure 1. The simulation engine solves the equation(s) defined in the model, using data from a data source. The visualization engine formats the output into a user-friendly form, such as the 2D or 3D representation of a robot in a simulated environment.

In the simulation system depicted in Figure 1, the model consists of an algebraic equation of the form y=f(x), which relates variables to the passage of time. Each second, the simulation engine takes data from a sensor, a database, or other data source, and uses them to solve the model. The visualization engine can range from a text formatting utility to a high-performance, real-time, 3D rendering engine that can depict the robot within a realistic, virtual environment.

FIGURE 1. Simulation system components.



Time vs. Event-Driven

There are two basic simulation

paradigms: time-driven and event-driven. Time-driven simulations, often referred to as continuous simulations, employ a model defined as differential and/or algebraic equations that are assumed to vary continuously with advancing time. For example, the distance a robot travels under constant acceleration may be represented by the algebraic equation:

$$s = \frac{1}{2} at^2$$

where s is meters, a is acceleration in meters/sec/sec, and t is time in seconds.

Event-driven or *discrete* simulation models robot activity over time as separate events. Event-driven simulation lends itself to simulating the behavior of robots when there are large periods in which conditions don't change appreciably with time. Consider a bumper switch may remain dormant for extended periods, especially if the robot uses good obstacle avoidance algorithms. Discrete simulation models are commonly defined in terms of logic statements keyed to specific events, such as:

IF (event) THEN advance to next stage

Theoretically, it's possible to create simulated robots with tools that support either time or event-driven models. For example, the activity of a bumper switch can be defined as a probability function of time, as in:

$$P(\text{bumper event}) = f(t) + K$$

where $f(t)$ is an algebraic or differential equation that may contain variables reflecting the amount of clutter in the environment, the effectiveness of the collision avoidance sensors and related circuitry, the speed of the robot, and the sources of false triggering. K is a constant.

The point of using simulation tools is to reduce the amount of mathematical contortions necessary to simulate a robot. As such, most robot simulations are based on a hybrid design in which the simulation engine is capable of evaluating the model as a function of both time and events.

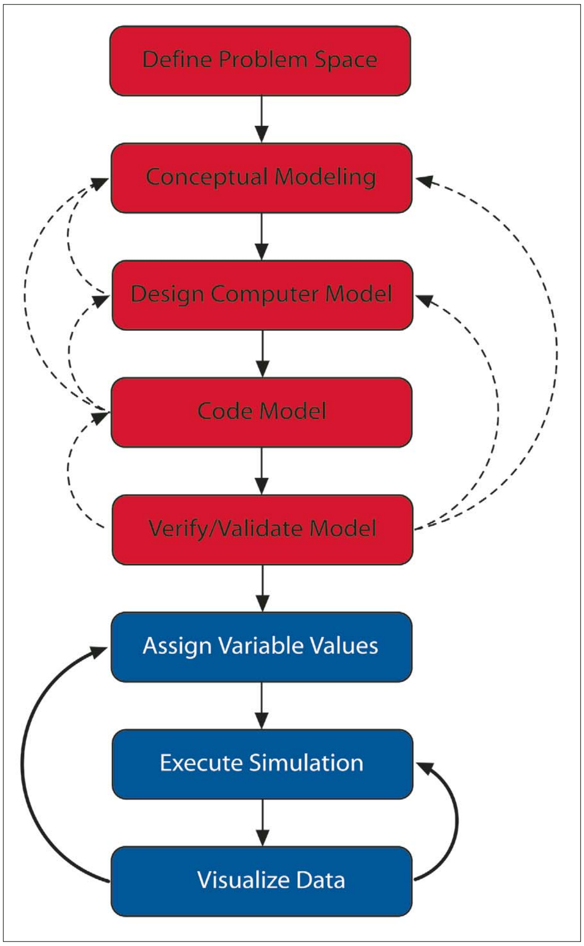


FIGURE 2. Robot simulation development process. Modeling activities shown in red; simulation activities in blue.

Focus

It's possible to simulate virtually every aspect of a robot. However, simulating everything from the effect of humidity on the response time of ultrasound rangefinders to slight variations in acceleration due to imperfect battery chemistry is usually prohibitively time consuming and computationally infeasible. Robot simulations typically focus on specific points of interest at the expense of others. A roboticist concerned with navigation and search algorithms can usually ignore the operation of the energy management system, for example. This focused approach reduces both simulation development time and computational overhead.

Process

Creating a robot simulation is an incremental, iterative process. As illustrated in Figure 2, the process begins with developing the model and ends with visualizing the simulation output. Some of these steps may be hidden, depending on the tool used to create the simulation. Even so, it's important to understand each component of the underlying process.

The modeling process starts with a definition of the problem space, such as high-level navigation behaviors. This is followed by conceptual modeling, which includes identifying the relevant variables, the degree of precision required, which elements of the real robot to include in the model, and which to ignore.

With a conceptual model in hand, the computer model is designed and then coded. Coding may involve defining C++ routines, using a high-level scripting language, or drawing in a CAD-like environment, depending on the available tools. Verification and validation are the final steps in the modeling segment of developing a robot simulation.

Verification is the process of determining whether the model coded in software accurately reflects the conceptual model. Verification is performed by testing the internal logic of a model to confirm that it is functioning as intended, for example. Validation involves assessing whether the operation of the software model is consistent with the real world, usually through comparison with data from the robot being simulated.

Figure 2 shows the incremental progression from a conceptual model that may have been formulated on a dinner napkin to a verified and validated computer model is an iterative process. Problems that appear in the modeling process may require revisiting previous stages, sometimes all the way back to conceptual modeling.

The simulation process proper begins with assigning value to the variables in the verified and validated model. The source of variable values may be a database, a random number generator, a function, or may be entered directly into the system with a keyboard and mouse. The subsequent simulation execution and visualization are the most important stages to the roboticist. The effectiveness of the user interface, quality of the graphic output, and ease of manipulating simulation variables define the overall usability and value of the robot simulation.

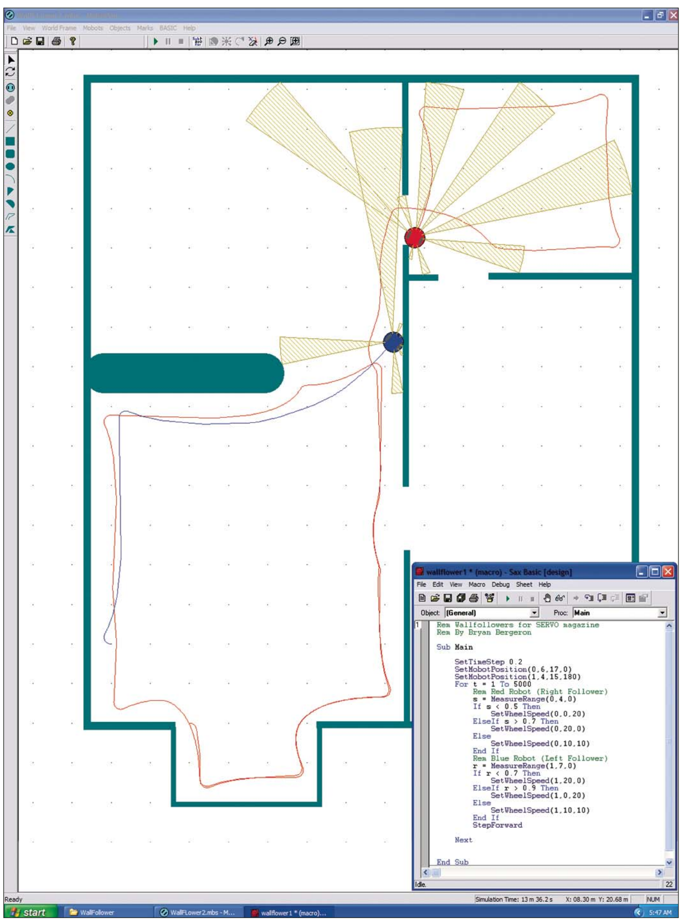
Simulating Robot AI Behaviors

The tools available for simulating robot AI behaviors such as navigation, path planning, learning, and search include spreadsheet programs, generic compilers, general-purpose simulation environments, game engines, and highly specialized simulations optimized for robotics. The downside of using MS Excel as a platform for robot simulation is poor performance, limited visualization capabilities, and extended development time.

A general-purpose spreadsheet — like a general-purpose language such as BASIC or C++ — is designed to solve a variety of problems. As such, it represents a compromise between development time, flexibility, and performance. Although a spreadsheet can be used to simulate a robot, a robot simulation developed in a spreadsheet will likely run several orders of magnitudes slower than a robot defined in an environment designed expressly for simulation. Similarly, coding a robot simulation in C++ may result in a robot simulation with a higher performance than can be obtained with a generic simulation environment. However, the time required to create a robot simulation from scratch in C++ will likely be several orders of magnitude greater than using an off-the-shelf simulation environment.

Consider that robot learning systems based on commercial neural network simulation are typically outperformed by classification systems developed in C++. However, creating a robot learning simulation with a dedicated neural network simulation may require only minutes of dragging and connecting icons with a mouse. With these caveats in mind, following is an introduction to several robot simulation development tools, discussed in the context of robot AI behaviors.

FIGURE 3. Dual wall follower simulation in MobotSim. The entire source code listing is shown in lower right.



Wall Following With MobotSim

Wall following is among the oldest and simplest of the biologically inspired robot behaviors. As mice demonstrate, consistently following either the wall to the left or right will eventually lead to an exit — assuming one exists. Creating a wall-follower robot is as simple as connecting a single IR or ultrasound rangefinder sensor to a differential drive robot. The control logic for a single sensor right wall follower typically takes the form:

```
IF Rangefinder Distance < Minimum THEN turn left
IF Rangefinder Distance > Maximum THEN turn right
ELSE go straight
```

Following the control logic, if the minimum robot-wall distance is, say, 0.5 m and the maximum robot-wall distance is 0.7 m, then the path of the robot should parallel the wall at a distance of between 0.5 m and 0.7 m. The success with which a robot parallels the wall, takes corners, and navigates through doorways depends on sensor pulse rate, accuracy, beam width and range, as well as robot speed, the size of the robot platform relative to the environment, and the operating environment.

One of the least expensive and easiest to use dedicated robot simulations capable of demonstrating wall following behavior is MobotSim, from MobotSoft. Thanks to an intuitive interface, use of the Basic language, and numerous examples, you can be running your first robot simulation 10 minutes after downloading the program.

I created the simulation run shown in Figure 3 by first drawing a room layout and then adding two robots — a right (red) and left (blue) wall follower. After defining the absolute sizes of the robots and the environment, I added the Basic code defining the behavior of each robot in the integrated development environment editor.

The following code snippet, based on a demonstration program, defines right follower behavior:

```
SetTimeStep 0.2
For t = 1 To 5000
s = MeasureRange(0,4,0)
If s < 0.5 Then
      SetWheelSpeed(0,0,20)
ElseIf s > 0.7 Then
      SetWheelSpeed(0,20,0)
Else
      SetWheelSpeed(0,10,10)
End If
StepForward
Next
```

From the code sample, it should be clear that MobotSim is a hybrid simulation, in that it is both time and event driven. In this example, the time step is 0.2, and the event is changes in the robot-wall distance. Decreasing the step size to 0.1 or 0.05 increases the precision of the simulation, at the expense of execution speed.

The first number in the MeasureRange and SetWheelSpeed triads refers to the robot. In this example, 0 refers to the right wall follower and 1 refers to the blue left wall follower. SetWheelSpeed defines the relative speed of the left and right motor for each robot. Driving one wheel and not the other results in a turn; driving both wheels equally results in straight-line travel.

As the robots move in the simulated environment, each leaves a trace showing the path taken over the time of the simulation. Starting in the left lower room in Figure 3, the left wall follower (blue) manages to follow a wall but then fails to hug a sharply rounded table jutting into the center of the room. Its journey ends jammed against a wall.

The right wall follower (red) was initially trapped by the jammed left wall follower, but escaped the room on the third loop. The simulated robot enters the room in the upper right, only to be jammed in the doorway. In attempting to better the performance, the angle and placement of the sensor can be adjusted, additional rangefinders can be added, and the control program can be modified.

Note the subtle variations in the paths recorded by the robot tracings. For example, the right wall follower veered away from the initial doorway. This behavior is presumably because of reflections from the corner formed by the doorway entrance. Unfortunately, there is no easy way to check for reflections, ghost images, and other sensor anomalies that might contribute to the behavior.

Despite limited access to low-level details, the functionality of the control algorithm defined in the IDE is primarily constrained by your fluency in Basic. For example, MobotSim is accompanied by a relatively sophisticated neural network-based learning robot, as well as a navigation program using force fields.

MobotSim does have significant limitations regarding the robot platform — it must be a two-wheeled, differential drive robot with simple rangefinder sensors. Even so, basic algorithms can be applied to virtually any robot platform, and the Basic code can be ported to a variety of microcontroller compilers. Although there is no microcontroller-export feature, the MobotSim Basic is reasonably compatible with Parallax PBASIC for the BASIC Stamp, the BASCOM AVR compiler for the Atmel line of microprocessors, and the BasicX development environment.

MobotSim is available as a $30 download. A full-featured, 30 day or 100 user demo is also available from the MobotSoft website (**www.mobotsoft.com**).

Collaboration With Webots 5

Effective collaborative behavior represents the final frontier of robotic AI. Robots working together and with humans toward a common goal require autonomous capabilities and the ability to communicate with or at least recognize the state of other robots and humans [1]. A common test bed for cooperative robot-robot behavior algorithms is robot soccer, typically held in tournaments at robotics conferences.

Webots 5 — a robot simulation environment originally developed for the Khepera differential-drive robot — provides a powerful environment for developing and testing algorithms that support cooperative behaviors. As shown in Figure 4, Webots provides a full-featured IDE, and the customizable visualization environment is rendered in 3D.

Working with Webots is much more involved than the relatively simple MobotSim. For example, fluency in MinGW is required to compile the C/C++ controllers. In exchange for this increased complexity, Webots provides support for complex functions such as messaging, trajectory recording, and supervisory functions related to collaboration. Predefined robot simulations support models popular in academic communities, including the Abido,

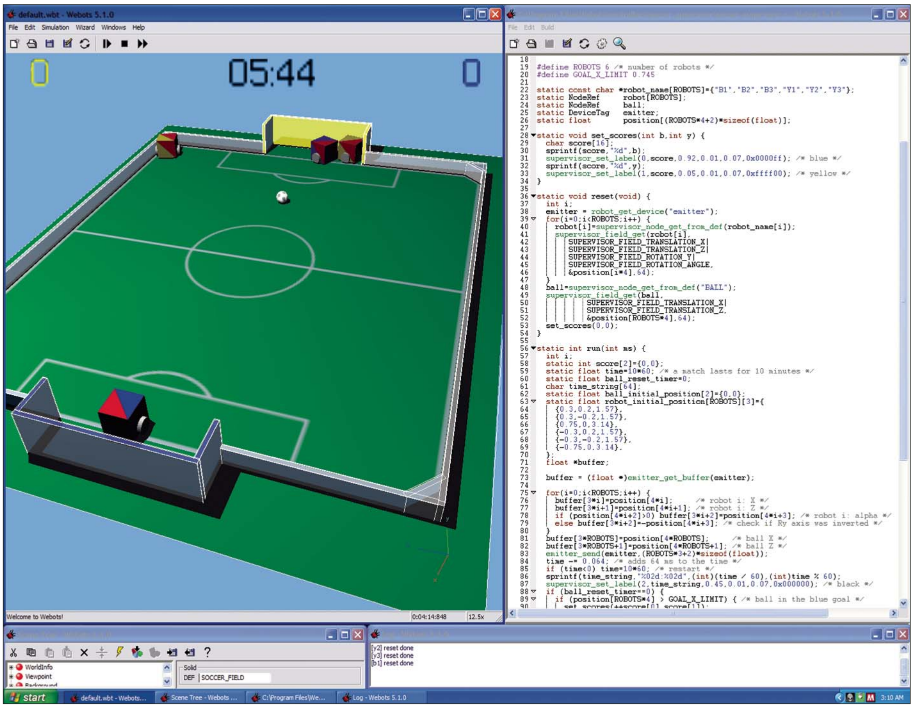


FIGURE 4. A multi-robot soccer simulation in the CyberBotics Ltd. Webots 5 simulation environment.

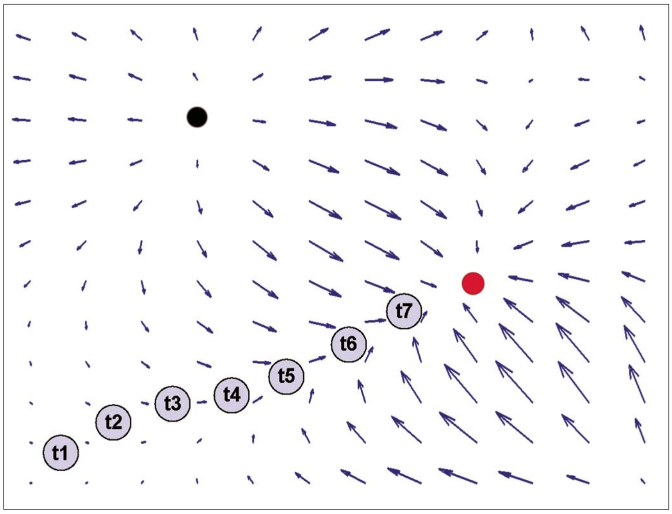


FIGURE 5. Path taken by a robot attracted to a goal (red) and repelled by an obstacle (black). Quiver display of potential field created in MatLab.

LEGO Mindstorm, Khepera, Kaola, and Hemission. Moreover, once the code is debugged and run in the simulated environment, it can be automatically downloaded and run on a real robot.

The utility of Webots isn't limited to the five predefined robots. With a bit of C/C++ programming, you can create simulations of virtually any robot configuration with the included tools. That said, the power and complexity of the Webots environment can be daunting to the uninitiated. The color-coded IDE, help files, and numerous examples help contain this complexity.

Webots 5 Standard is available for $240 as a download. Webots 5 Pro, which supports custom physics programming, a fast simulation mode, recording, and communications, and other supervisory functions, is about $2,900. A limited demo version is available for free download.

Potential Fields With MatLab and Simulink

Two of the most influential reactive architectures in robotics are subsumption and potential fields. The subsumption architecture — commonly used in hobby robot designs — is based on the ability of higher layer behavior modules to override or subsume the output from behavior modules in lower levels [2]. The potential fields architecture, in contrast, uses vectors to represent individual behaviors and vector summation to produce overall behavior patterns.

Using navigation as an example behavior, assume a robot may move from a repulsive field and move to an attractive field. If obstacles are represented by repulsive fields and goals by attractive fields, then the optimal path of a robot headed toward a goal can be computed by taking the sum of vectors at the location of the robot.

Following the example in Figure 5, the simulated robot starts in the lower left corner of the field at time t1. At time t2, the robot samples the environment and adjusts its trajectory based on the sum of local vectors. Robot velocity is adjusted so that it is proportional to arrow length. This process continues until the robot reaches the goal.

Potential fields are often represented as electric or magnetic fields drawn as quiver or contour plots. The specific representation of the underlying vectors is irrelevant, as long as it provides the user with an intuitive grasp of why specific routes are selected over others.

Figure 6 shows the path of a simulated robot past two obstacles (black) toward a goal (red), as computed by summing local field vectors. The contours surrounding the obstacles represent forces that repel the robot, while the contours surrounding the goal represent forces pulling the robot forward. Stated another way, the contour map is a topographical map in which obstacles are hills and the goal is the lowest point in a valley. The robot is simply traveling downhill, seeking the lowest point in the environment.

Figure 6 was created with a high-level technical computing environment, MatLab, Simulink, and an interactive 3D environment add-on, Virtual Reality Toolbox. MatLab (Matrix Laboratory) is a powerful, extensible mathematics environment that can be used to create simulated robots with sophisticated behaviors.

MatLab differs from C++ and other low-level, generic languages in that it provides a huge library of mathematical functions that can be applied to robotic simulation. For example, creating the colored contour lines around the poten-

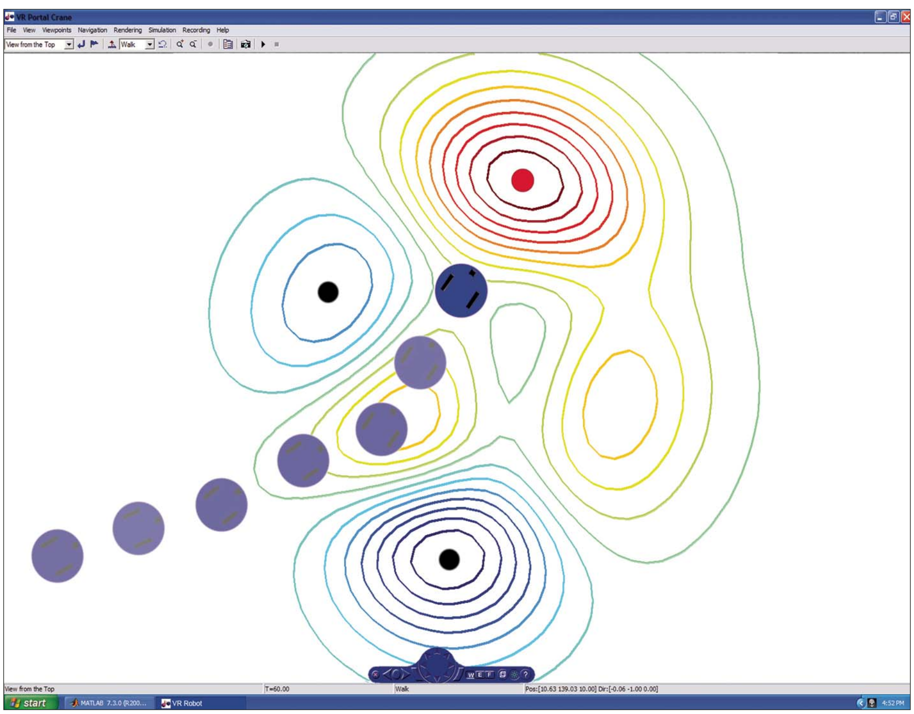


FIGURE 6. Robot simulation showing time-elapsed navigation using potential fields. Goal shown in red. Obstacles in black. Built with MatLab, Simulink, and the Virtual Reality Toolbox.

tial fields shown in Figure 6 involves little more than calling the contour function:

Contour(z,n)

where *z* is a matrix representing heights along the z-axis with respect to the x-y plane of the environment, and n is the number of contour levels. Similarly, vectors can be overlaid on the display with the quiver (as in archery) function:

Quiver(x,y,dx,dy)

where *x* and *y* are matrix values and *dx* and *dy* are the associated gradient values. The quiver function was used to create the vector field in Figure 5.

In addition to powerful functions, MatLab supports traditional programming features, including arithmetic operators, flow control, data structures, data types, and debugging utilities. Although MatLab is powerful, it does take some time to think of everything in terms of matrices.

Simulink is an interactive, graphical environment that can be used alone or with MatLab to create robot simulations. Simulink can be seamlessly linked with models created in MatLab and programmed by graphically connecting customizable clock libraries.

As with MatLab, Simulink can be extended with add-on toolboxes of functions. The Virtual Reality Toolbox extends MatLab and Simulink with virtual reality functions that can be used to control the position, rotation, and dimensions of the 3-D images defined in the virtual reality environment. The environment shown in Figure 6 was defined in the Virtual Reality Toolbox.

The family of MatLab and Simulink programs and add-on toolboxes can create any robot simulation imaginable — given time and expertise in the environment. Expect to spend several days to come up to speed in each environment. Furthermore, MatLab and Simulink are expensive — about $3,000 each for a single license ($500 each for academics). Tool-boxes are $200 for academics. Fortunately, the robotics research community has many MatLab/Simulink — compatible toolboxes in the public domain. Time-limited copies of MatLab/Simulink are also available for evaluation.

Waypoint Navigation With DarkBasic Professional

Pathfinding involves identifying an optimal path through an environment. One method of implementing pathfinding involves first identifying the complete route and then decomposing the route into segments. The end of each segment leading to the goal is called a waypoint. Waypoints can be thought of as subgoals along the way to the final goal.

Figure 7 illustrates elements of waypoint navigation in an interactive, 3D environment. The robot (depicted as a silver cone in the upper third of the simulated environment) follows the waypoints toward the goal. The goal can be repositioned with the mouse, and the robot will select the appropriate waypoints to reach the new goal position.

The key variables in the simulation include the obstacle boundary distances, the number of waypoints between the initial position and final goal, and underlying algorithm used to compute the optimum path. The most popular algorithm is a variation of the basic A* (A Star) graph search algorithm, an example of a best-first search. An important property of the A* algorithm is that it will always find a solution if there is one [3].

The robot simulation in Figure 7 was created with DarkBasic Professional, an entry-level game engine that provides a Basic wrapper around Microsoft's DirectX. Game engines are software components that handle activities such as rendering, AI, and physics, separate from the sounds, images, textures, and other media of a video game [4].

As a full-fledged Basic compiler, DarkBasic Professional can be used to create robot simulations with any number of AI behaviors. However, with the addition of the Dark AI extension pack, waypoint-based pathfinding, collaborative teams, pursuit and avoidance behaviors, and other AI behaviors can be implemented with simple function calls. Another advantage of using DarkBasic Professional or other game engine over Visual Basic or C++ is the ease with which 2D and 3D graphics and sounds can be integrated into the simulation. Furthermore, engine add-ons dedicated to AI and Physics make creating complex, realistic robot simulations relatively painless.

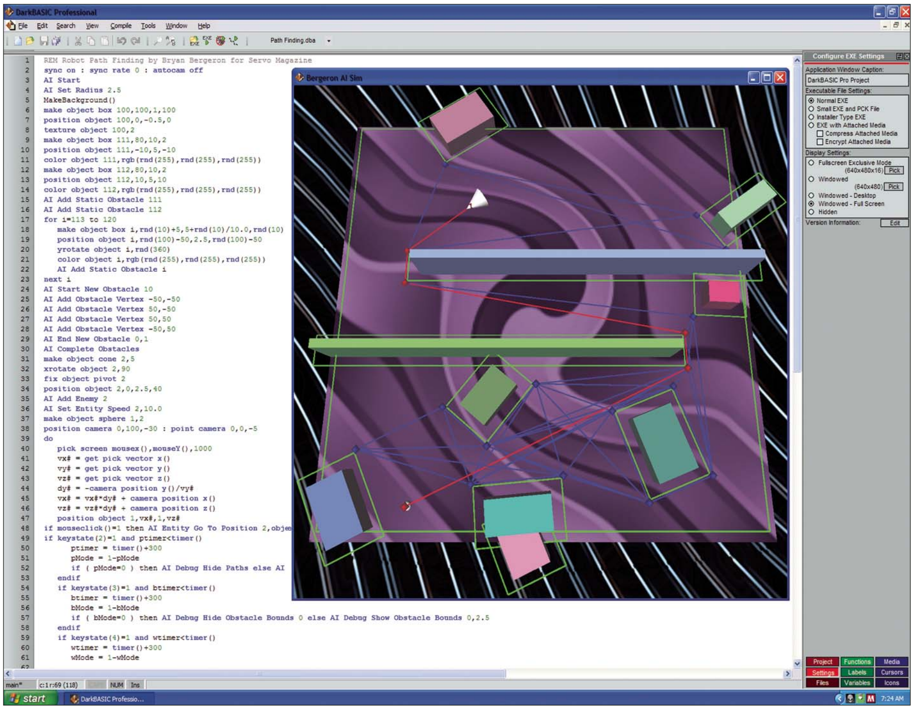


FIGURE 7. Simulated robot using waypoint navigation. Developed in DarkBasic Professional with the Dark AI. Silver cone: robot. Red lines: robot path. White lines: obstacle boundaries. Blue nodes and lines: waypoints and waypoint edges.

References

[1] Millington, I. (2006). *Artificial Intelligence for Games.* Boston, Morgan Kaufmann Publishers.

[2] Bergeron, B. (2006). *Developing Serious Games.* Boston, Charles River Media/Thompson.

[3] Choset, H., K. Lynch, et al. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations.* Cambridge, MIT Press.

[4] Thrun, S., W. Burgard, et al. (2005). *Probabilistic Robotics.* Cambridge, MIT Press.

[5] RoboCup 2006 (2006). **www.robocup2006.org**

[6] Murphy, R. R. (2000). *Introduction to AI Robotics.* Cambridge, MA, MIT Press.

DarkBasic Professional is available from The Game Developers for $99, and the AI engine is $45. There are no royalties or license needed to distribute the compiled applications. Furthermore, like the Basic used in MobotSim, DarkBasic can be manually ported to other Basic environments with little or no change. Another advantage of DarkBasic Professional is that an affordable physics engine is also available, and this engine supports the PhysX accelerator chip. (This will be discussed in the second article of this series.)

Reality Check

Robot simulation is a means of identifying problem areas and verifying that all variables are known before construction of a robot is begun. As an analysis tool, a robot simulation can help explain why certain events occur, identify inefficiencies, and determine whether specific modifications of a robot will compensate for or remove these inefficiencies.

However, the practical value of robot simulation isn't always obvious. For example, the robotics community is particularly concerned about the validation stage of the simulation development process. This so-called *correspondence problem* — the degree to which simulation results translate to real robots — will likely remain a topic of contention for some time. However, this hasn't impeded many researchers and roboticists from embracing robot simulation.

The acceptance of simulation as a means of developing and testing the behavioral algorithms is reflected in the Soccer Simulation League component of the annual RoboCup contests [5]. Furthermore, entire areas of robotic research — such as genetic algorithm-based evolutionary robotics — would be untenable without robot simulation. Evolutionary robotics involves running hundreds to thousands of robot simulations to arrive at optimal behavior [6].

From Here

Robotics — including the simulated variety — is a hands-on activity. At a minimum, consider experimenting with the time-limited free downloads of MobotSim or Webots. A quick search on the web will also reveal dozens of other commercial and open source robot simulation options. Many of the open source simulation tools require fluency in Linux and open source C/C++ compilers.

Two of the most popular open source robot simulations are the Stage and Gazebo simulations. Stage is a 2D simulation of multiple robots in an indoor environment, and Gazebo is a 3D simulation of multiple robots in a virtual outdoor world. Another compelling robot simulation is VSOC (virtual soccer), which allows you to train soccer players with genetic algorithms and control play through neural networks. All three open source robot simulations may be downloaded from **sourceforge.net**.

Out of the references listed here, *Murphy's Introduction to AI Robotics* is the most approachable introduction to AI behaviors. Although designed for the game developer, Millington's *Artificial Intelligence for Games* is an excellent source of behavior algorithms. The CD that accompanies the text provides C++ source code for dozens of behaviors, including the behaviors discussed here.

The next installment in this series will build upon the simulation techniques used for AI behaviors and focus on simulating the physics of the robot hardware platform. **SV**

Resources

DarkBasic Professional and Dark AI. The Game Creators.
www.TheGameCreators.com

MatLab/Simulink. The MathWorks.
www.mathworks.com

MobotSim. MobotSoft.
www.mobotsoft.com

Webots 5. CyberBotics, LTD.
www.cyberbotics.com

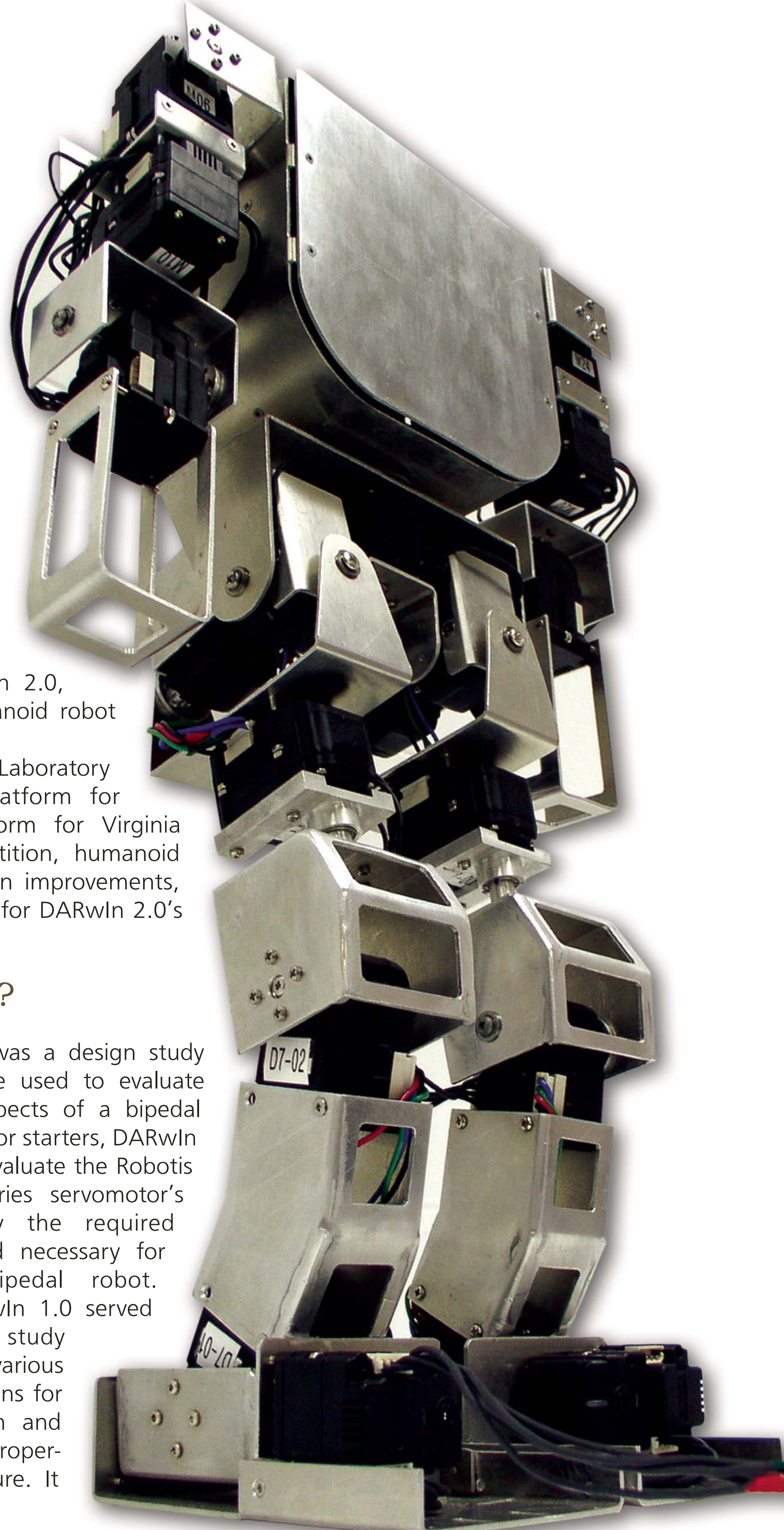
Player/Stage.
Playerstage.sourceforge.net

VSOC.
VSoc.sourceforge.net

DARwIn

PART 3: DARwIn 2.0: The Next Generation

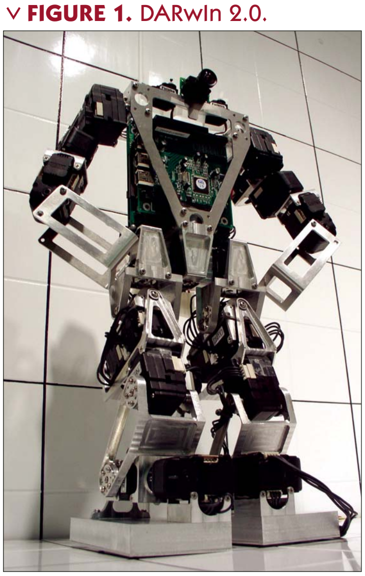
by: Karl Muecke, Patrick Cox, and Dennis Hong
RoMeLa (Robotics & Mechanisms Lab) at Virginia Tech;
www.me.vt.edu/romela



Last month, we detailed the inner workings of the first generation of DARwIn (Dynamic Anthropomorphic Robot with Intelligence) — a humanoid robot capable of bipedal walking and performing human-like motions. This month, we reveal our latest design: DARwIn 2.0, the next step in evolution of Virginia Tech's humanoid robot (Figure 1).

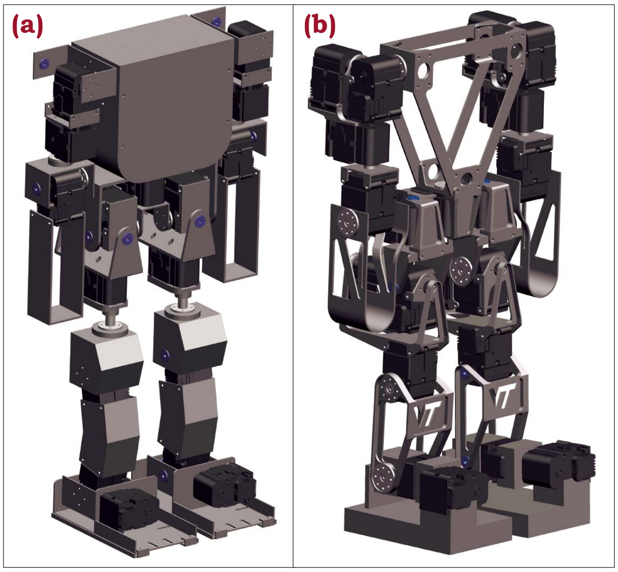
Developed at the Robotics & Mechanisms Laboratory (RoMeLa), DARwIn 2.0 is a new research platform for studying robot locomotion and also the platform for Virginia Tech's first entry to the 2007 RoboCup competition, humanoid division. This article will detail some of the design improvements, new features, and touch on some of the software for DARwIn 2.0's intelligence.

FIGURE 1. DARwIn 2.0.

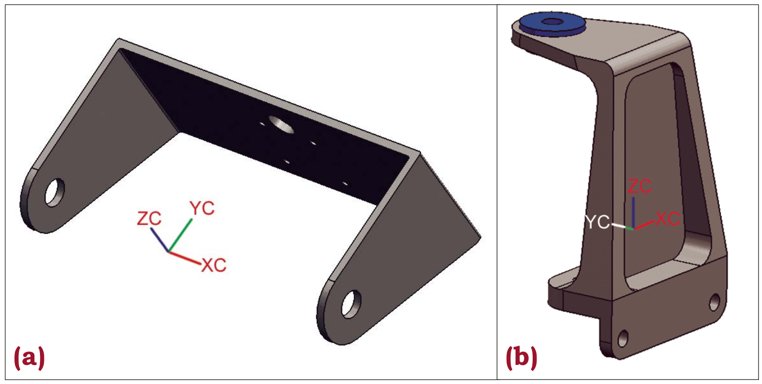


What's New?

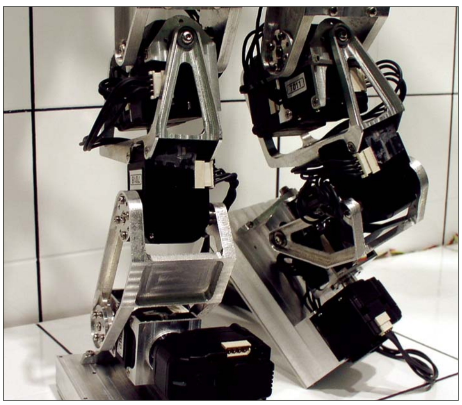
DARwIn 1.0 was a design study prototype that we used to evaluate various design aspects of a bipedal humanoid robot. For starters, DARwIn 1.0 was used to evaluate the Robotis Dynamixel DX series servomotor's ability to supply the required torque and speed necessary for a small-scale bipedal robot. In addition, DARwIn 1.0 served as a case study for investigating various motor configurations for a compact design and better kinematic properties of the structure. It



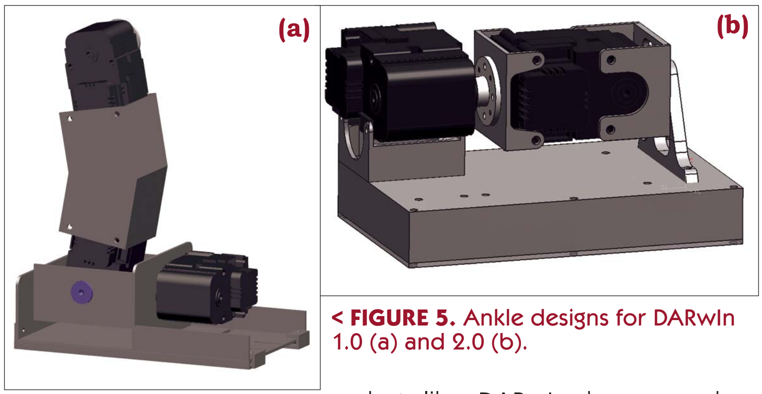
^ **FIGURE 2.** CAD rendering of DARwIn 1.0 (a) and the new DARwIn 2.0 (b).



^ **FIGURE 3.** Waist bracket of DARwIn 1.0 (a) and strengthened design of DARwIn 2.0 (b).



< **FIGURE 4.** Close-up of the new CNC-milled aluminum parts of the legs.



< **FIGURE 5.** Ankle designs for DARwIn 1.0 (a) and 2.0 (b).

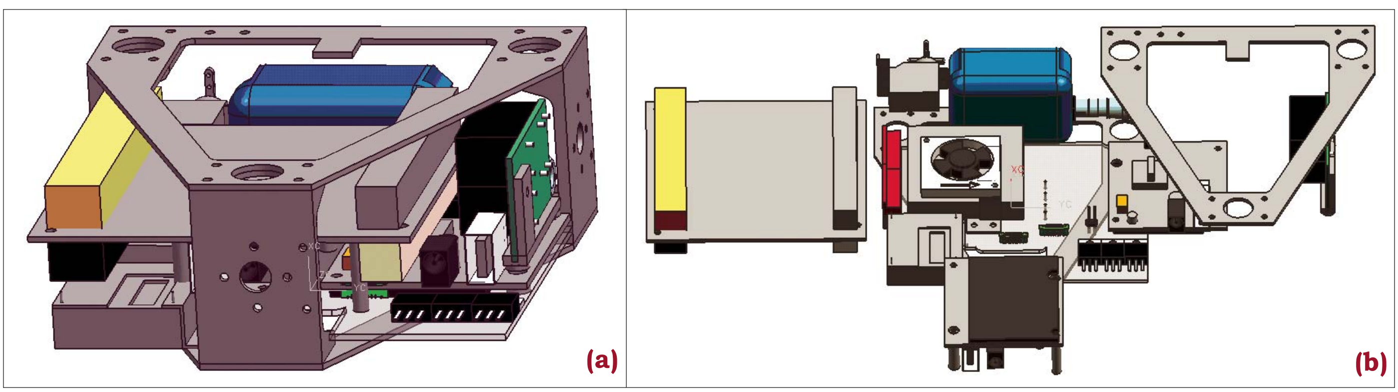
< **FIGURE 6.** Easy access to the lithium polymer batteries and lowering the center of gravity.

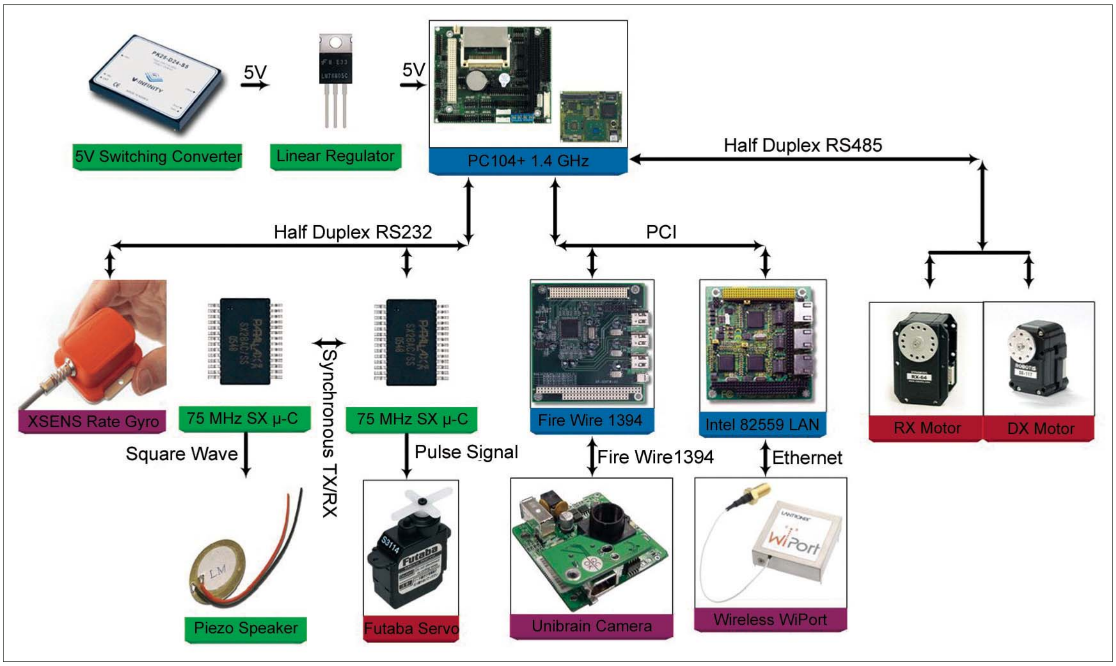
was also used as a test platform for testing simple walking gaits. An improvement based on the discoveries and design ideas from DARwIn 1.0, DARwIn 2.0 is a superior re-designed robot (Figure 2a and 2b).

Some areas of concern that developed from designing DARwIn 1.0 were the strength, stiffness, and weight of the links forming the joints and connecting the motors together. Stiffness and strength are a priority when designing a bipedal robot like DARwIn because when implementing analytically-generated walking gaits based on kinematics and dynamics, links that flex or bend will not be in the position and orientation you assumed when generating the actual movements for the gaits. Especially with the heavy weight of the robot and the long moment arms that generate large bending forces on the links, the stiffness of the links is a high priority in the design.

DARwIn 1.0 was fabricated chiefly using sheet metal bent into shapes as

v **FIGURE 7.** Electronic components in the chest (a) and an exploded view (b).





> **FIGURE 8.** Hardware architecture of DARwIn 2.0.

specified by CAD drawings. Figure 3a shows DARwIn 1.0's waist link — a U-shaped bracket with no other structural support to keep itself from deforming from bending. All of DARwIn 2.0's parts were milled out of aluminum using a CNC machine, making them smaller, lighter, and stiffer by incorporating design features — such as ribs — and optimizing the geometry for strength and weight, as shown in Figure 3b and 4. These features are very difficult to implement in bent sheet metal alone.

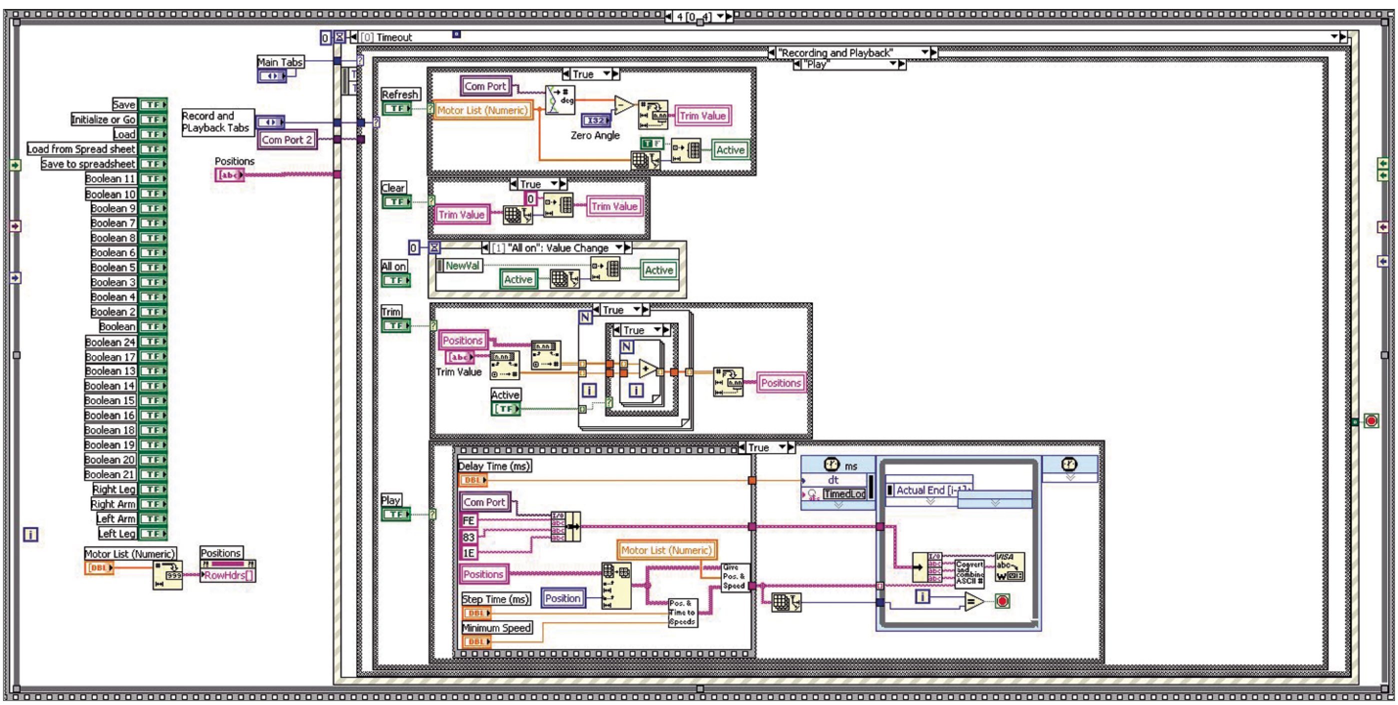
Some joints in DARwIn 1.0 had a limited range of motion due to the chosen motor configuration and the resulting motor mounts. One particular example is shown in Figure 5a. Due to the motor mounting plate, the shin was limited in forward motion. By placing both motors as shown for DARwIn 2.0 (Figure 5b), a larger range of motion for the shin link is possible since the motor plate is no longer present and the heel pivot is further away from the motor at the toe.

Additionally, we moved the lithium polymer batteries from the chest to the foot (Figure 6) to lower the center of gravity and to give easier access to them. Now the batteries can be easily slid out for recharging. This also allowed additional space in the chest to incorporate other electronics hardware.

The Brawn

We are using Robotis' Dynamixel servomotors, models DX-117. We are

∨ **FIGURE 9.** A screenshot of a LabVIEW block diagram.



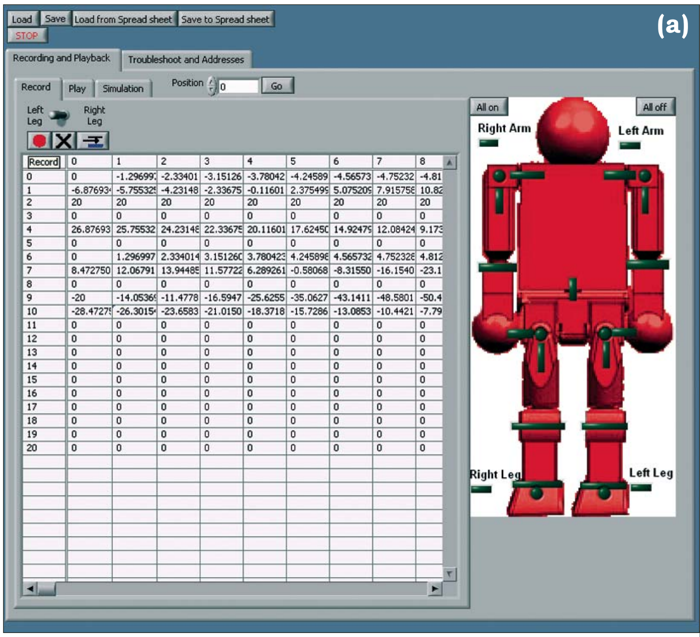
^ **FIGURE 10.** (a) A screenshot showing motor addresses read by LabVIEW. (b)-(d) Screenshots showing the graphical user interface used to access the motors' information.

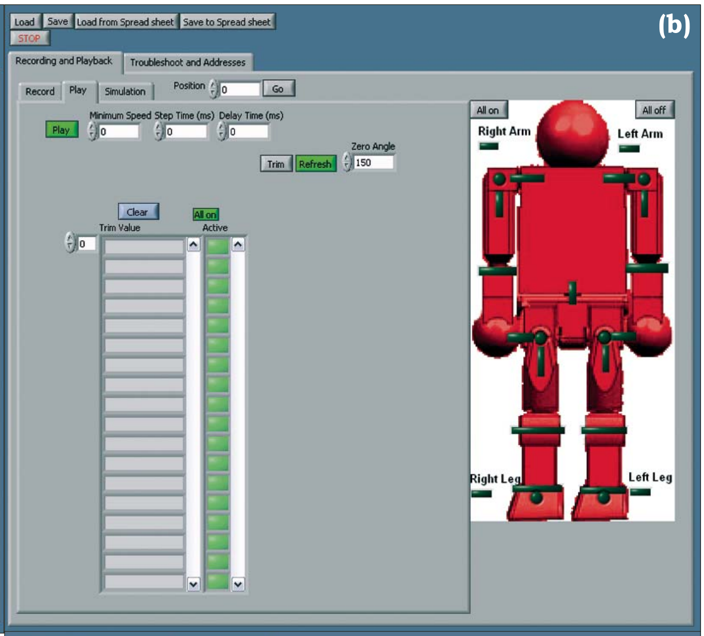
planning to use the new RX-64 motors for the joints which require heavy torque loads — such as the knee joint and the ankle joint — for future design modifications. The DX-117 has a maximum torque of 39 kg-cm and the RX-64 has a maximum torque of 64 kg-cm. Both have built-in position and speed controllers with feedback, and user control is through asynchronous RS485 serial communication with a daisy chain connection that makes wiring all the motors much cleaner and easier.

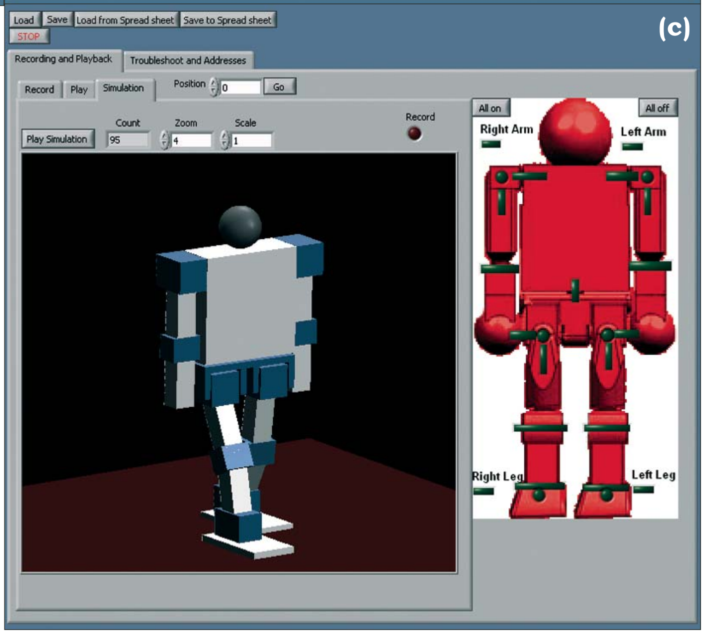
The Brains

Both DARwIn 1.0 and 2.0 use a PC/104+ computer as their brain. The PC/104+ board we use supports a 1.4 GHz Pentium M processor, which is used to control all of DARwIn's behavior functions, as well as for processing information from the various sensors and to interpret them for mapping and localization. The PC/104+ board is housed inside the metal shell of DARwIn 1.0's body. For DARwIn 2.0, the board will be placed on its back like a backpack to give DARwIn the appearance of a slimmer chest.

Included with the PC/104+ board are additional electronics for power management, communication, and interfacing with the various sub systems. The extra electronics shown in Figures 7a and 7b serve the purpose of regulating power to the computer and sensors, providing a controller for the two Futaba miniature hobby servos







used in DARwIn's pan and tilt camera unit in the head, checking the status of the lithium polymer batteries, enabling Wi-Fi communication, and interfacing to the various sensors including rate gyros from Xsens. These components were also packed in DARwIn 1.0's chest, but in DARwIn 2.0, these are separated from the computer to produce a modular design where the electronics are easily removable. The components, such as the IEEE 1394 card, camera board, fans, etc., are layered on top of one another in order to fit them in the small volume of DARwIn 2.0's chest, while allowing air flow for cooling.

For simplicity, all these components are mounted onto the circuit board which will be providing power regulation and servo control. In addition, although not shown, all the necessary wire connections between components will be etched onto the circuit board, which should reduce the overall weight of the electronics. In Figure 8, a full hardware architecture diagram is shown detailing the information bus between all the components.

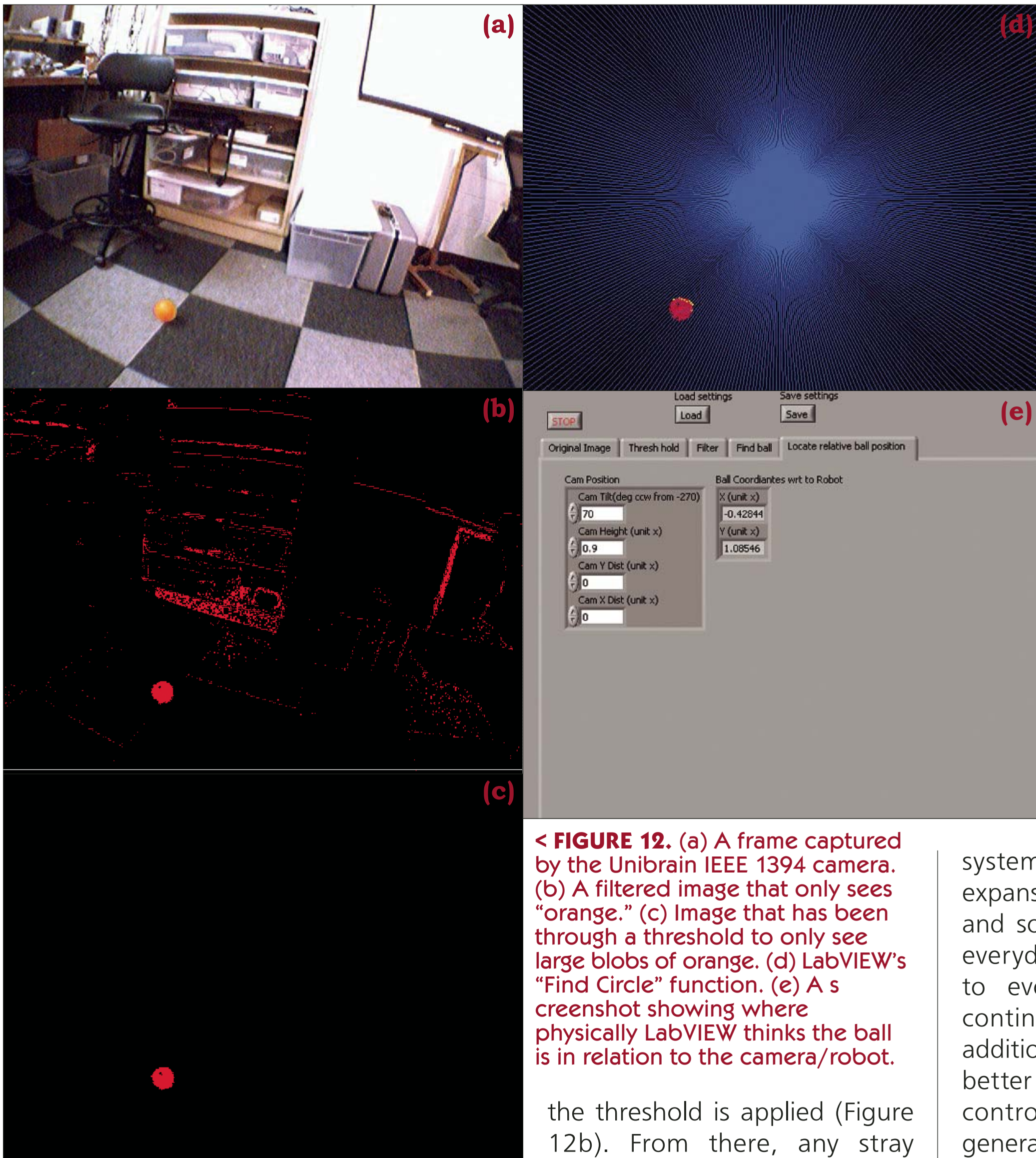
A Quick Introduction to LabVIEW

For those of you unfamiliar with LabVIEW, it can best be explained with a picture (Figure 9). Our LabVIEW program is used to read 36 properties from 21-24 motors (Figure 10a), give a graphical user interface for every motor (Figure 10b-10d), record joint positions (Figure 11a), play back generated gaits (Figure 11b), display a 3D OpenGL model of the robot (Figure 11c), and serve as a way of controlling the entire robotic system including mapping, localization, and behaviors.

> FIGURE 11. (a) A screenshot showing a user recording robot poses. (b) A screenshot showing the playback interface to make the robot walk/move. (c) A screenshot showing the OpenGL model animation of the robot.

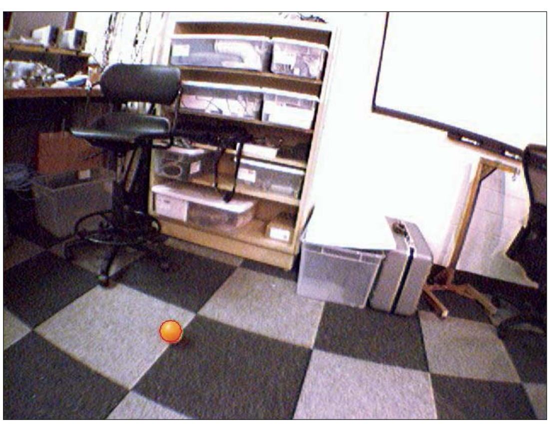
Figure 9 is a screenshot of our latest LabVIEW code used to run everything mentioned above (case structures hide some code). Instead of going through hundreds of lines of code, you need only look at a picture for your programming as LabVIEW uses a graphical way of coding.

LabVIEW comes with many built-in features that make vision processing a simple task. Figure 12a shows an image captured by the IEEE 1394 Unibrain Fire-i camera which is mounted on a pan and tilt unit on the head that serves as the eyes for DARwIn. LabVIEW has its own set of drivers for IEEE 1394 cameras so that when you put your code on a PC/104+ board, you don't need to worry about the drivers.



< FIGURE 12. (a) A frame captured by the Unibrain IEEE 1394 camera. (b) A filtered image that only sees "orange." (c) Image that has been through a threshold to only see large blobs of orange. (d) LabVIEW's "Find Circle" function. (e) A s creenshot showing where physically LabVIEW thinks the ball is in relation to the camera/robot.

After LabVIEW captures the first frame, it performs a basic color threshold to the image. Only pixels that are bright orange remain after the threshold is applied (Figure 12b). From there, any stray pixels are filtered out (Figure 12c). With the cleaned up image of only a ball in the frame, LabVIEW runs a "Find Circle" routine that finds the best circle in the image (Figure 12d) to locate the ball. The program overlays the result on the original picture to show where it thinks the ball is (Figure 13). Just knowing where the ball is in the picture is not very useful, so we also use the kinematic model of the robot with the joint position info from the motors and orientation information from the rate gyros to find out where the ball is with respect to the robot (Figure 12e). In a similar manner, the robot figures out its position in the playing field by using known markers as the goal posts.



< FIGURE 13. Original image captured with a circle overlay of where LabVIEW thinks the ball is.

Conclusion: See you at RoboCup 2007!

With a vastly improved mechanical design, new system architecture for future expansion, better computing power, and software that is being improved everyday, we plan on continuing to evolve DARwIn. We will be continuously adding and testing additional software components for better vision recognition, behavior control, path planning, and gait generation.

For RoboCup 2007, most of the innovations will be seen in software and electronics as we prepare to battle it out against the best bipedal robots in the world. DARwIn is a robot that truly follows RoMeLa's philosophy and motto: "Robot Evolution by Intelligent Design." **SV**

Thank Yous

I would like to say thank you to the 2006-2007 senior design team members that designed and built DARwIn 2.0 (Abhijit Chakraborty, Marilyn Duncan, Andrew Lynch, Robert Mayo, Ryan Misjan, Laurence O'Neil, Bill Pannell, and Eric Steinberg) and to our advisor Prof. Dennis Hong and graduate advisor Karl Muecke.

THE FLAMING LOTUS GIRLS

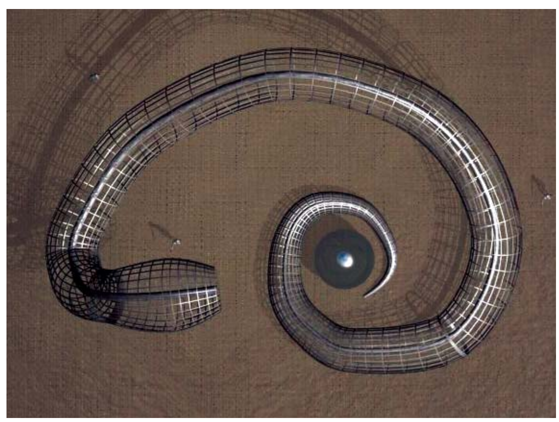
by Steven Kirk Nelson

AND THE SERPENT MOTHER

I first met Charlie Gadeken in May, 2004. My father and I went to the Box Shop in San Francisco, CA, for a Power Tool Drag Racer test and build day. While wandering around the yard, I noticed a large collection of what had to be flame-producing technology and art. I asked Charlie, "What the heck is all of this stuff for?" Charlie told me that this was the home for The Flaming Lotus Girls. He then explained to me that there was a group of women and men that worked together to build LARGE art projects for events like the Burning Man.

Little did I know, that this was the beginning of a relationship with the ultimate group of industrial fire artists, free thinkers, and builders I have ever had the privilege to work with. In May, 2006 we did the Power Tool Drag Races again. Of course, I met up with Charlie and mentioned that I needed to do a little welding on one of my racers. He told me to take it to the Box Shop.

The Box Shop is located at 951 Hudson Ave., near Hunters Point in San Francisco. The Shop is basically a large yard filled with shipping containers that are available for rent. They are usually used to store materials for folks that like to build mechanical things like art cars, intricate sculptures of industrial art, and assorted pyrotechnic devices. It is a magical place, filled with imagination, fabrication, multiple personalities (sometimes in the same person), periodic flashes of fire, plasma arcs, and other assorted mayhem. The Box Shop contains a large welding and fabrication facility, and teaches classes in welding, metal shaping, and art fabrication for comparatively modest fees. It is also the home



Disclaimer

Building fire art can be very dangerous. The skills in plumbing, welding, fire control, and fire safety demonstrated by the Flaming Lotus Girls were developed over years of consulting with experts in many different industrial disciplines and a lot of learning from mistakes. In no way should you try and/or duplicate their efforts at home. The Flaming Lotus Girls, the author, and SERVO Magazine bear no responsibility for your efforts or mistakes made using this technology.

Lynn Bryant taking hammer marks out of the copper egg shell on the English wheel. Lynn and her team were also responsible for the construction and design of the Egg. Lynn told me that she's real good at making wagon wheels and elbow macaroni after bending all of the steel tubing used in the egg's framework.

Rebecca "Hot Metal" Anders and Charlie Gadeken weld yet another stainless steel vertebrae together. The respirators are worn to protect the Girls from the dust, gasses, and nasty chromium by-products that are produced when grinding or welding stainless steel. It gives you a bad headache. The manufacturer painted them pink for some reason. Although, hot pink is the official color of any tool or part claimed (often by convenience) by the Girls at the Box Shop.

of the Flaming Lotus Girls.

The Serpent Mother

That's where I saw *it*! On one of the work tables was the first prototype of a vertebra made out of mild steel. Taped to the walls of the Box Shop was an endless collection of "to do" lists, parts lists, drawings, schematics, plumbing diagrams, etc. Also, there was a small detailed model of the Serpent Mother with its tail wrapped around an egg. When I saw Charlie at the races, I asked him "How big is this thing?" He said, "about 165 feet long." I asked him how long it took to make that vertebra. He said "About two days, but the real ones will be made from stainless steel." I then asked how many were going to be made, to which he replied "about 45 of the large vertebrae and about 30 smaller ones, plus four or five simpler ones to cover the hydraulics for the neck going to the head."

"You're going to cut, bend, shape, and weld all of that stainless steel sheet metal, build a heavy pipe frame, add 41 propane poofers, a five-axis hydraulic powered head, plumb this machine, and build the electronic controls and write the control software in less than three months?" I asked in astonishment. "Yep," Charlie replied. I told him "you're crazy man!" and he just smiled.

Fortunately, Charlie has a lot of talented, motivated artists, engineers, structural welders, metal sculptors, pyro plumbers, electronic engineers, and programmers to help make the Lotus Girls dreams come true. The design drawings, parts lists, and budget (plus a "miscellaneous lipstick fund") were written up in a proposal and submitted to the Burning Man in January. Most of the funding for this project came from a $60,000 grant from Burning Man which didn't arrive until May. Nothing like a little time pressure to get the Girls motivated.

The Build

The next time I saw the Lotus Girls was in early June at The Fire Arts festival in Oakland, CA. They had already built the head, the hydraulic jaw mechanism, and the mounting for the first three hydraulic cylinders, along with about 20 feet of the spine. There were five poofers and the vertebrae, plus the teeth were burning. I couldn't believe it! They had a fully-working first section with fire and the basic hydraulic head movement done in about one month.

This took a maximum effort from all of the Girls and they worked on the project day and night whenever they could get time to go to the shop. (Keep in mind that these folks have jobs and lives like everyone else.) During this build cycle, many of their lives were put on hold and they were less than one third done. This pattern of self-sacrifice would continue for another two months of very hard work.

After the show, they put me to work disassembling the machine and loading it on the truck. The Girls have a term called VHTs (that's Lotus-ese for VERY HEAVY THINGS). Lifting 250 to 300 lb sections of the Serpent Mother into a truck is definitely "quality" time. (I never have figured out why folks think that fat guys — like me — like to lift heavy things.)

At this point, there was a lot of concern that they may not finish the project. So, they put out a call for help on the Internet. I started going to San Francisco on the weekends.

Working at the Box Shop with the girls is a unique experience. As a man, you're supposed to assist the Girls and leave your male attitude at the door. The Girls say "It takes a great man to be a Flaming Lotus Girl." The Girls are in charge of the project, so you have to ask them what needs to be need done and then you are given a task to perform. If Pouneh Mortazavi (Shop Leader) or another Girl sees that you're not busy, she will ask if you have ever cut pipe before. If you say no, she will

Annealing copper with a big torch to make it soft for shaping. Once shaped, the metal is then hammered until it hardens to hold its shape and then it is soldered together to make larger pieces.

tell you to go grab a piece of pipe and bring it over to the saw so she can teach you how. After you cut your first piece to length, she'll look at it and tell you "that's a good job, now cut 45 more like it." (This is how your days usually go at the Box Shop. You never know what you will be doing, *but you will be doing something*.)

I should also mention that the Lotus Girls have a unique way of solving problems. They discuss their next step in groups and every person's idea is explored and possibly tested until a solution is found that is both easy and practical. Having worked in shops usually with an Alpha male telling me exactly what to do and when to do it, I found this procedural difference to be both interesting and a beautiful thing to witness and be a part of.

The majority of the Girls at the Box Shop may have never cut, drilled, shaped, ground, heated, or welded metal or created endless feet of plumbing before they joined this group. It is the goal of The Flaming Lotus Girls to empower women by teaching them skills in metal work and industrial fire art. Hopefully, they will take their new skills and abilities and continue to create their own art, as well as help out with the group's projects. The motto of the Flaming Lotus Girls hangs on the wall in the Box shop. It simply says, WE CAN DO IT!

The Girls use several types of hand drawings, blueprints, schematics, and CAD (Cardboard Aided Design) drawings. and Plywood Aided Design are used when building the templates for metal cutting. Also, steel jigs are used to hold the parts in place for tack and finish welding.

The Vertebra

Building the vertebra took hundreds of hours and lots of people. Each one is made from five pieces of stainless steel that is plasma cut using wooden guide templates. Then it is hand-hammered with a ball peen hammer on a steel shot bean bag to create the curves. The edges are rolled with a hammer over the head of a railroad spike mounted in a vice. The hammer marks are removed with a pneumatic hammer and the English wheel. Parts are curved by running them through a roller, then they are mounted in a jig and then tack-welded and hammered some more.

Following that, they are seam-welded with a tri-mix shielding gas and a MIG welder using 304 stainless steel wire. Next, the welds are ground smooth, welded again, ground some more, then polished. Countless hammer strikes and 12-hour work days make this all possible. You really wouldn't want to arm wrestle with the women and men that pulled off this little part of the project.

The Poofers

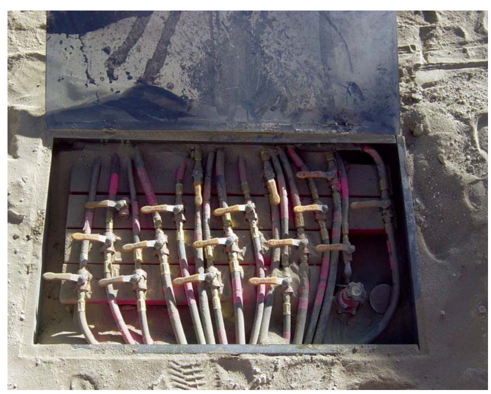
Fuel is provided to the Mother Serpent through two separate systems: the 'continuous flame' or burners running off two 88 gallon liquid feed propane tanks leading into a vaporizer. These devices boil the propane from the main storage tank and prevent the tank from getting cold and freezing the propane. (The freezing effect comes from the high fuel flow rates and a subsequent pressure drop created when running lots of burners or poofing.)

Without the vaporizers, the Serpent Mother wouldn't have worked for long. The vaporizer sucks liquid propane from the tanks and into a chamber where the liquid is vaporized, and the pressure created by the vaporization pushes the propane through three 1/2-inch feed pipes underground. The pipes split again into 1/4-inch hose before it gets to the ball valve farm. Ignition is provided by a horizontal burner that is made from 1/2-inch steel pipe about 12 inches long.

The burners are fueled from the fuel depot and their gas flow is controlled by ball valves. The multitude of ball valves was buried underground in an enclosure near the egg in the center of the sculpture. This part of the control system is called the Ball Valve Farm.

Once the valves were opened, fuel could flow to their respective burners. A pressure regulator was used at the

Every kid should have the chance to fire a flame thrower on a giant snake from time to time.



Fuel lines, wiring, and, of course, poofers!

fuel depot vaporizer to adjust the rate of flow to the burners. Some of the burners on the tail section were further controlled to reduce the size of the flames that were close to the folks standing next to them. The burner pipe is capped on one end and has several small 1/16-inch holes drilled in it to act as fuel jets. The burner pipe is covered with stainless steel wool to diffuse the fuel across the burner.

The steel wool also makes the burner mostly wind-proof. The burners were electronically lit using cannibalized stun guns. The electronics team mentioned to me that they got some very strange looks from the folks at the electronics store when they ordered 50 stun guns! Once the burners were lit, the stun guns were turned off.

The 41 propane poofers that are placed on the spine of the Serpent Mother are basically rather large (momentary) flame throwers. Propane was fed from a large 250 gallon tank to another vaporizer. The fuel from the vaporizer was plumbed underground through four 1/2-inch ball valves and hoses to fill the fire extinguishers that act as expansion tanks and allow the fuel to vaporize some more and collect as the tanks fill. The fuel in the expansion tanks is fed to a 120 VAC, electrically-controlled normally-closed gas valve through a 1/4-inch steel pipe. When the valve is opened, the gas in the tank is released almost instantly. POOF!!

The Poofers will each produce about eight-foot-high fire balls for about one to two seconds. One of the interactive features of this beasty allows the kids viewing the machine to push the 41 manual buttons mounted on the ribs for controlling the poofers on demand. Every kid should have the chance to fire a flame thrower on a giant snake from time to time.

Plumbing

The plumbing team (headed by Caroline Miller and Rosa Anna Defilippis) had the daunting task of cutting, fitting, sealing, and testing about a billion feet (maybe a bit less) of LPG hose, steel gas pipe, steel and brass fittings, and a ton of valves and mounting hardware. Actually, there are hundreds of connections and many folks spent time working with these components. Imagine spending several months of your life threading pipe and dreaming about plumbing nightmares and gas leaks.

You might find it difficult to understand the dedication or obsession of this team, but their work speaks for itself. Without the Pyro Princesses, the Serpent Mother would have been a nice piece of sculpture. Because of the plumbing team, the Serpent Mother is a warm and awe-inspiring interactive experience for all who witness it in operation.

Control System

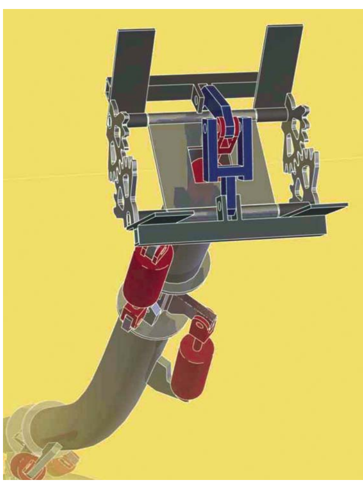
The poofers also have a computer control system that allows an operator to program firing sequences from a laptop computer. Jessica Hobbes managed the electronics team. Rich Humphrey assembled the innards of the fire controller boxes, but the boxes themselves were built by many, many hands. Tad Rollow wrote the AVR firmware that makes the boxes go. Lee Chubb wrote the computer interface software. It's a MAX/MSP patch. (Max/MSP is a graphical environment for music, audio, and multimedia. For more information about this software check out: **www.creativesynth.com/MAXMSP/maxmspmain.html**)

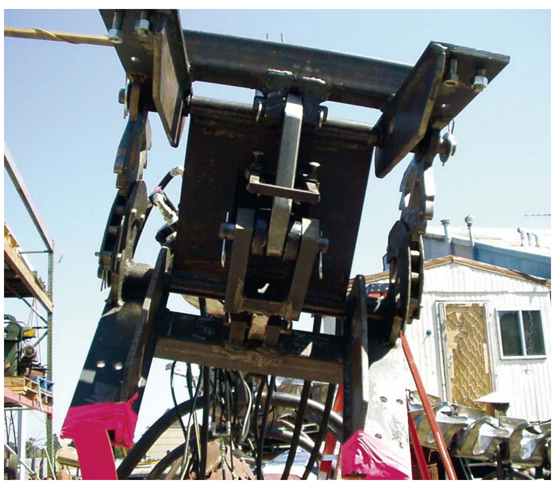
The interface that Lee created is really very easy to use and fun to play with, as well. Point and click poofing ... now that's an innovation in fire art.

The fire control boxes are ammunition boxes (painted hot pink, of course) with six 30-amp sealed mechanical relays inside. There are six outlets on the outside of each box that are independently controllable. Each box can hear all the traffic on the line. They only react for their own address. The data travels at 19200 baud. A signal has to be constantly sent to each poofer control or the poofers will shut down. This was an important safety feature.

You can also clone boxes, having more than one listen on an address. They are also completely overridden by the manual button boxes. The manual buttons are wired across the relay contacts so that a total computer failure would still provide us with a manual snake.

The brain is an Atmel ATMEGA8 programmed with AVR-GCC. Each box has an address, which is set by DIP switches inside the box. They are all listening on an RS-485 network, run over XLR microphone cables. It's terminated

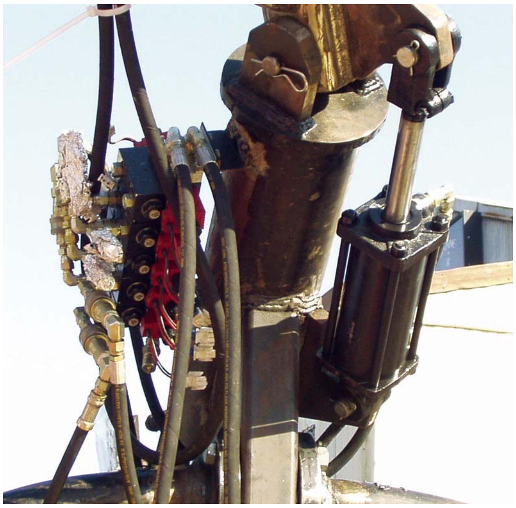


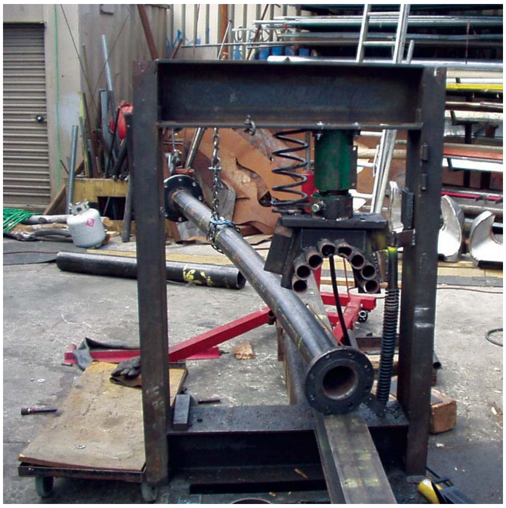


at the end by a 120 ohm resistor in an XLR plug.

There were some issues with noise in the power and control systems. We were switching coil valves with mechanical relays, so there was a lot of noise generated when we turned them on and off. There was also some noise coming from the stun gun igniters, as well as all the other problems you have out there, like generator noise and static.

The stun guns were only used to light the burners and then they were shut off. The electric gas valves on each poofer had a RC snubber circuit mounted across each coil. The nice thing about using embedded microcontrollers is, if they lock up because of noise, you can unplug them and they re-boot. They actually worked flawlessly. There was also a LED lighting effect mounted in the vertebrae. The LEDs could be flashed in multiple patterns to add even more eye-catching illumination to the sculpture.





Hydraulics

Mike Prados (P.E.) designed the mechanical and hydraulic system for the Serpent's head and neck.

Manual control was achieved with a set of mechanical relays driven by a joystick box using three separate arcade-quality joysticks. Limiting switches were attached to the hydraulic mounts and pivots to keep the cylinder from exceeding the mechanical limits of the sculpture. This kept the Serpent from eating itself.

Future control will be with an Atmel Atmega16 microcontroller, with potentiometers attached to the hydraulic mounts for position feedback.

Hydraulics for the Serpent's Head

Specs of the hydraulics include:

- 24,000 lbs peak force capability at 2500 PSI

- Hydraulic power pack pumps 1.3 GPM at 2,000 psi (a bit less at 2,500 psi peak)

- Power pack is a 2 HP electric pump, powered by 240 VAC

- Five double-acting cylinders at 3.5 inch bore, six-inch stroke

- Approximately 120′ of 3/8″ steel braided hydraulic hose

- Four CNC plasma cut gears, from 1/2 inch stainless steel plate

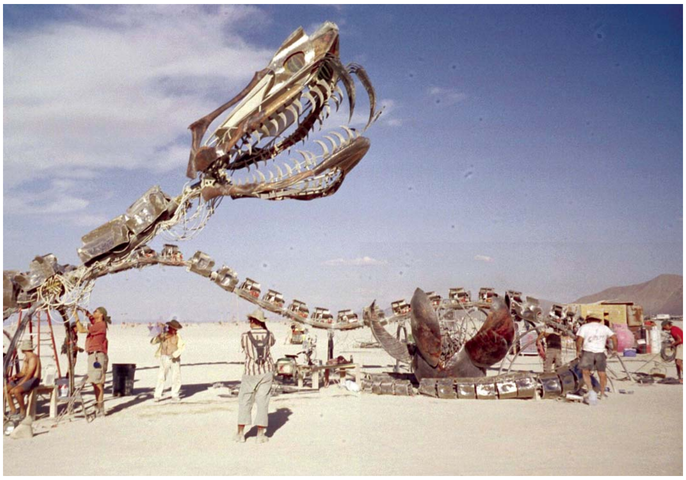
- Teflon coated, steel backed bronze bushings on the lower joints, rated at 36,000 lbs load

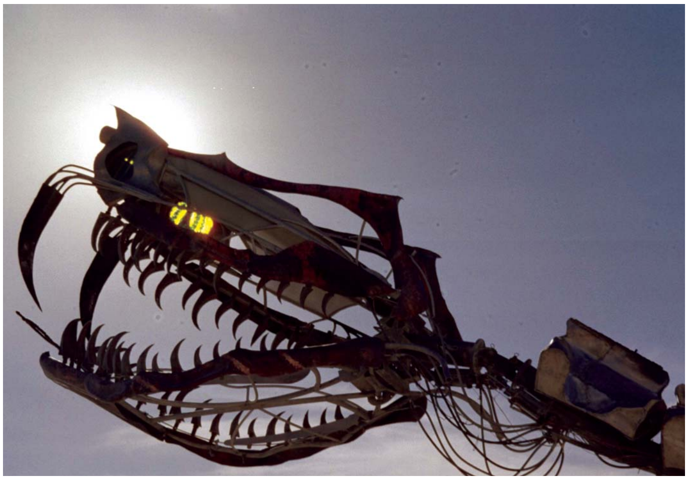
The Spine

The spine of the Serpent Mother is very massive. Even the flanges that couple the pipe sections together are about one inch thick steel. Spine specs include:

- 168 feet uncoiled length (this includes the neck with the head)

- Twenty eight-foot segments
  - Eight segments, six-inch diameter pipe
  - Six segments, four-inch diameter pipe





triangular stainless steel teeth of various lengths. The teeth were first plasma cut from sheet metal in three pieces and then welded together to form a hollow triangle. Intricate slots were plasma cut in the sides of the teeth to allow propane to flow from them. The teeth are mounted to a stainless steel fuel line. Propane is fed through this line, and escapes through 1/16-inch holes under the nipple that supports each tooth. The nipples have holes drilled in them to act as air correctors.

- – Six segments, two-inch diameter pipe

- Archway, about 16 feet tall

- Eight-foot length of the six-inch pipe weighs 180 lbs (before the vertebrae, flanges, plumbing, etc.)

Steve Monahan was responsible for most of the primary bending of the pipe on a custom-built, hydraulic powered pipe bending machine. Steve also did most of the structural welding since he's a certified welder and these parts had to be very strong for safety. There is also a ladder structure that the support ribs of the serpent connect to. The ladder structure is buried underground and is required to provide stability for the spine.

In talking to Michael Prados (our mechanical engineer), he told me that several factors were considered when the dimensions for the spine were developed, including the open spans, the weight of the components, and even wind loading from the high velocity desert winds common in northern Nevada.

The Head

The Serpent's head is the size of a small automobile. It is built in two parts forming the upper and lower jaws. The support frame is made from aluminum tubing custom-formed and welded.

There are 64 hand-made curved triangular stainless steel teeth …

Basically, this arrangement allows the teeth to work similar to a propane torch and produces a blue flame. "Hot Metal" Anders did quite a bit of testing and redesigning of this system. There was some discussion about making a blue flame and the intense heat produced, and its possible annealing (softening) effect on the aluminum structure. In testing with a infrared thermometer, it was found to heat the aluminum to about 350° F, which proved to be acceptable.

The two large fangs were fed with a separate fuel line and also burn with a yellow to blue flame. Of course, there

Introducing ... The Flaming Lotus Girls

Aimee Eade	David Ellsworth	Kiki Pettit Lani	Rich Humphrey
Aly HeinEric Stahl	Epona and Phil	Laura Kimpton	Rosa Anna Defilippis
Angela Knowles	Eric Smith	Lee Chubb	Sara Peyrot
Annie Geluardi	Flare Gaspo	Lee Sonko	Scott Cotner
Ariel & Jon Spear	Geoff Leland	Liam McNamara	Scott Sparky Bartlett
Baba Frey	Gole Mawaz-Khan	Lynn Bryant	Shanon
B'anna Federico	Hazmatt Snyder	Mark Farrier	Sharon Burke
Brent Coons	India Farrier	Marlies Tallman	Shawna Shandrick
Carly Perez	Jack Schroll	Mary Newsom	Simone Davalos
Caroline Miller	Jacqulynn Schmitz	Matt Cline	Simone Sigrid Marticke
Carson Best	James Stauffer	Michael & Lorelei	Stella Rubenstein
Catherine Lynch	Jen Clemente	Michael Curry	Steve Monahan
Cathy Lynch	Jeremy Travis	Michael Prados	Steve Nelson
Cecelia Camenga	Jessica Bruder	Michelle Palmer	Steven T. Jones
Charles J. Gallagher	Jessica Hobbs	Moira Mcnamara	Sue Duesberg
Charlie Gadeken	Jill Manthei	Naemi Frey	Suzun Hughes
Charlotte Sanford	Joe Romano	Nick	Tad Rollow
Cheryl Fralick	John Berens	Nicola Ginzler	Tamara Li
Colinne Hemrich	John DeVenezia	Olivia Sawi	Tasha Berg
Cory	John Wilson	Oona Squire	Tori Tait
Olivier Bonin	Jon Foote	Paul	Vanessa Montiel Waschka
Dan DasMann	Jordana Joseph	Phil Spitler	Wendy Blackburn
Dan Ramsauer	Josh Hunter Judy Castro	Pouneh Mortazavi	Will Flare
Dave Best	Karen Cusolito	Ray	Xanat
Dave X	Kezia Zichichi	Rebecca (Hot metal) Anders	Yasmin Mawaz-Khan

is also a forked tongue that shoots fire. The large green eyes are made from glass custom poured and formed by Peggy Wilson. The eyes are illuminated with a green laser beam. The head is fleshed out with hand-formed and hammered copper sheet.

The Egg

The tail of the Serpent Mother coils around the Egg. The Egg frame is made from steel tubing that was hand curved. The frame is covered in hand-formed copper sheet. There are five sections that open to reveal the methanol-fueled flame system. This system uses nitrogen gas to pressurize a storage tank filled with liquid methanol. The pressurized liquid is fed through underground hoses to five electrically controlled valves. Each valve can be individually fired.

The high pressure liquid is sprayed through nozzles across a propane burner and it ignites. Boric acid is mixed with the methanol to provide a green flame. When firing the Egg, no one was allowed to stand within a 150-foot radius of the sculpture. Some folks mentioned to me that they felt rain drops. I told them, "It wasn't water you felt falling from the sky, it was un-burned fuel."

Once I mentioned this fact, most folks understood the reason for the safety perimeter. The nightly fuel consumption for the Serpent Mother at Burning Man was about 500 gallons of propane and about 30 gallons of methanol (when the Egg was demonstrated).

Closing Thoughts

Ya know, if a 260 lb, redneck garage builder man can learn humility and understanding in an industrial and yet nurturing shop environment, all the while working with talented and amazing women and men, then maybe, just maybe, there's hope for us all. I've done a lot of things in my lifetime, but I will always cherish that special day when I finally became a Girl. (Well, a Flaming Lotus Girl, that is.)

It is impossible to list all of the individual work provided by all of the members of the Flaming Lotus Girls. This interactive, animatronic flaming sculpture is definitely a team effort and hopefully it will inspire others to think big. In my opinion, the Girls rock! SV

Websites

www.flaminglotus.com/

www.teamkiss.com

www.qbox.org/

Videos

- About the Girls
  www.madnomad.net/film/flg/
- About the Serpent
  www.madnomad.net/film/snake/
- About the Burning Man
  www.madnomad.net/film/bm/

Seeing With OpenCV

Finding Faces in Images

by Robin Hewitt

PART 2

Last month's article in this series introduced OpenCV — Intel's free, open-source computer vision library for C/C++ programmers. It covered the basics — downloading and installing OpenCV, reading and writing image files, capturing video, and working with the IplImage data structure.

This month, I'll show you how to use OpenCV to detect faces. I'll explain how the face detection algorithm works, and give you tips for getting the most out of it.

Background and Preliminaries

OpenCV uses a type of face detector called a Haar Cascade classifier. The sidebar, "How Face Detection Works, or What's a Haar Cascade Classifier, Anyhow?" explains what this mouthful means. Figure 1 shows an example of OpenCV's face detector in action.

Given an image — which can come from a file or from live video — the face detector examines each image location and classifies it as "Face" or "Not Face." Classification assumes a fixed scale for the face, say 50 x 50 pixels. Since faces in an image might be smaller or larger than this, the classifier runs over the image several times, to search for faces across a range of scales. This may seem like an enormous amount of processing, but thanks to algorithmic tricks (explained in the sidebar), classification is very fast, even when it's applied at several scales.

The classifier uses data stored in an XML file to decide how to classify each image location. The OpenCV download includes four flavors of XML data for frontal face detection, and one for profile faces. It also includes three non-face XML files: one for full body (pedestrian) detection, one for upper body, and one for lower body.

You'll need to tell the classifier where to find the data file you want it to use. The one I'll be using is called haarcascade_frontalface_default.xml. In OpenCV version 1.0, it's located at:

```
[OPENCV_ROOT]/data/haarcascades/
haarcascade_frontalface_default.
xml
```

where `[OPENCV_ROOT]` is the path to your OpenCV installation. For example, if you're on Windows XP and you selected the default installation location, you'd use:

```
[OPENCV_ROOT] = "C:/Program Files/
OpenCV"
```

(If you're working with an older, 16-bit version of Windows, you'd use '\' as the directory separator, instead of '/'.)

It's a good idea to locate the XML file you want to use and make sure your path to it is correct before you code the rest of your face detection program.

You'll also need an image to process. The image lena.jpg is a good one to test with. It's located in the OpenCV samples/c directory. If you copy it to your program's working directory, you'll easily be able to compare your program's output with the output from the code in Figure 2.

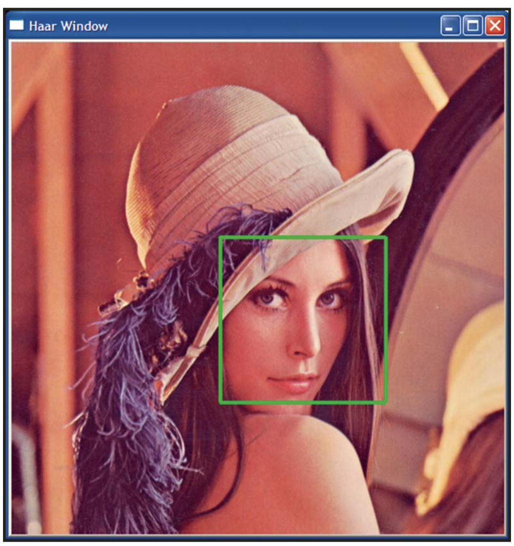


FIGURE 1. Face detection with OpenCV, using default parameters. The input image is lena.jpg, in the samples/c directory.

Implementing Face Detection, Step by Step

Figure 2 shows the source code to load an image from a file, detect faces in it, and display the image with detected faces outlined in green. Figure 1 shows the display produced by this program when it's run from the command line, using:

```
DetectFaces lena.jpg
```

Initializing (and running) the Detector

The variable `CvHaarClassifier Cascade * pCascade` (Line 2) holds the data from the XML file you located earlier. To load the XML data into `pCascade`, you can use the `cvLoad()` function, as in Lines 11-13. `cvLoad()` is a general-purpose function for loading data from files. It takes up to

FIGURE 2. Source code to detect faces in one image. Usage: Detect Faces <image file>.

```
 1 // declarations
 2 CvHaarClassifierCascade * pCascade = 0;      // the face detector
 3 CvMemStorage * pStorage = 0;                 // expandable memory buffer
 4 CvSeq * pFaceRectSeq;                        // list of detected faces
 5 int i;
 6
 7 // initializations
 8 IplImage * pInpImg = (argc > 1) ?
 9    cvLoadImage(argv[1], CV_LOAD_IMAGE_COLOR) : 0;
10 pStorage = cvCreateMemStorage(0);
11 pCascade = (CvHaarClassifierCascade *)cvLoad
12   ((OPENCV_ROOT"/data/haarcascades/haarcascade_frontalface_default.xml"),
13   0, 0, 0 );
14
15 // validate that everything initialized properly
16 if( !pInpImg || !pStorage || !pCascade )
17 {
18    printf("Initialization failed: %s \n",
19       (!pInpImg)?    "didn't load image file" :
20       (!pCascade)?   "didn't load Haar cascade -- "
21                      "make sure path is correct" :
22       "failed to allocate memory for data storage");
23    exit(-1);
24 }
25
26 // detect faces in image
27 pFaceRectSeq = cvHaarDetectObjects
28    (pInpImg, pCascade, pStorage,
29    1.1,                          // increase search scale by 10% each pass
30    3,                            // drop groups of fewer than three detections
31    CV_HAAR_DO_CANNY_PRUNING,     // skip regions unlikely to contain a face
32    cvSize(0,0));                 // use XML default for smallest search scale
33
34 // create a window to display detected faces
35 cvNamedWindow("Haar Window", CV_WINDOW_AUTOSIZE);
36
37 // draw a rectangular outline around each detection
38 for(i=0;i<(pFaceRectSeq? pFaceRectSeq->total:0); i++ )
39 {
40    CvRect * r = (CvRect*)cvGetSeqElem(pFaceRectSeq, i);
41    CvPoint pt1 = { r->x, r->y };
42    CvPoint pt2 = { r->x + r->width, r->y + r->height };
43    cvRectangle(pInpImg, pt1, pt2, CV_RGB(0,255,0), 3, 4, 0);
44 }
45
46 // display face detections
47 cvShowImage("Haar Window", pInpImg);
48 cvWaitKey(0);
49 cvDestroyWindow("Haar Window");
50
51 // clean up and release resources
52 cvReleaseImage(&pInpImg);
53 if(pCascade) cvReleaseHaarClassifierCascade(&pCascade);
54 if(pStorage) cvReleaseMemStorage(&pStorage);
```

three input parameters. For this example, you'll only need the first parameter. This is the path to an XML file containing a valid Haar Cascade. Here, I've loaded the default frontal face detector included with OpenCV. If you're coding in C, set the remaining parameters to 0. If you're coding in C++, you can simply omit the unused parameters from your function call.

Before detecting faces in images, you'll also need to instantiate a `CvMemStorage` object (`pStorage`, declared at Line 3). This is a memory buffer that expands automatically, as needed. The face detector will put the list of detected faces into this buffer. Since the buffer is expandable, you won't need to worry about overflowing it. All you'll have to do is create it (Line 10), then release it when you're finished (Line 54).

You'll often need to load data from files with OpenCV. Since it's easy to get a path wrong, it's a good idea to insert a quick check to make sure everything loaded and initialized properly. Lines 16-24 do a simple error check, print a diagnostic message, and exit if initialization fails.

Running the Detector

Lines 27-32 call `cvHaarDetectObjects()` to run the face detector. This function takes up to seven parameters. The first three are the image pointer, XML data, and memory buffer. The remaining four parameters are set to their C++ defaults. These last four parameters are described below, in the section, "Parameters and Tuning."

Viewing the Results

A quick way to check if your program works is to display the results in an OpenCV window. You can create a display window using the `cvNamedWindow()` function, as in Line 35. The first parameter is a string, with a window name. The second, `CV_WINDOW_AUTOSIZE`, is a flag that tells the window to automatically resize itself to fit the image you give it to display.

To pass an image for display, call `cvShowImage()` with the name you previously assigned the window, and the image you want it to display. The `cvWaitKey()` call at Line 48 pauses the application until you close the window. If the window fails to close by clicking its close icon, click inside the window's display area, then press a keyboard key. Also, make sure your program calls `cvDestroyWindow()` (Line 49) to close the window.

Face detections are stored as a list of `CvRect struct` pointers. Lines 38-44 access each detection rectangle and add its outline to the `pInpImg` variable, which holds the in-memory image loaded from the file.

Releasing Resources

Lines 52-54 release the resources used by the input image, the XML data, and the storage buffer. If you'll be detecting faces in multiple images, you don't need to release the

How Face Detection Works

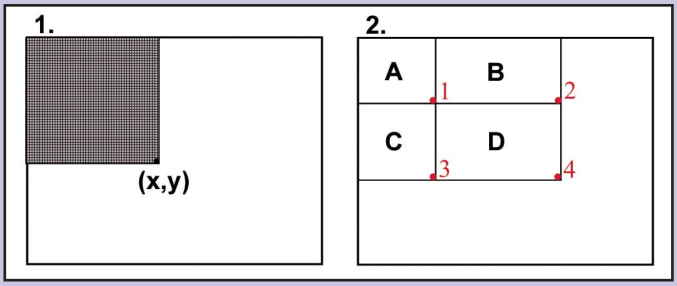


FIGURE B. The Integral Image trick. After integrating, the pixel at (x,y) contains the sum of all pixel values in the shaded rectangle. The sum of pixel values in rectangle D is (x4, y4) − (x2, y2) − (x3, y3) + (x1, y1).

OpenCV's face detector uses a method that Paul Viola and Michael Jones published in 2001. Usually called simply the Viola-Jones method, or even just Viola-Jones, this approach to detecting objects in images combines four key concepts:

- Simple rectangular features, called Haar features.
- An Integral Image for rapid feature detection.
- The AdaBoost machine-learning method.
- A cascaded classifier to combine many features efficiently.

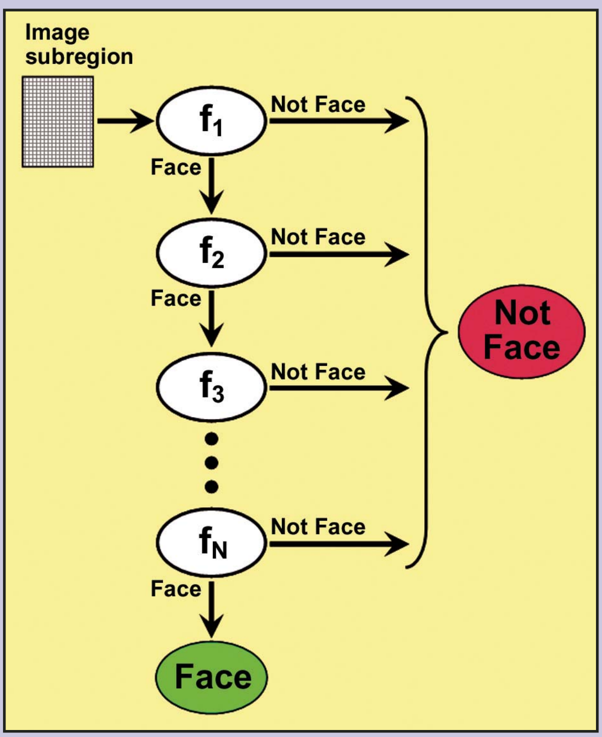


FIGURE C. The classifier cascade is a chain of single-feature filters. Image subregions that make it through the entire cascade are classified as "Face." All others are classified as "Not Face."

The features that Viola and Jones used are based on Haar wavelets. Haar wavelets are single-wavelength square waves (one high interval and one low interval). In two dimensions, a square wave is a pair of adjacent rectangles — one light and one dark.

The actual rectangle combinations used for visual object detection are not true Haar wavelets. Instead, they contain rectangle combinations better suited to visual recognition tasks. Because of that difference, these features are called Haar features, or Haarlike features, rather than Haar wavelets. Figure A shows the features that OpenCV uses.

The presence of a Haar feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present.

To determine the presence or absence of hundreds of Haar features at every image location and at several scales efficiently, Viola and Jones used a technique called an Integral Image. In general, "integrating" means adding small units together. In this case, the small units are pixel values. The integral value for each pixel is the sum of all the pixels above it and to its left. Starting at the top left and traversing to the right and down, the entire image can be integrated with a few integer operations per pixel.

FIGURE A. Examples of the Haar features used in OpenCV.

As Figure B1 shows, after integration, the value at each pixel location, (x,y) contains the sum of all pixel values within a rectangular region that has one corner at the top left of the image and the other at location (x,y). To find the average pixel value in this rectangle, you'd only need to divide the value at (x,y) by the rectangle's area.

But what if you want to know the summed values for some other rectangle, one that doesn't have one corner at the upper left of the image? Figure B2 shows the solution to that problem. Suppose you want the summed values in D. You can think of that as being the sum of pixel values in the combined rectangle, A+B+C+D, minus the sums in rectangles A+B and A+C, plus the sum of pixel values in A. In other words,

D = A+B+C+D − (A+B) − (A+C) + A.

Conveniently, A+B+C+D is the Integral Image's value at location 4, A+B is the value at location 2, A+C is the value at location 3, and A is the value at location 1. So, with an Integral Image, you can find the sum of pixel values for any rectangle in the original image with just three integer operations:

(x4, y4) − (x2, y2) − (x3, y3) + (x1, y1).

To select the specific Haar features to use and to set threshold levels, Viola and Jones use a machine-learning method called AdaBoost. AdaBoost combines many "weak" classifiers to create one "strong" classifier. "Weak" here means the classifier only gets the right answer a little more often than random guessing would. That's not very good. But if you had a whole lot of these weak classifiers and each one "pushed" the final answer a little bit in the right direction, you'd have a strong, combined force for arriving at the correct solution. AdaBoost selects a set of weak classifiers to combine and assigns a weight to each. This weighted combination is the strong classifier.

Viola and Jones combined weak classifiers as a filter chain, shown in Figure C, that's especial-

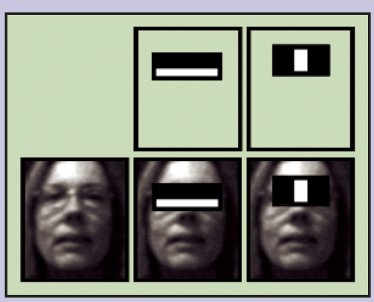


FIGURE D. The first two Haar features in the original Viola-Jones cascade.

ly efficient for classifying image regions. Each filter is a weak classifier consisting of one Haar feature. The threshold for each filter is set low enough that it passes all, or nearly all, face examples in the training set. (The training set is a large database of faces, maybe a thousand or so.) During use, if any one of these filters fails to pass an image region, that region is immediately classified as "Not Face." When a filter passes an image region, it goes to the next filter in the chain. Image regions that pass through all filters in the chain are classified as "Face." Viola and Jones dubbed this filtering chain a cascade.

The order of filters in the cascade is determined by weights that AdaBoost assigns. The more heavily weighted filters come first, to eliminate non-face image regions as quickly as possible. Figure D shows the first two features from the original Viola-Jones cascade superimposed on my face. The first one keys off the cheek area being lighter than the eye region. The second uses the fact that the bridge of the nose is lighter than the eyes.

XML data or the buffer until after you're done detecting faces.

Parameters and Tuning

There are several parameters you can adjust to tune the face detector for your application.

Minimum Detection Scale

The seventh parameter in the call to `cvHaarDetectObjects()` is the size of the smallest face to search for. In C, you can select the default for this by setting the scale to 0x0, as in Figure 2, Line 32. (In C++, simply omit this parameter to use the default.) But what is the default? You can find out by opening the XML file you'll be using. Look for the `<size>` tag. In the default frontal face detector, it's:

```
<size> 24 24 </size>.
```

So, for this cascade, the default minimum scale is 24 x 24.

Depending on the resolution you're using, this default size may be a very small portion of your overall image. A face image this small may not be meaningful or useful, and detecting it takes up CPU cycles you could use for other purposes. For these reasons — and also to minimize the number of face detections your own code needs to process — it's best to set the minimum detection scale only as small as you truly need.

To set the minimum scale higher than the default value, set this parameter to the size you want. A good rule of thumb is to use some fraction of your input image's width or height as the minimum scale — for example, 1/4 of the image width. If you specify a minimum scale other than the default, be sure its aspect ratio (the ratio of width to height) is the same as the default's. In this case, aspect ratio is 1:1.

Minimum Neighbors Threshold

One of the things that happens "behind the scenes" when you call the face detector is that each positive face region actually generates many hits from the Haar detector. Figure 3 shows OpenCV's internal rectangle list for the example image, lena.jpg. The face region itself generates the largest cluster of rectangles. These largely overlap. In addition, there's one small detection to the (viewer's) left, and two larger detections slightly above and left of the main face cluster.

Usually, isolated detections are false detections, so it makes sense to discard these. It also makes sense to somehow merge the multiple detections for each face region into a single detection. OpenCV does both these before returning its list of detected faces. The merge step first groups rectangles that contain a large amount of overlap, then finds the average rectangle for the group. It then replaces all rectangles in the group with the average rectangle.

Between isolated rectangles and large groupings are smaller groupings that may be faces, or may be false detections. The minimum-neighbors threshold sets the cutoff level for discarding or keeping rectangle groups based on how many raw detections are in the group. The C++ default for this parameter is three, which means to merge groups of three or more and discard groups with fewer rectangles. If you find that your face detector is missing a lot of faces, you might try lowering this threshold to two or one.

If you set it to 0, OpenCV will return the complete list of raw detections from the Haar classifier. While you're tuning your face detector, it's helpful to do this just to see what's going on inside OpenCV. Viewing the raw detections will improve your intuition about the effects of changing other parameters, which will help you tune them.

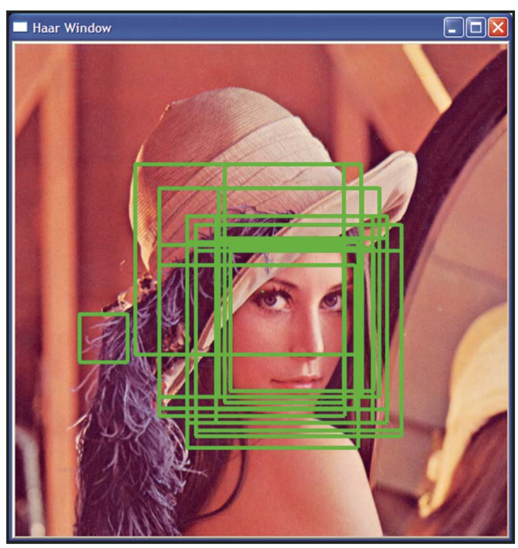


FIGURE 3. OpenCV's internal detection rectangles. To see these, use min_neighbors = 0.

Scale Increase Rate

The fourth input parameter to `cvHaarDetectObjects()` specifies how quickly OpenCV should increase the scale for face detections with each pass it makes over an image. Setting this higher makes the detector run faster (by running fewer passes), but if it's too high, you may jump too quickly between scales and miss faces. The default in OpenCV is 1.1, in other words, scale increases by a factor of 1.1 (10%) each pass.

Canny Pruning Flag

The sixth parameter to `cvHaarDetectObjects()` is a flag variable. There are currently only two options: 0 or `CV_HAAR_DO_CANNY_PRUNING`. If the Canny Pruning option is selected, the detector skips image regions that are unlikely to contain a face, reducing computational overhead and possibly

References and Resources

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

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

- G. Bradski, A. Kaehler, and V. Pisarevsky, "Learning-Based Computer Vision with Intel's Open Source Computer Vision Library," *Intel Technology Journal*, Vol 9(1), May 19, 2005. **www.intel.com/technology/itj/2005/volume09issue02/art03_learning_vision/p01_abstract.htm**

- R.E. Schapire, "A Brief Introduction to Boosting," *Joint Conference on Artificial Intelligence*, Morgan Kauman, San Francisco, pp. 1401-1406, 1999.

- P. Viola and M.J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," CVPR, 2001.

eliminating some false detections. The regions to skip are identified by running an edge detector (the Canny edge detector) over the image before running the face detector.

Again, the choice of whether or not to set this flag is a tradeoff between speed and detecting more faces. Setting this flag speeds processing, but may cause you to miss some faces. In general, you can do well with it set, but if you're having difficulty detecting faces, clearing this flag may allow you to detect more reliably. Setting the minimum-neighbors threshold to 0 so you can view the raw detections will help you better gauge the effect of using Canny Pruning.

The Haar Cascade

There are several frontal face detector cascades in OpenCV. The best choice for you will depend on your set-up. It's easy to switch between them — just change the file name. Why not try each?

It's also possible to create your own, custom XML file using the HaarTraining application, in OpenCV's apps directory. Using that application is beyond the scope of this article. However, the instructions are in OpenCV's apps/haartraining/docs directory.

Coming Up ...

Now that you've found a face, you might want to follow it around. Next month, I'll show you how to use Camshift, OpenCV's face tracking method, to do just that. Be seeing you! **SV**

About the Author

Robin Hewitt is an independent software consultant working in the areas of computer vision and robotics. She has worked as a Computer Vision Algorithm Developer at Evolution Robotics and is a member of SO(3), a computer-vision research group at UC San Diego. She is one of the original developers of SodaVision, an experimental face-recognition system at UC San Diego. SodaVision was built with OpenCV.

Beginner's Robotics on $50 a Month

– by Paul Pawelski

PART 3: Sensors and Output

Amateur robotics is FUN.

Amateur robotics is EDUCATIONAL.

Amateur robotics is EXPENSIVE!

Last month, we built a robot that met the minimum requirements (in my opinion) to be called a robot. It could move through its environment, it could sense its environment, it could modify its actions based on the readings of its sensors, and it could be easily modified. This month, we are going to greatly expand the capabilities of your CIRC Bot by adding an infrared object detector, bump sensors, and an LCD display.

With these additions, the CIRC Bot will have a pretty good idea of what is in front of it and will be able to tell you how it is doing. To do all this with our little microprocessor, we are going to need to add some "glue logic" chips — a multiplexer and an inverter.

We are also going to build a logic tester so we can make sure the sensors and glue logic work correctly before being connected to the microprocessor.

Parts in Kit

Figure 1 lists the parts included in the kit from Wright Hobbies that goes with this month's article. Figure 2 is the schematic for this month's project. The lines shown on the microprocessor with numbers below 20 are from last month's article and have been left in to remind you what ports have already been used.

Description	Qty	Description	Qty
74HC04 Inverter DIP Chip	1	Header Pin Strip	1
DIP Switch	1	10K Potentiometer	1
74HC151 Multiplexer Chip	1	Tall PC Board Standoffs (F-F)	4
IS471FE IR Detector/Modulator	1	Tamiya Universal Plate Set	1
IR LED	6	Breadboard 400 Contact (small)	2
.33 µF Mylar Cap	1	22 Ga Wire Kit (red, green, black)	1
450K ohm 1/4 watt Resistors	4	4-40 x 3/8" Machine Screw	8
100K ohm 1/4 watt Resistors	4	Ceiling Hanger Wire	1
Mini Breadboard Switches	2	Red LED	1
LCD 16x1 Character Display	1	360 ohm 1/4 watt Resistors	1

FIGURE 1. Parts List for this month's kit.

Assembly

The Logic Tester

There are devices where the transition between a high signal and a low signal occurs at a voltage level know as the transition voltage (V_T). However, to ensure that the signal you are sending a chip is clearly understood as being high, you need to present a voltage above the V_{IH} or voltage input, high. That level varies some by the type of chip (CMOS or TTL), but staying above 3.7V on a 5V system is sufficient. Likewise, to ensure that a low signal occurs, we need to stay below V_{IL} or voltage input, low which is always at least 0.8V on a 5V system. Take a look at **www.interfacebus.com/voltage_threshold.html** and the pages linked to it for more information on different types of logic chips and what needs to be done to interface between them.

So, to test logic, we need to have a way to present voltages above 3.7V for high signals and below 0.8V for low signals. We could do this just by hooking up jumpers to the +5V and ground lines on the bot, but this gets a little tedious when you are trying to test several ports. It would be much more convenient to have a switch that could be flicked between high and low. However, a simple switch goes between on and off. If you hook it up to 5V on one side and the logic gate which you want to

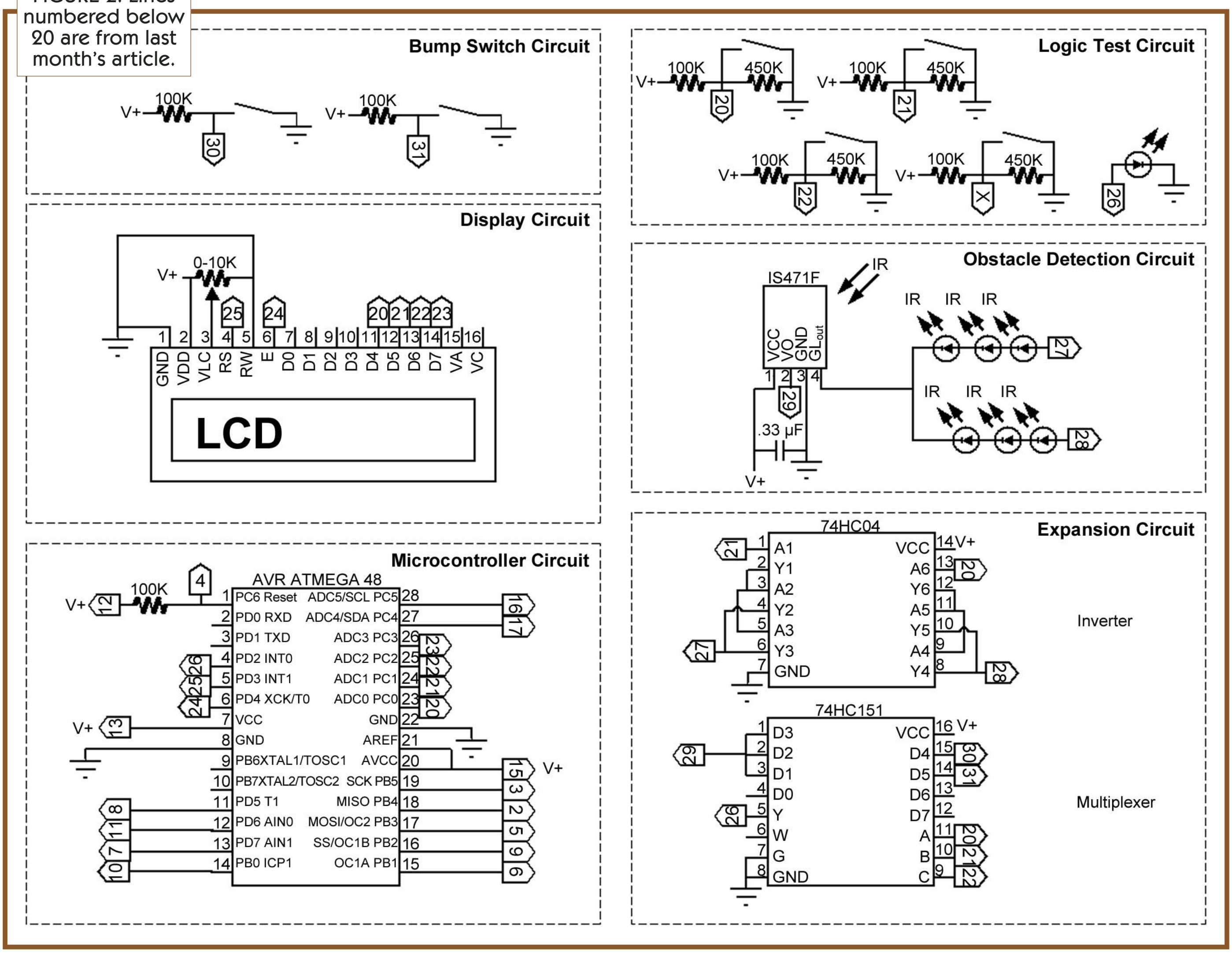


FIGURE 2. Lines numbered below 20 are from last month's article.

test on the other, you will get 5V when it is on, but the chip won't necessarily see 0V when it is off.

Recall from last month that I described the light sensors as being voltage dividers. The voltage at the measuring node (between the resistor and the CDS cell) which was seen by the A/D pin on the AVR was determined by the ratio of the CDS cell resistance to the resistance of the fixed value resistor. If you had a voltage divider made up of a 100K resistor connected between the measuring node and +5V and a 470K resistor connected between the node and ground, the voltage at the node would be:

5V x 470K / (100K + 470K) = 4.12V

If the 470K resistor was shorted out by turning on a switch which is connected between the node and ground, the voltage at the node would drop to almost zero (the switch does have a very small resistance value so it won't be exactly zero, but it will be less than 0.8V).

To test logic, you not only have to control the input, but you also need to read the output. We could read the output with the multimeter, but temporarily placing a simple LED between the output pin of the device being tested and ground will allow quicker readings. It will glow when a high voltage is seen. (The chips we are using put out enough current to light an LED without needing an additional resistor.) It would be better to have a logic probe.

A logic probe has three lights on it. One light lights up when the probe sees a high signal above V_{IH}. A different one lights up when a signal is seen below V_{IL}. The third light indicates when the signal is pulsing. The guides section of the Wright Hobbies website (**www.wrighthobbies.net**) contains a description of how to make a simple logic probe using a single NOR logic chip, a diode, a few resistors, and a couple of capacitors. You can also buy ready-made logic testers for around $15 from several sites online (use Google to search for "logic probe").

We are going to build the logic tester on one of the small breadboards provided in this month's kit. To connect it to the CIRC Bot to test circuits, we will make a set of jumpers from some of the wire leftover from making the programming cable last month. Cut a 10" piece from the leftover cable. Slide the cable sleeve so that about three inches of it extends beyond the end of the wires. Cut off the extended piece of sleeve and push the rest of the sleeve so that it is centered on the wires. Strip about 1/4" off each end of each wire. Snap off a set of six connector pins from the strip provided in the kit. Snap off another six pins one at a time (use pliers). You should now have the items shown in Figure 3.

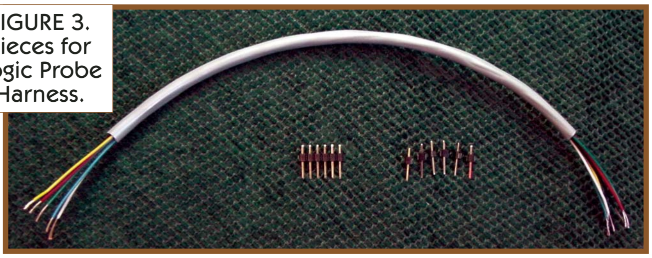


FIGURE 3. Pieces for Logic Probe Harness.

On one end of each wire, solder a single pin (attach the wire to the short side of the pin). The other end of the wires will be attached to the connector strip. Attach the wires in the following order: *red, black, blue, green, yellow, white*. Now, use pliers to snap the first two connectors (with the red and black wires) and the last connector (with the white wire) off the strip. Now build the rest of the tester on the breadboard, as indicated by the table in Figure 4. Note that the 470K resistors will have to have double the lead length (18 mm on each end) so that they can sit over the 100K resistors. Figure 5 shows the finished logic tester.

Now, you need to make sure that your logic tester is working properly. Connect the red lead from the tester to the +5V bus on the CIRC Bot. Connect the black lead to the ground bus on the CIRC Bot. The logic probe uses power from the CIRC Bot so the bot will have to be on during the test. To keep the bot from moving, disconnect the power line to the H-bridge

Description	From	To	Length/Notes
Red Wire	Upper +1	I1	11 mm (add 18 mm for leads on all wires)
Red Wire	F1	E1	9 mm
Red Wire	A1	Lower +V 1	11 mm
Black Wire	Upper GND 2	J2	11 mm
Black Wire	F2	E2	9 mm
Black Wire	A2	Lower GND 2	8 mm
100K Resistor	Upper +V3	J3	8 mm (add 18 mm for leads on all 100Ks)
100K Resistor	Upper +V4	J4	8 mm
100K Resistor	Upper +V5	J5	8 mm
100K Resistor	Upper +V7	J7	8 mm
470K Resistor	Upper GND 3	I3	14 mm (add 32 mm for leads on all 470Ks)
470K Resistor	Upper GND 4	I4	14 mm
470K Resistor	Upper GND 5	I5	14 mm
470K Resistor	Upper GND 7	I7	14 mm
Black Wire	A3	Lower GND 3	8 mm
Black Wire	A4	Lower GND 4	8 mm
Black Wire	A5	Lower GND 5	8 mm
Black Wire	A7	Lower GND 7	8 mm
Test Lead — Red	H1	—	(Black lead at H2)
Test Lead — Blue	H3	—	(Yellow lead at H5)
Test Lead — White	H7	—	
DIP Switch	E3	H10	Switch

FIGURE 4. Logic Tester.

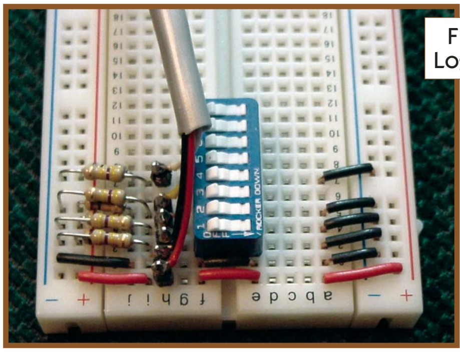


FIGURE 5. Logic Tester.

(it is the red wire from J50 to J60).

For now, leave the rest of the leads free. Turn the bot on. Using your multimeter set to read voltage, you should see about five volts between the red and black leads. If you do not get this, you might have dead batteries in the bot or bad solder connections on the leads.

With all the switches in the open (off) position, the voltage between the black lead and each of the other leads should be around 4.1V. If it is around 0.8V, you have the 470K and 100K resistors reversed. If it is around 0.1V, you have the switch in the closed position.

One at a time, move the switches from off to on. On each line where you have closed a switch, the lead connected to that line should change to around 0.1V. If the voltage does not change, you either have a wiring mistake or the switch has worked itself loose from the breadboard.

Adding Input Ports

There are certain things in life of which you can always use more. One of them is ports on a microcontroller. While being able to dedicate a port to each sensor is the ideal solution, you can often get by if you just had a way to select between multiple input signals on one port pin.

The 74HC151 is an 8-to-1 data selector/multiplexer. Look at the data sheet for the chip at **http://focus.ti.com/lit/ds/symlink/sn74hc151.pdf**. The selector lets you use three output lines from you microcontroller to choose which of eight digital input signals will go to one input line on your micro — effectively doubling the number of inputs you can have, compared to using all four lines from your micro as input.

Install and wire the multiplexer on the CIRC Bot and hook up the logic tester as indicated by the table in Figure 6 and shown in Figure 7. By setting the switches on the logic probe and placing the white wire as shown in the table in Figure 8, you can confirm that the chip is working properly (the LED should light each time the switch connected to the white wire is in the high position).

IR Obstacle Sensor Circuit

It is better for your bot (and for the objects around which it maneuvers) if it can sense an object before it hits. There are four ways of identifying objects before they make contact: video image processing, reflected light detection, reflected sound detection, and electromagnetic field disturbance. The first and the last methods listed are beyond the scope of this series. This month, we will cover reflected light detection. Next month, reflected sound detection.

Last month, you made half of a simple reflected light detection circuit. The two CDS cells you mounted on your CIRC Bot could detect visible light. One of the design variations (which was suggested for you to try) was to add a light and use the CDS sensors to make a line follower. If you tried that, you probably discovered several things.

The first thing was that different colored surfaces reflect different amounts of light (if they didn't, you couldn't make a line follower). Second, you probably noticed that ambient light reeked havoc with your sensor's ability

Description	From	To	Length/Notes
74HC151	E22	F29	Pin 1 @ E22
Red Wire	F22	Lower +V22	12 mm (add 18 mm for leads on all wires)
Black Wire	D29	Upper GND 29	15 mm
Black Wire	D28	Upper GND 28	15 mm
Red LED	B26	Upper GND 25	Flat part to GND side
Black Probe Wire	—	Lower GND 24	
Red Probe Wire		Lower +V24	
Blue Probe Wire	—	G27	Switch 1, line 20, pin 11 (A)
Green Probe Wire	—	H28	Switch 2, line 21, pin 10 (B)
Yellow Probe Wire	—	I29	Switch 2, line 22, pin 9
White Probe Wire	—	Various	See Logic Test Table, Figure 8

FIGURE 6. Multiplexer Testing Circuit.

Location of White Wire (Switch 5)	Switch 1 (A)	Switch 2 (B)	Switch 3 (C)
Pin 4 (D0)	L	L	L
Pin 3 (D1)	H	L	L
Pin 2 (D2)	L	H	L
Pin 1 (D3)	H	H	L
Pin 15 (D4)	L	L	H
Pin 14 (D5)	H	L	H
Pin 13 (D6)	L	H	H
Pin 12 (D7)	H	H	H

FIGURE 8. Logic Test for mux — Output at pin 5 will match setting of switch 5.

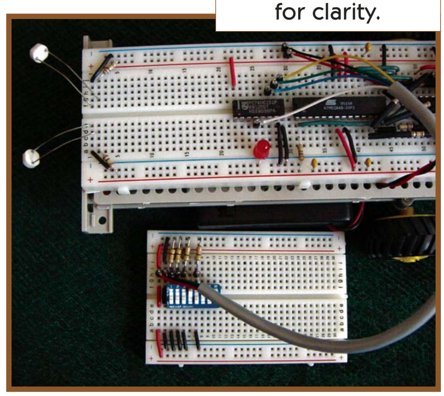


FIGURE 7. Test Setup for mux — CDS cell wires removed for clarity.

to correctly identify the line. Your sensor probably worked best when the sensors and the light were very close to the ground. If you were creative, you might have even built a light shield to put around your light and sensors to keep out most of the outside lighting.

While the shield worked okay for a line follower, it isn't useful for an object detector since you need to detect objects that are not right up against the shield. You need another method (or two) to keep ambient lighting from interfering with your object detector.

By switching to a sensor that uses infrared (IR) light, interference from most man-made lighting can be avoided. However, sunlight has a lot of IR in it. The way to avoid interference from natural light is to modulate (turn on and off at a fixed rate) the light source and put an electronic filter in the light sensor so that only light that is modulated at the same frequency is measured.

For the CIRC Bot, we are going to use the IS471F IR light detector. It is a small sensor that has the modulation circuit for an IR LED light source built in. The data sheet for the sensor can be found at **www.junun.org/MarkIII/datasheets/IS471F.pdf**. Note that in many of the figures in the datasheet, the position of pins 3 and 4 are reversed.

The sensor is going to be placed on one of the small breadboards. Eventually, that breadboard will be mounted on the front of the CIRC Bot with its holes facing forward. That means that the leads on the IR detector will have to be bent so that the sensor faces forward. Figure 9 shows how to bend the leads.

Connecting the negative side of an LED to the GL_{out} pin of the IS471 (pin 4) and putting a voltage to the LED's positive pin through a small (200-400 ohm) resistor will cause it to produce modulated light. If you try this with a visible light LED, you will see it light, but you will not be able to tell if it is being modulated since the human eye cannot see something that fast (and the sensor won't see it either). The easy way to test the modulation is to see if the sensor can pick up IR light reflected off an object. Make the circuit described by the table in Figure 10 and shown in Figure 11. This is not the final configuration we will use on the bot, but it is simple and easy to test.

IMPORTANT: The IS471 sensor will fry if it is hooked up incorrectly. Before applying power, make sure that pin 1 (the bent pin closest to an edge of the sensor) is connected to power, pin 3 (the other bent pin) is connected to ground, and that pins 2 and 4 never get connected directly to power.

Hook the power and ground buses on the small breadboard to the buses on the CIRC bot. Use the multimeter to measure the voltage on the V_o line (pin 2) on the sensor relitive to ground. If there is nothing in front of the sensor to reflect the IR light back to it, the voltage should be around 5V. Now place your hand about three inches away from the sensor. The sensor should pick up the IR light from the IR LEDs reflected off your hand. This will cause the voltage to drop to almost zero.

If V_o was low even when there was nothing in front of the sensor, there is either a wiring error (was the IR LED put in backwards?), there is no power going to the sensor, or the sensor was fried.

The rest of this month's project is online at the *SERVO Magazine* website (**www.servomagazine.com**) and at **www.wrighthobbies.net/guides/**. In the online section, you will learn how to make the IR sensor detect not just that an object is in front of it, but also whether it is more to the left or right of center. You will also build bump sensors, an LCD interface, connect everything to your CIRC bot, and load a BASCOM program that uses these features.

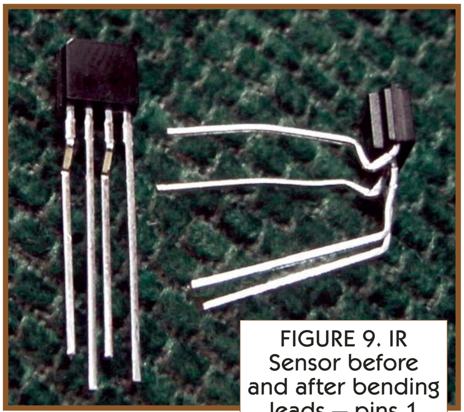


FIGURE 9. IR Sensor before and after bending leads — pins 1 and 3 are bent.

Conclusion

Your CIRC Bot is becoming more complex. Last month, it had the hardware necessary to seek or avoid light — based on how you programmed it. This month, it can also know when it has hit or is about to hit an obstacle. If you strike out on your own and try some variations, it can also be made capable of line following, edge detection, or dead reckoning. By mixing and matching the bits you have built on your CIRC bot and creating the right code, you can make a bot that can do maze solving, sumo wrestling, object retrieval, or line following.

Next month, you will learn how to free your bot of the limitations of a single small microcontroller and give it the gift of sight with scanning sonar. Until then, have fun with your project! **SV**

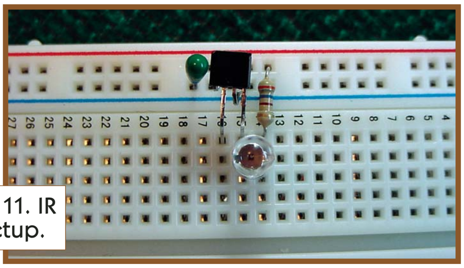


FIGURE 11. IR Test Setup.

Description	From	To	Length/Notes
.33 µF	Upper +V18	Upper GND 18	
360 ohm Resistor	Upper +V14	A14	11 mm (add 18 mm for leads)
IR LED	B15	B14	Flat at B15
IS471 Sensor pin 1	Upper +V16	—	See Figure 9 for how to bend leads
IS471 Sensor pin 2	A16	—	See Figure 9 for how to bend leads
IS471 Sensor pin 3	Upper GND 15	—	See Figure 9 for how to bend leads
IS471 Sensor pin 4	A15	—	See Figure 9 for how to bend leads

FIGURE 10. IR Test Circuit — jumper +V and GND to CIRC Bot for Power.

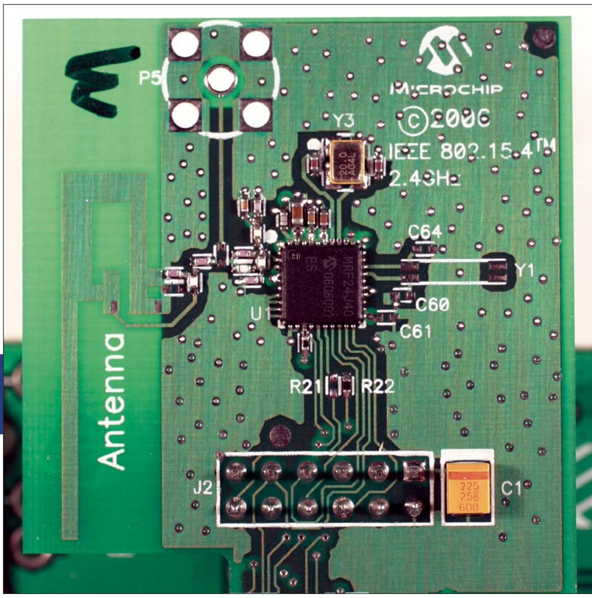


PHOTO 1. All we have to do is plug it in and talk to it via the PIC's SPI interface. If you are feeling froggy and want to build up your own MRF24J40 daughterboard, Microchip provides the complete set of printed circuit board layouts for you.

Low Power Robot Communications

by Peter Best

Some things robotic are givens. Motors, gears, belts, and wheels are pretty much robotic commodity items. Unless you pour gasoline into your creation or boil water to create pressure to move the mechanical parts, you power your mechanical monster with electricity, which is most always provided by a battery-based power supply. Therefore, power consumption is a very important aspect of your robotic design.

I would like to introduce you to a low-power, low-data-rate radio solution that you can include in your robotic designs. The radio we'll be discussing is available from Microchip as an IC and is called the MRF24J40. Drawing only 18 mA in receive mode and 22 mA in transmit mode, the MRF24J40 is an insignificant part of your robotic power budget. The addition of an MRF24J40-based transceiver to your mechanics enables peer-to-peer communications between robotic nodes, wireless data collection, and wireless interconnects between your creation's internal mechanical and electromechanical modules.

If you're thinking ahead and see large and complex operating stacks in your future, please stop thinking. We can implement the MRF24J40 radios without the firmware complication. All we need is a PIC microcontroller that supports a SPI interface and a little bit of MRF24J40 and IEEE 802.15.4 know-how. You probably already have your PIC. So, I'll provide you with some MRF24J40 know-how.

The Microchip MRF24J40

The MRF24J40 is an IEEE 802.15.4-compliant transceiver that is designed to support ZigBee and other proprietary protocols. ZigBee will not be a player here as we're going to manipulate the MRF24J40 features with simple PIC routines. We will, however, use a spattering of IEEE 802.15.4.

Everything RF is taken care of by the MRF24J40. To make it even easier to put an MRF24J40 to work, Microchip offers a MRF24J40-based radio module that we can simply plug into our PIC. One of the MRF24J40 daughterboard modules we'll be firing up is shown in Photo 1.

I like to keep things simple. So, don't let the innards of the MRF24J40 cause you grief. Just think of the MRF24J40 as a self-contained black box that only needs you to provide it with an SPI interface, a reset I/O line, and, optionally, a PIC interrupt input line. All of the communications between the PIC and the MRF24J40 are done via the SPI interface. We'll write some simple SPI data transfer routines that will turn the data transfer process into a bunch of simple functions that we can call from our application code.

The MRF24J40 is controlled and monitored by way of its internal registers, just like a PIC. The MRF24J40 Short Address Control Registers and Long Address Control Registers are used for configuration, control, and status indication. You and I can access the Short Address Control Registers and Long Address Control Registers directly via the SPI interface. Don't get excited, as there's only a handful of registers we'll have to deal with.

To show you how easy it is to deal with the MRF24J40 control registers, let's write our Short Address Control Register read function right now. Before we write the Short Address Control Register read function, let's put down our base SPI read and write functions.

```
void SPIPut(char data)
{
     SSPIF = 0;
     do
     {
          WCOL = 0;
          SSPBUF = data;
     } while( WCOL );

     while( SSPIF == 0 );
}

char SPIGet(void)
{
     SPIPut(0x00);
     return SSPBUF;
}
```

The SPIPut function clears the SPI interrupt flag bit (SSPIF) and enters the send data loop. If the data is sent to the SPI data buffer (SSPBUF) while an SPI transmission or SPI reception is in progress, the SPI write collision detect bit (WCOL) will be set. Simply writing the data to the SSPBUF initiates the SPI transfer and the bits are clocked out of the PIC's SDO pin. When the SPI transmission is complete, the SSPIF flag is set.

Bits are clocked out of the PIC's SPI interface on the rising edge of the SPI clock. Bits are received into the PIC's SDI pin on the falling edge of the SPI clock. So, to receive bits, we can simply clock out a null byte (0x00) and the incoming bits will be collected in the SSPBUF.

Now we have the basis for putting down our Short Address Control Register read function. The first and last bits of our Short Address Control Register read command byte must be zeros. The six bits between the leading and trailing zeros contain the address of the Short Address Control Register we want to read. So, our code will look like this:

```
char GetShortRAMAddr(char address)
{
  char data;
  MRF_CS = 0;
  SPIPut((address<<1) & 0b01111110);
  data = SPIGet();
  MRF_CS = 1;
  return data;
}
```

To keep the code for our Short Address Control Register read function as simple as possible, let's use the Short Address Control Register address definitions provided with the MRF24J40.h file, which we can get freely from the Microchip website. The Microchip Short Address Control Register definitions have the shift and the logical AND built into the address value. For instance, the PANIDL Short Address Control Register is at physical address 0x01. Shifting and logical ANDing as shown in our initial GetShortRAMAddr function results in sending 0b00000010 (0x02) as our Short Address Control Register read command byte. It's much easier and more efficient to use the Microchip definition:

```
#define READ_PANIDL (0x02)
```

That allows us to write the Short Address Control Register read function like this:

```
char GetShortRAMAddr(char address)
{
   char data;
   MRF_CS = 0;
   SPIPut(address);
   data = SPIGet();
   MRF_CS = 1;
   return data;
}
```

where MRF_CS represents the PIC general-purpose I/O line that drives the MRF24J40's SPI chip select line.

The next clock edge following the final zero of the command byte will shift out the most significant bit of the addressed Short Address Control Register's data. The format of the Short Address Control Register read command and the resulting clocked out data looks like this:

00 A5 A4 A3 A2 A1 A0 00 D7 D6 D5 D4 D3 D2 D1 D0

where: AX = Address bits
DX = Incoming Data bits

Piece of Cake, a Walk in the Park

Writing to a Short Address Control Register is just as easy. The final bit of the Short Address Control Register write command is a 1 (one) instead of a zero. Otherwise, the address shift and mask logic we just examined in the Short Address Control Register read function remains the same. Instead of writing a null to read the data that is clocked in as a result, we send our data to the addressed Short Address Control Register in the second byte sent over the SPI interface. Again, we use the Microchip-supplied definitions to eliminate the shift and mask operations, which results in us putting down this Short Address Control Register write function code:

```
void SetShortRAMAddr(char address, char data)
{
    MRF_CS = 0;
    SPIPut(address);
    SPIPut(data);
    MRF_CS = 1;
}
```

Here's the address information you would plug in for a write to the PANIDL register:

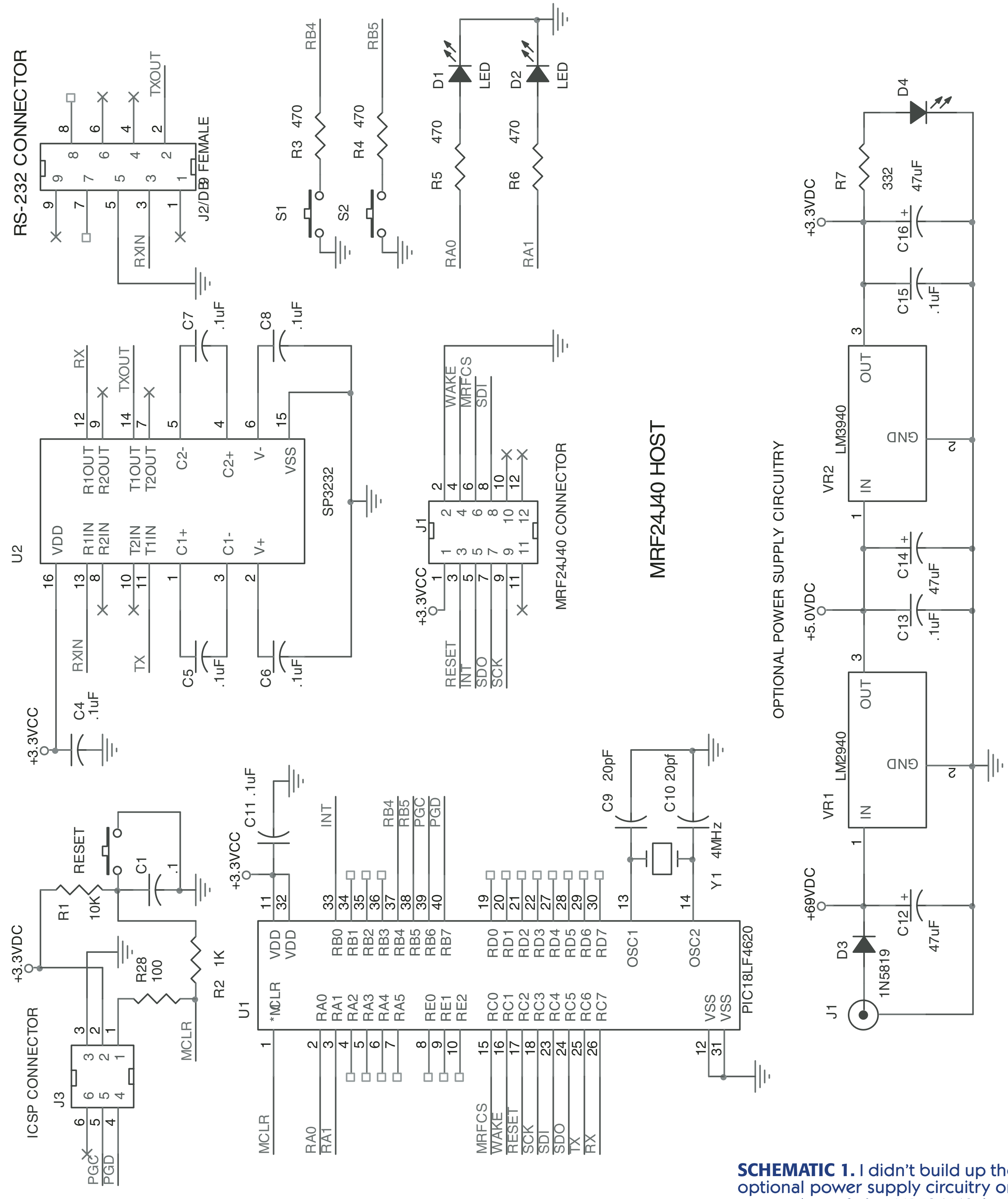
```
#define WRITE_PANIDL (0x03)
```

Here's the Short Address Control Register write command format from the SPI interface's point-of-view:

00 A5 A4 A3 A2 A1 A0 01 D7 D6 D5 D4 D3 D2 D1 D0

where: AX = Address bits
DX = Outgoing Data bits

We might as well get the Long Address Control Register read and write functions behind us, too. Unfortunately, we can't call on the Microchip prefab address definitions as there are none for the Long Address Control Registers. The Long Address Control Register read command begins with a 1 (one) and ends with a zero, with 10 bits of address information stuffed in between. Two eight-bit SPI transfers are required for both the Long Address Control Register read and write operations with the last four bits of the second eight-bit transfer being ignored by the MRF24J40. All of the Long Address Control Register addresses are 10 bits in length. Thus, the 16-bit command byte for a Long Address Control Register read would have the following format:



SCHEMATIC 1. I didn't build up the optional power supply circuitry on my version of the MRF24J40 host module. However, you may need it for your MRF24J40 application.

01 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 00 X X X X D7 D6 D5 D4 D3 D2 D1 D0

where: X = Don't care
AX = Address bits
DX = Incoming Data bits

```
char GetLongRAMAddr(unsigned int address)
{
     char data;
     MRF_CS = 0;
     SPIPut(((address>>3)
       &0b01111111)|0x80);
     SPIPut(((address<<5)
       &0b11100000));
 data = SPIGet();
 MRF_CS = 1;
 return data;
}
```

You've noticed that a 1 (one) at the end of the address bits signals a write operation and here's what the Long Address Control Register write SPI sequence looks like:

01 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 01 X X X X D7 D6 D5 D4 D3 D2 D1 D0

where: X = Don't care
AX = Address bits
DX = Outgoing Data bits

To force the Long Address Control Register write SPI sequence out of the PIC's SDO line requires the following code:

```
void SetLongRAMAddr(unsigned int address, char data)
{
     MRF_CS = 0;
     SPIPut((((char)(address>>3)
       )&0b01111111)|0x80);
     SPIPut((((char)(address<<5)
       )&0b11100000)|0x10);
     SPIPut(data);
     MRF_CS = 1;
}
```

Everything we will do in terms of firmware will be based on the simple SPI I/O functions we just created. If you're SPI challenged, take a look at the Master Synchronous Serial Port Module (MSSP) section of the PIC18LF4620 datasheet. You'll find that the PIC does most all of the work once you get the SPI interface configured. So, now that you know which PIC will be in charge of the MRF24J40, this is a good time to get the hardware ready.

The MRF24J40 Support Hardware

The PIC18LF4620 contains more than enough program Flash and SRAM to support the MRF24J40 in this application. All we need to do is help the PIC18LF4620 communicate with the MRF24J40 and with us. Communicating with the MRF24J40 is easy as the PIC18LF4620 is equipped with an integral SPI interface. Speaking human is also an easy task for the PIC18LF4620. The addition of an ST3232 RS-232 converter IC aids the PIC18LF4620's EUSART (Enhanced Universal Synchronous Receiver Transmitter) in transferring bits to us in human-readable ASCII format via a terminal emulator running on a PC.

To keep things easy (the way I like them), I've installed a standard RJ-12 jack to enable the connection of a Microchip MPLAB ICD 2 programmer/debugger module. You may want to use another type of programming/debugging connector, depending on how you apply your MRF24J40 node.

As I won't be mounting my MRF24J40 module inside of a mechanical beast just yet, for the purposes of discussion within this article, I will use a +3.3 VDC wall-wart to power my MRF24J40 module. If your mechanical creation's electrical supply can provide a regulated +3.3 VDC, you won't need an additional voltage regulator to power your MRF24J40 module. However, if your electromechanical modules run at +5.0 VDC, you'll need to add a suitable +3.3 VDC voltage regulator to power your MRF24J40 module.

A small SOT-23-packaged +3.3 VDC voltage regulator device like the LP2981 works well. If space is not a problem, you can use an LM3940, which is a +3.3 VDC voltage regulator packaged in a standard TO-220 case. The LM3940 is designed to feed directly from the output of a +5.0 VDC voltage regulator in dual-power systems. I've included an example of a dual-power supply in Schematic 1. The MRF24J40 daughterboard draws very little power. If you keep the transmit duty cycle very low, you can easily power your MRF24J40 module with a battery pack.

The PIC18LF4620 can only attain a maximum clock frequency of 25 MHz when running at +3.3 VDC. So, we'll use a 4 MHz crystal and the PIC18LF4620's internal 4X PLL to provide a PIC18LF4620 clock frequency of 16 MHz at +3.3 VDC. If you've ever worked with PIC microcontrollers, there is

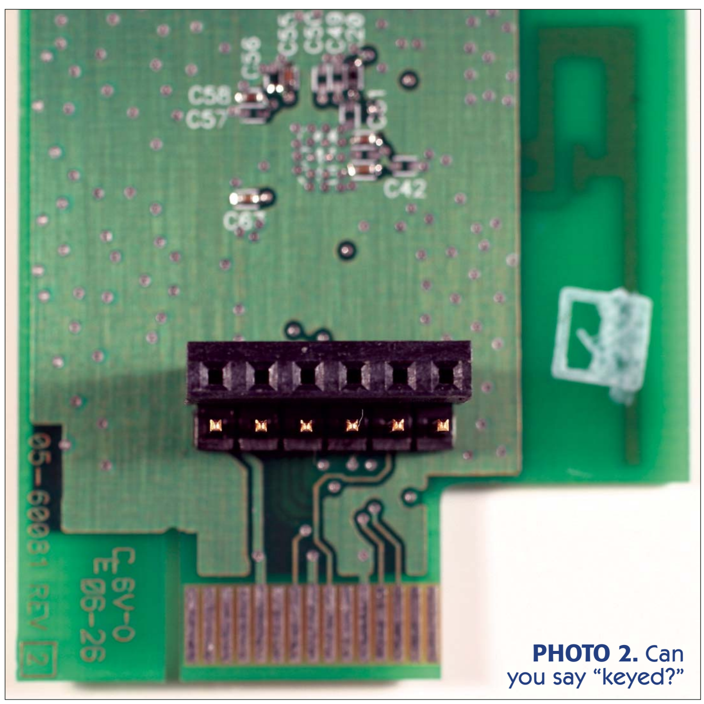
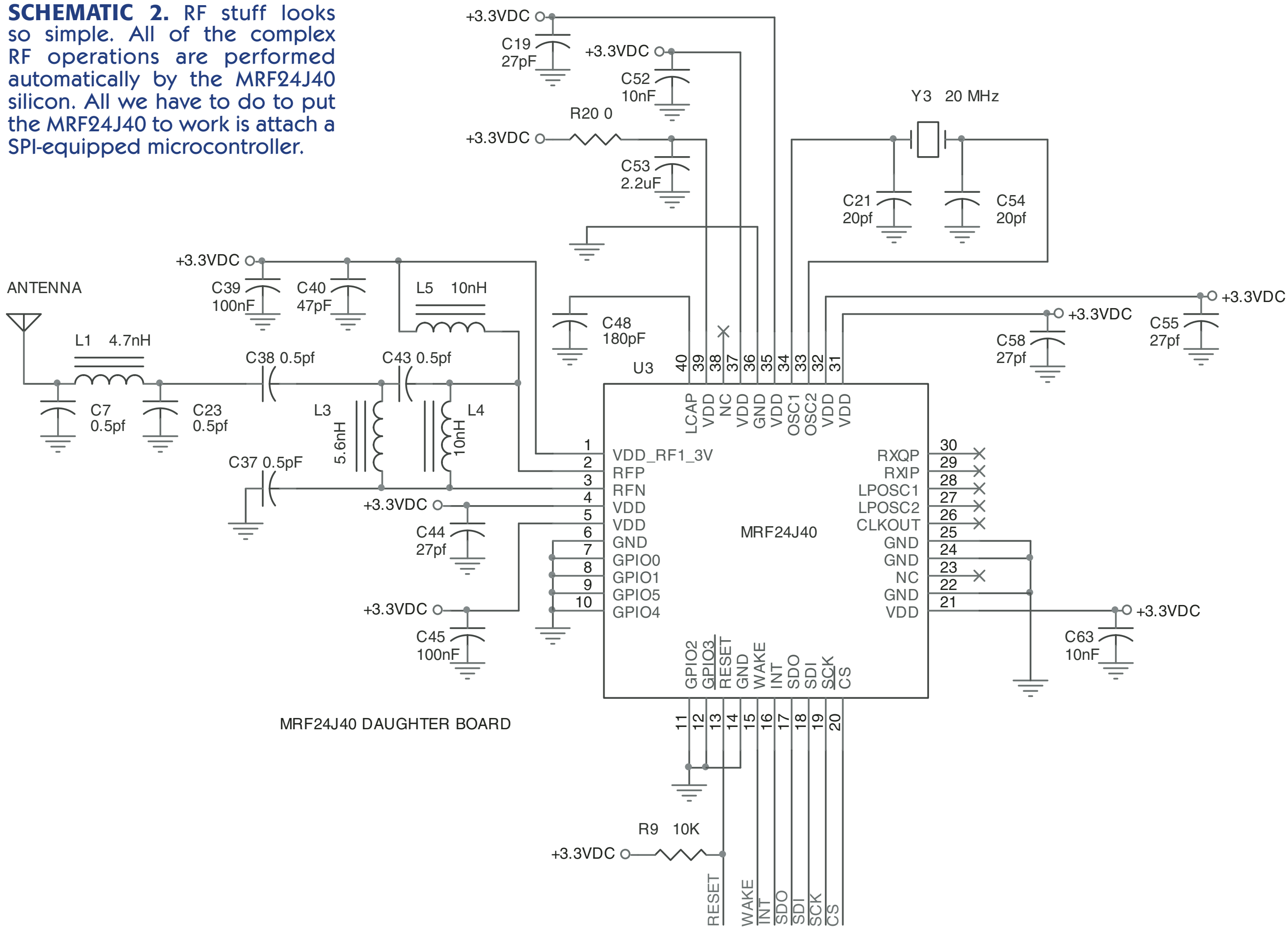


PHOTO 2. Can you say "keyed?"



SCHEMATIC 2. RF stuff looks so simple. All of the complex RF operations are performed automatically by the MRF24J40 silicon. All we have to do to put the MRF24J40 to work is attach a SPI-equipped microcontroller.

nothing remarkable about the RESET, ICSP, and clock circuitry you see depicted in Schematic 1. If you don't feel comfortable with the PIC support circuitry shown in Schematic 1, I suggest putting Chuck Hellebuyck's monthly *Nuts & Volts* column, "Getting Started With PICs," on your required reading list.

Everything you see in Photo 1 is displayed schematically in Schematic 2. With the exception of the RF I/O, most of the components that support the MRF24J40 2.4 GHz transceiver IC are power bypass capacitors. The PIC18LF4620 interfaces to the MRF24J40 using the MRF24J40's SPI interface (SDO, SDI, SCK, and CS). The MRF24J40 issues interrupt requests through its INT output line and can be awakened from sleep with a signal applied to its WAKE input line. The PIC18LF4620 resets the MRF24J40 by pulling the MRF24J40 RESET line logically low.

What you don't see in Photo 1 is the MRF24J40 daughterboard's 12-pin interface. The odd-numbered pins of the MRF24J40 12-pin interface are terminated with a six-pin female header spaced at the standard 0.1-inch pitch. The even-numbered pins are terminated with standard 0.1-inch pitch pins. If the PIC18LF4620 Host module's MRF24J40 daughterboard physical interface is installed correctly, this arrangement of male and female headers makes it very difficult — if not impossible — to incorrectly mount the MRF24J40 daughterboard. I caught a shot of the MRF24J40 daughterboard interface for you in Photo 2.

I assembled my MRF24J40 host modules using perfboard and point-to-point wiring. Note that I said "modules." You'll need at least two MRF24J40 host modules and a pair of MRF24J40 daughterboards to form a simple peer-to-peer 2.4 GHz network. My version of a MRF24J40 host module can be seen in Photo 3.

Initializing the MRF24J40 Host Module

Before we can do anything RF, we must configure the PIC18LF4620's general-purpose I/O pins in a fashion suitable to drive the MRF24J40 IC, the RS-232 converter IC, and the other stuff we have hanging off of the PIC18LF4620. Let's begin with PORTA.

PORTA initializes at reset to analog inputs. Since we have a couple of LEDs attached to RA0 and RA1, we need to configure PORTA as a digital port and assign RA0 and RA1 as outputs. This is easily done by writing 0x0F to the ADCON1 register and setting the I/O direction mask to 0b11111100.

PORTB of the PIC18LF4620 is look-

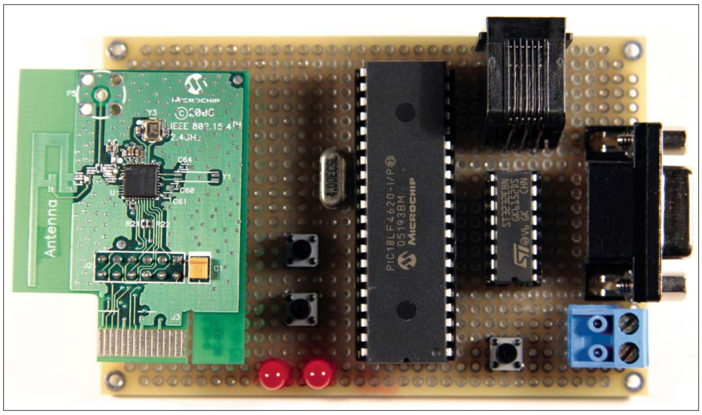


PHOTO 3. Nothing fancy here. It's just a standard PIC microcontroller system with a 2.4 GHz IEEE 802.15.4-compliant radio attached. This simple design has lots of uses including inter-robotic communications. If you're wondering where the passives are, they're on the other side in SMT packages.

ing for a falling edge on the INT0 (RB0) pin that would indicate that the MRF24J40 has issued an interrupt request. The only other application-oriented PORTB pins in use are RB4 and RB5, which are inputs for switches S1 and S2. So, all we need to configure PORTB to do is enable its weak pullups for the switches and look for a falling edge on the INT0 pin.

Writing 0xFF to the TRISB register takes care of putting all of PORTB's pins into input mode and clearing bit 7 of the INTCON2 register enables the individual pullups on the PORTB input pins. Clearing bit 6 of the INTCON2 register sets the INT0 interrupt input pin to trigger on the falling edge of the MRF24J40-generated interrupt request pulse.

The SPI interface is the target of the configuration of PORTC. The RS-232 port is also driven by PORTC, but we will initialize the serial port at another time in another function. The PIC18LF4620 SPI interface is configured to transmit on the falling edge of SCK and read on the rising edge of SCK. We also take the liberty of putting the MRF_CS chip select pin in a state such as to disable the MRF24J40's SPI interface.

PORTD and PORTE are not dedicated to the MRF24J40 and can be used for anything you wish. Thus, you can add the PORTD and PORTE configuration code to our MRF24J40 module initialization function:

```
void MRF24J40_Init (void)
{

    //Configure PORTA
    ADCON1 = 0x0F;      //all PORTA I/O digital
    TRISA = 0xFC;       //bits RA0 and RA1 = output

    //Configure PORTB
    TRISB = 0xFF;       //all PORTB pins = input
    INTCON = 0x00;      //disable global interrupts
    INTCON2 = 0x00;     //enable pullups-falling
                        //edge on INT0
    INTCON3 = 0x00;     //disable all other INTX
                        //interrupts

    //Configure PORTC
    TRISC = 0xD0;       //set up for SPI operation
    LATC = 0xFD;        //set MRF_CS=1
    SSPSTAT = 0xC0;     //transmit on falling edge
                        //of SCK
    SPCON1 = 0x20;      //enable SPI interface-idle
                        //state = low

    //Configure PORTD

             //Your code goes here

    //Configure PORTE

            //Your code goes here
}
```

Okay ... let's get our bearings. As of right now:

- We have established our PIC18LF4620 hardware platform.
- We have written our base MRF24J40 SPI I/O functions.
- We have written our MRF24J40 host module initialization code.

Once you build up the MRF24J40 host module hardware, you'll have a working PIC system capable of blinking the LEDs and sensing the switches, among other things. If you obtain your MRF24J40 daughterboards before we meet again, be very careful with your wiring of the MRF24J40 daughterboard interface as you won't be able to fix the MRF24J40 daughterboard with a soldering iron if you let the magic smoke out of it.

What's Next

Doublecheck your MRF24J40 module wiring as next time we'll push into initializing the MRF24J40 daughterboard. In the process of configuring the MRF24J40, we'll investigate the basics of IEEE 802.15.4 and determine what components of the IEEE 802.15.4 specification we will need to establish a communications link between our pair of MRF24J40 modules. When all is said and done in the next installment, we will have established a 2.4 GHz IEEE 802.15.4 link between our MRF24J40 modules and transferred data across the link.

Here's what we'll do when we return:

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

You can bet I'll be showing you the important operational stuff at the bit levels and you can bet we'll have some fun doing it. See you next time. **SV**

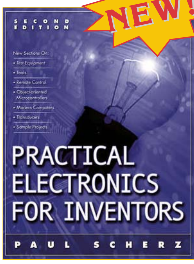
Sources

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

Mind Candy For Today's Roboticist

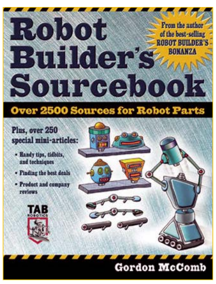
The *SERVO* Store

Practical Electronics for Inventors
by Paul Scherz



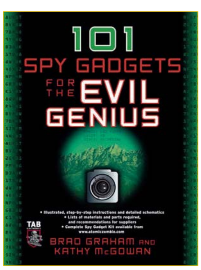
This intuitive, applications-driven guide to electronics for hobbyists, engineers, and students doesn't overload readers with technical detail. Instead, it tells you — and shows you — what basic and advanced electronics parts and components do, and how they work. Chock-full of illustrations, *Practical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

Robot Builder's Sourcebook
by Gordon McComb



Fascinated by the world of robotics, but don't know how to tap into the incredible amount of information available on the subject? Clueless as to locating specific information on robotics? Want the names, addresses, phone numbers, and websites of companies that can supply the exact part, plan, kit, building material, programming language, operating system, computer system, or publication you've been searching for? Turn to *Robot Builder's Sourcebook* — a unique clearinghouse of information for that will open 2,500+ new doors and spark almost as many new ideas. **$24.95**

101 Spy Gadgets for the Evil Genius
by Brad Graham/Kathy McGowan

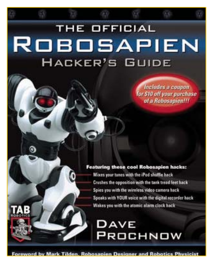


Utilizing inexpensive, easily-obtainable components, you can build the same information gathering, covert sleuthing devices used by your favorite film secret agent. Projects range from simple to sophisticated and come complete with a list of required parts and tools, numerous illustrations, and step-by-step assembly instructions. **$24.95**

SERVO CD-ROM

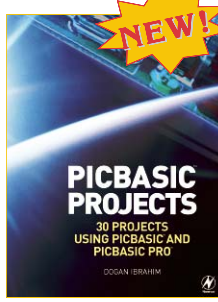
Are you ready for some good news? Along with the first 26 issues of *SERVO Magazine*, all issues from the 2006 calendar year are now available, as well. These CDs include all of Volume 1, issues 11-12, Volume 2, issues 1-12, Volume 3, issues 1-12, and Volume 4, issues 1-12 for a total of 38 issues all 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!**

The Official Robosapien Hacker's Guide
by Dave Prochnow



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

PIC Basic Projects
by Dogan Ibrahim



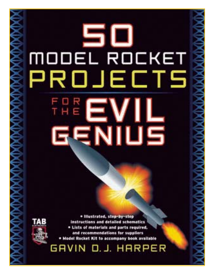
Covering the PIC BASIC and PIC BASIC PRO compilers, *PIC Basic Projects* provides an easy-to-use toolkit for developing applications with PIC BASIC. Numerous simple projects give clear and concrete examples of how PIC BASIC can be used to develop electronics applications, while larger and more advanced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

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

LEGO MINDSTORMS NXT Hacker's Guide
by Dave Prochnow

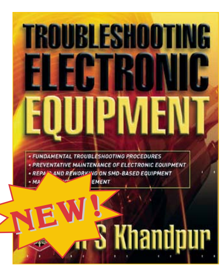
Here is an awesome next-generation collection of LEGO MINDSTORMS projects that enables you to build and program a real working robot in just 30 minutes! New technologies and expanded sensor capabilities make it easier than ever to add a level of sophistication to robotic and architectural creations.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

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**

Troubleshooting Electronic Equipment
by R. S. Khandpur

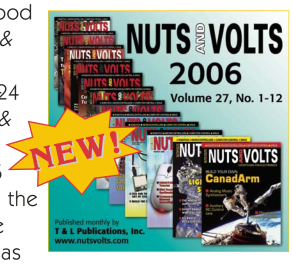


From cell phones to medical instruments to digital and microprocessor based equipment, this hands-on, heavily illustrated guide clearly explains how to troubleshoot, maintain, and repair all types of electrical equipment. The author covers all the essentials such as necessary tools, soldering techniques,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

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

Mind Candy For Today's Roboticist

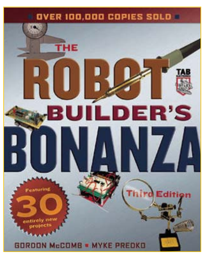
Nuts & Volts CD-ROM



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

Robot Builder's Bonanza Third Edition

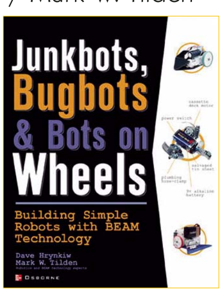
by Gordon McComb / Myke Predko



Everybody's favorite amateur robotics book is bolder and better than ever — and now features the field's "grand master" Myke Predko as the new author! Author duo McComb and Predko bring their expertise to this fully-illustrated robotics "bible" to enhance the already incomparable content on how to build — and have a universe of fun — with robots. Projects vary in complexity so everyone from novices to advanced hobbyists will find something of interest. Among the many new editions, this book features 30 completely new projects! **$27.95**

JunkBots, Bugbots, and Bots on Wheels

by Dave Hrynkiw / Mark W. Tilden



From the publishers of *BattleBots: The Official Guide* comes this do-it-yourself guide to BEAM (Biology, Electronics, Aesthetics, Mechanics) robots. They're cheap, simple, and can be built by beginners in just a few hours, with help from this expert guide complete with full-color photos. Get ready for some dumpster-diving! **$24.99**

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

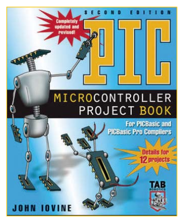
Take This Stuff and Hack It!

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**

PIC Microcontroller Project Book

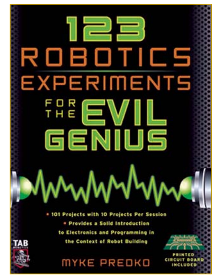
by John Iovine



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

123 Robotics Experiments for the Evil Genius

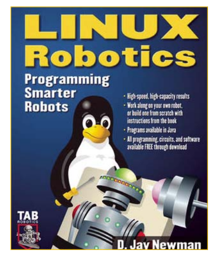
by Myke Predko



If you enjoy tinkering in your workshop and have a fascination for robotics, you'll have hours of fun working through the 123 experiments found in this innovative project book. More than just an enjoyable way to spend time, these exciting experiments also provide a solid grounding in robotics, electronics, and programming. Each experiment builds on the skills acquired in those before it so you develop a hands-on, nuts-and-bolts understanding of robotics — from the ground up. **$25.00**

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**

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

From HomoSapien to RoboSapien

Before R2D2 there was R1D1

SERVO Magazine T-Shirts

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

ROBO-LINKS

For the finest in robots, parts, and services, go to www.servomagazine.com and click on Robo-Links.

Build a Sensor That Locates the Nearest Object

by Jim Miller

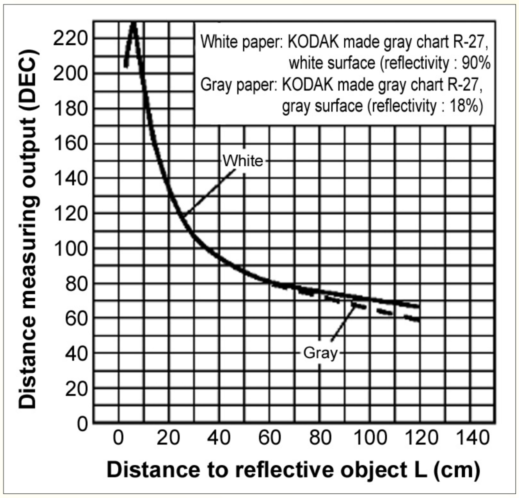


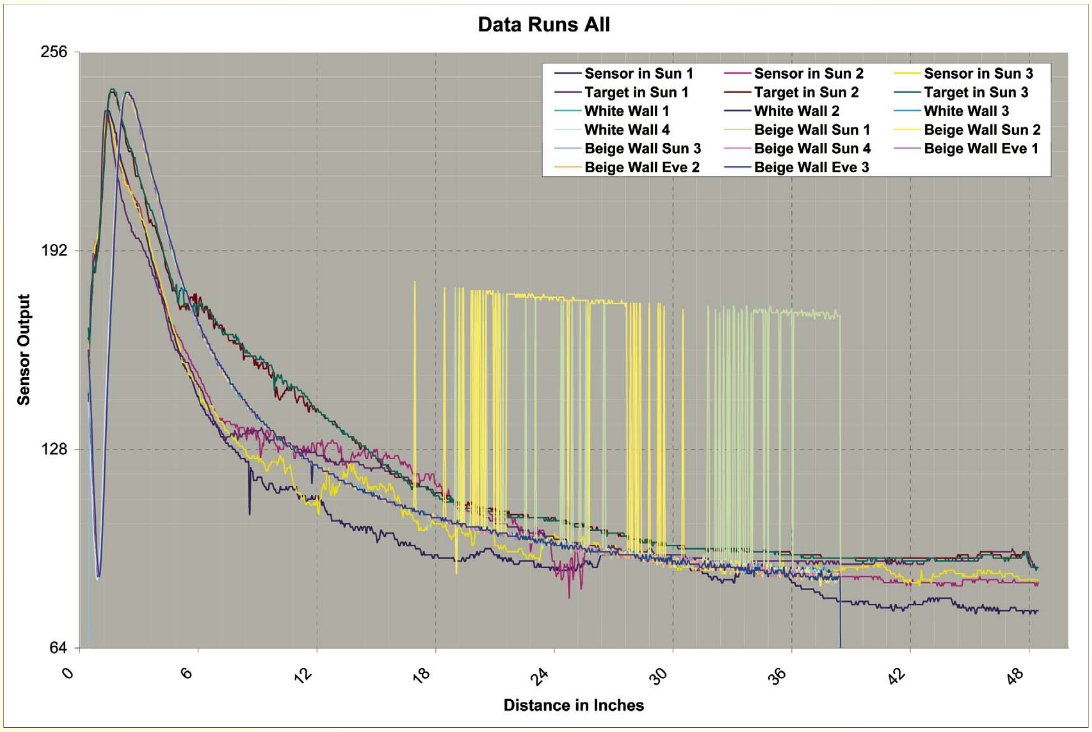
FIGURE 1. Distance Measuring Output vs. Distance to Reflective Object.

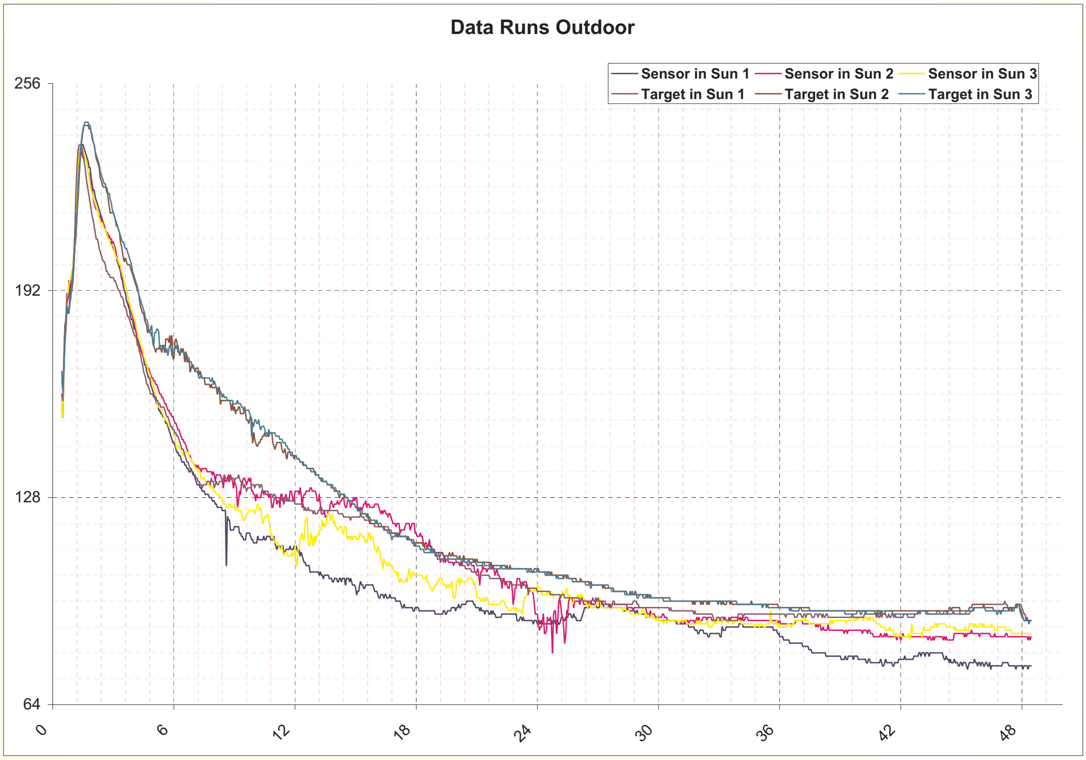
I have always been perplexed at the amount of data that one can get out of tech data sheets and still be left sort-of in the dark about what the gizmo actually does or how it really behaves. In this case, the Sharp GD2D02 is proclaimed to be an IR sensor that can determine the distance between it and an object. Its usable range is stated to be between eight and 120 cm or 3" to about 4'. As the distance grows, the accuracy declines. My ultimate plan for the sensor was a scanning device that could localize the nearest object.

To start, I built a test jig using a BiPOM 8051 22.8 MHz microcontroller, an Optrex 16x2 LCD display, and a Hitec miniature servo to sweep it side-to-side. The controller does everything including the PWM for the servo. The LCD display numbers are nice for troubleshooting, but the data jumps around a lot and, in hex format, it was pretty hard to understand, so I shipped it into an Excel spreadsheet to graphically understand the data. This got me a little closer to the data conversion formula but I needed to do some more work.

Okay, time to concentrate on conversion math.

The Sharp documentation





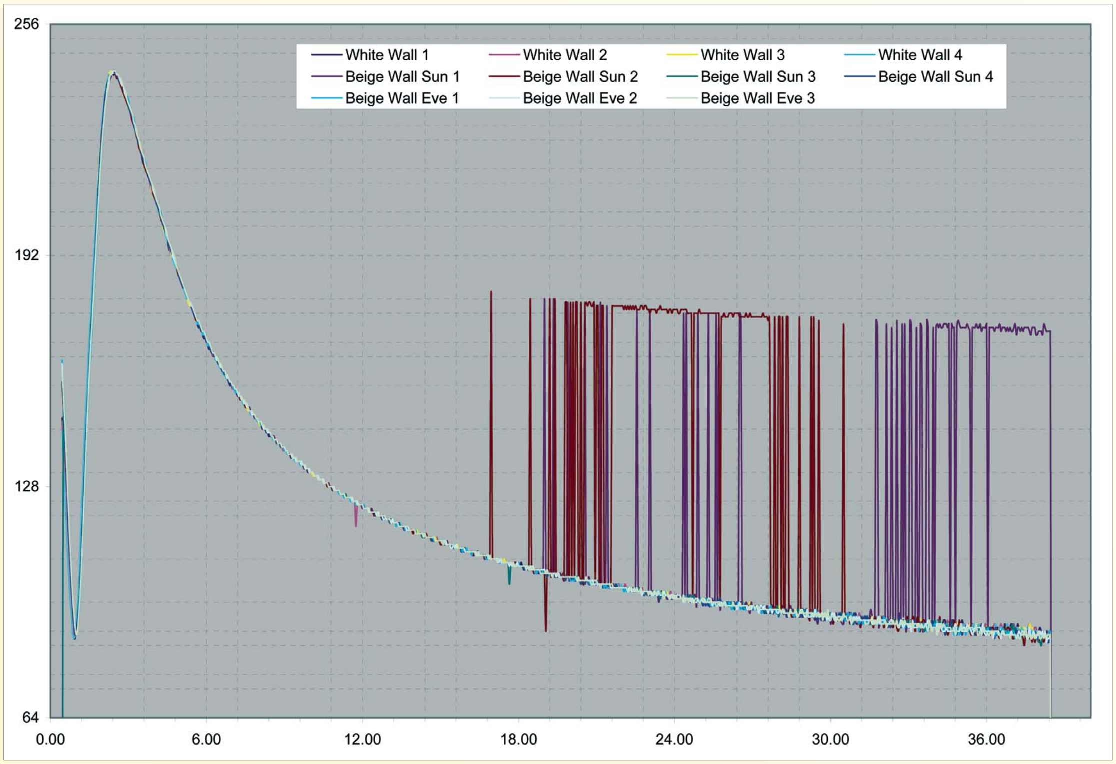
results on a radar-type display. The data jumps and bumps were much larger than expected so I carefully re-mapped the Sharp graph and did all sorts of math gyrations. I discovered that the distance measurements were inconsistent as the angle of the reflective surface changed. Intuitively, this is no surprise but certainly something to consider in your localization algorithms. Anyway, I broke the curve down into 10 separate line segments, but the output still just looked 'funny' (engineering term). To be useful, I was going to need more than the low resolution graph with an interpolated conversion

on the sensor is confusing. The calibration curve data has the axis reversed. Sensor data is on Y and distance is on X. One would think given data is on the X axis and the calculated distance on the Y. Confused? It gets better. The sensor works by measuring the angle of the reflected light, as the object moves further away, the data output gets smaller. Smaller output = longer distance. A final twist is that it's all in metric units. Ultimately, the graph and the data are counterintuitive so it's hard to get your head wrapped around pure numbers.

I wrote some VB code to map function. (In the movie available at **http://cannibalrobotics.com/ir-ranger.htm**, you can see glimpses of the radar display.)

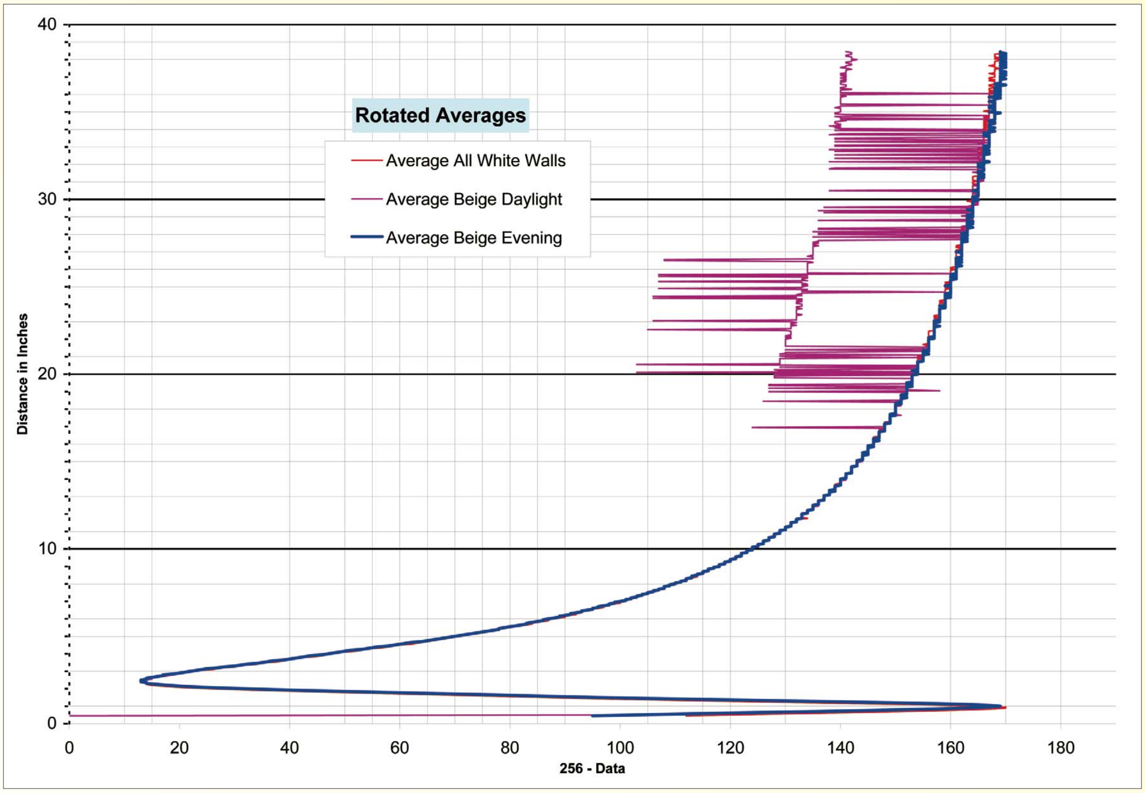
A look-up table was going to be the answer. For less than 255 data points, this would probably be even faster than math anyway. I poked around for awhile on the Internet to find a point-to-point conversion table, but this yielded nothing. To make my own table, I would need some controlled tests. To do that, I'd need a way to accurately control the distance from the sensor to the surface. Ah ha! A chance to build something!

I put the whole sweep apparatus on a little car track fashioned from an aluminum vertical blinds track I'd saved. (See honey, I told you, all this junk will save us some money someday.) I moved the apparatus along the



track with a 1/4-20 all-thread worm gear drive. The drive is motivated by a MAXCNC stepper motor I bought years ago, and I dusted off the old Model 100 computer to step the motor.

The setup worked well and generated more questions than answers. The data prompted me to look at different environmental conditions, such as dark, sunlight, different color walls, etc. I moved the apparatus around to different environments. I gathered up a great data set and made several application discoveries along the way. In Graph 2, outputs are the vertical, and distance in inches runs horizontal.

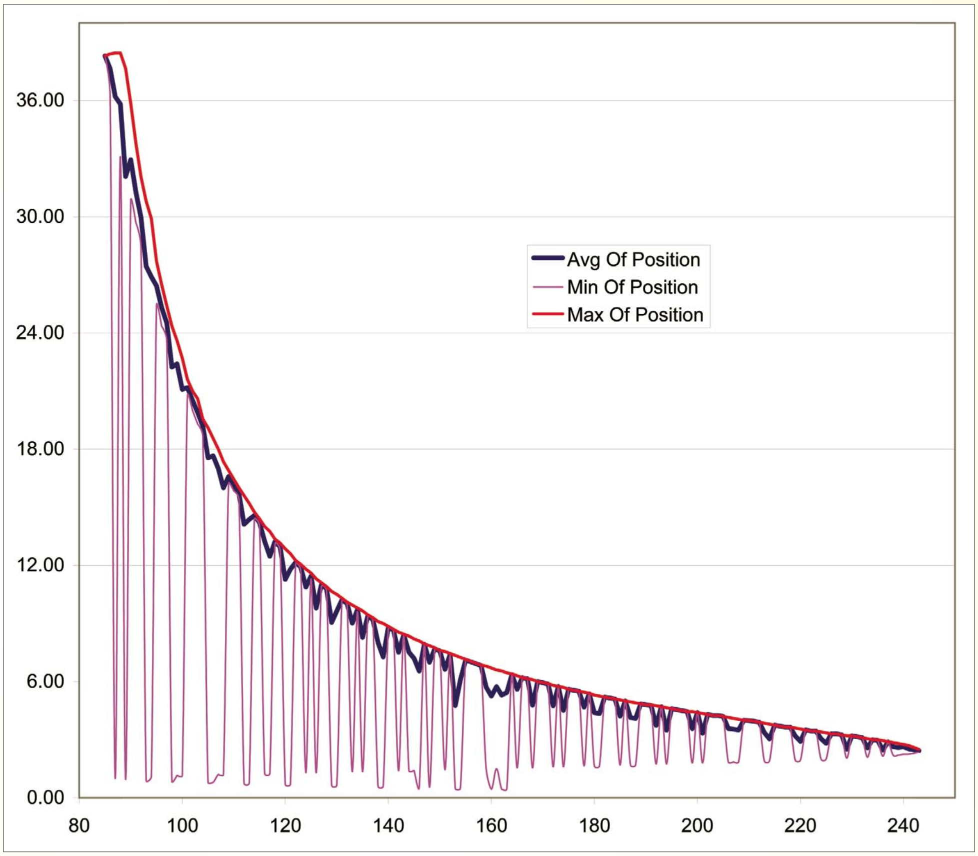


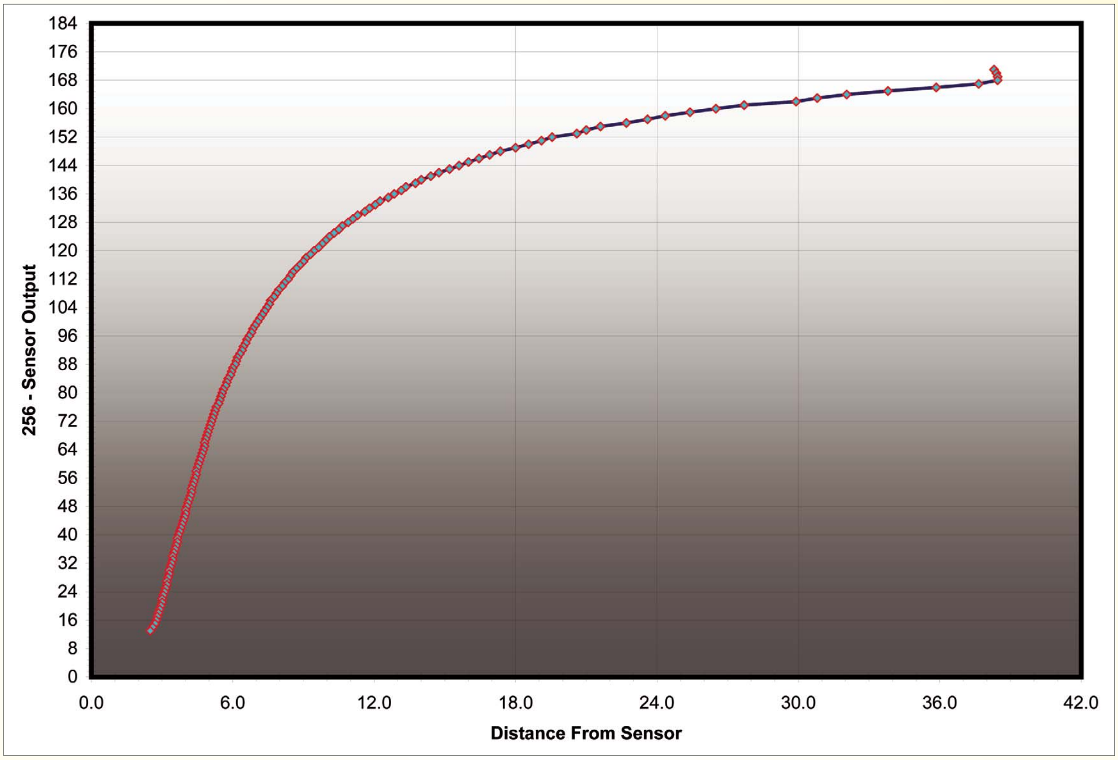
Enlightenments included:

Shiny surfaces tend to enhance a droop in the data set in the very near range. The three runs labeled 'Shop #' were done against a flat shiny side sheet of aluminum foil. The duality in the data between two and four inches appears in all data sets but it's more pronounced on shiny surfaces. The solution for this duality is simple: Place the sensor at least 4" away from the edge of your robot or just ignore that range. But, shiny surfaces are an enigma; they appear an inch closer, go figure.

The sun is pretty cruel to the sensor. If the target is in the sun, your 6-24 inch accuracy is ±4". If the sensor is in the sun, you lose accuracy in the 8-24" range, as well. Remember, when looking at the graph, the accuracy of the sensor is the horizontal disparity between the lines.

I took the gizmo into the house and started taking data there. Surprisingly enough, the indoor data was both better and worse! From two to 16 inches, the data was consistent and relatively free of noise. In general, it was reliable but there is just something about beige walls. On a beige wall when the curtains are open on a sunny day, I got a huge series of spikes. On a white wall in the same





room, there were no spikes.

I now had a better understanding of the noise in the sensor and environmental effects, but I was not getting closer to the data set for a robot. So, I averaged out the data from the interior walls, rotated the axis, and inverted the sensor data so what you see is (256 – data). Now it looks like a graph I understand, trending up from left to right with data on X and distance on Y. As the data value goes up, the distance goes up. I concluded that from a stability point-of-view, it looks as though indoors, your accuracy is pretty decent from about 3.5" to 16.5". After that, it's going to depend on how much you know about the ambient light and the color of the reflective surface. All of the noise beyond that was troubling.

To eliminate the noise, the final test and data set involved a new collection process. No matter where the sensor is positioned, the data is erratic. The original data collection involved stepping to a position, then getting a sample and moving on. I rewrote the driver routine to step to a position and collect five data. My assumption was that averaging the data would give me a reasonable result. I was closer, but wrong.

As you can see, averaging the data is not going to give the optimum data set. It is still bouncy. While playing with the data, it was discovered that the maximum positions out of all of the data gives a nice smooth curve. Voila! A rule! We've arrived: the sensor consistently errors closer. Or, take several samples and the furthest away will be the most accurate. (Remember, this will be the smallest value you get back from the sensor because of the inverse relationship between distance and sensor data. At the end of the day, this turns out to be a safe side error if used for collision detection.)

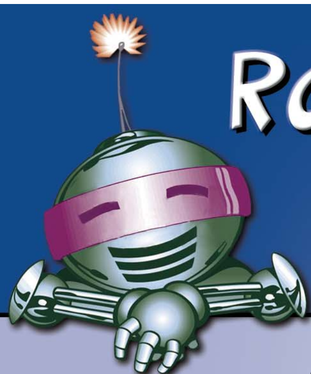
Finally, I arrived at the data set I needed. To apply it, use these simple rules:

1) Filter the data, get 5+ samples, and use the lowest value.

2) Subtract that value from 256.

The distance from the sensor can then be "reliably" determined from the table and curve shown.

Since we started with a curve and found out we needed a table, I'm not going to leave you hanging. The look-up table and code snippets are on the *SERVO Magazine* website (**www.servomagazine.com**) as well as mine (**www.cannibalrobotics.com**). Email me at jin@cannibalrobotics.com and let me know how it goes. **SV**



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

I Can Get it For You Wholesale!

Not long ago, "buying wholesale" meant going downtown to some non-descript warehouse, pulling your car up to a loading dock, and looking for ways to jam a bunch of huge boxes into your family sedan. Today, you can still find plenty of wholesale warehouses in most any downtown business district, but the landscape also includes convenient Internet shopping, as well as some of the world's largest retailers who sell at distributor prices to the public.

Why bother buying wholesale? The savings, of course. It costs less because you are buying more of something. A distributor or retailer passes along savings to you because you are buying in volume. What those savings are depends on the wholesaler, the merchandise, and the quantity, but it's not unusual to pay 20-50 percent less at wholesale quantity prices.

In this months' column, we'll talk about how to find wholesale goodies for your robot building. Some of these sources may be just around the corner, and a few may be across vast oceans. Either way, the potential savings can be significant, and are worth considering if you're looking to buy several of the same item, or are looking to purchase in bulk for a robot user group or school.

Local Wholesale Retailers

The second largest retailer in the United States is Costco Wholesale. Costco is open to the public and has over 400 "warehouse stores," with the bulk in the US and Canada (some are in Japan and other Asian countries). They are open to the public via a membership program, which requires an annual fee. Food and clothing make up a large part of the products sold by Costco, but there's also batteries, tools, and other merchandise of particular interest to us robot builders. The batteries come in large case packages — eight nine-volt batteries instead of the usual one or two sold in the typical retail package; the tools are sold in sets. You save by buying more.

Costco may be the largest of its kind, but it's certainly not the first wholesale membership retailer, nor is it the only one you may find. For example, Sam's Club — a unit of Wal-Mart — provides the basic service. There's also BJ's Wholesale Club, which operates several large warehouses in the eastern part of the US.

Although local Costco, Sam's Club, and BJ's Wholesale Club stores emphasize local shopping, all three have Internet sites, where you can get a sampling of the wares and the prices. Check out the Sources listing that follows for company URLs.

Besides these well-known wholesale retailers, your area may be home to a smaller wholesale chain, or even a one-of-a-kind non-food wholesale store. For example, for several decades (don't want to say how many!), I've shopped Star Restaurant Equipment and Supply in the Los Angeles area — like many of its kind, this store is geared toward serving the restaurant industry, but it's also open to the public. Before you think a restaurant supply store is an odd place to buy robot parts, visit one, and you'll see all sorts of interesting and unusual things. I've used stainless steel bowls and sheets to build inexpensive bodies, two ounce sauce cups for holding nuts, bolts, and other parts when building, kitchen utensils for cheap but effective robot bumpers, whiskers, and other parts, and lots more.

Your local Yellow Pages (or equivalent) is the best source for finding local wholesale retailers. Look under Wholesale, as well as the individual headings of things you wish to purchase: plastics for a plastics wholesaler, electronics for an electronics wholesaler — you get the idea. While searching on Google may be tempting, in fact, most local businesses without a hefty web presence are under-represented on Google and other online search engines, and you'll likely miss some great opportunities.

eBay (and other Online Auction) Wholesale Distributors

As the popularity of eBay and other online auction sites have grown, so have online distributors of wholesale products. These companies provide a way for individuals and small businesses to purchase at a quantity discount, often without the need for a state resale license — which is the case of the typical local wholesale warehouse. (However, if you do not have a state resale license, you must tack on state sales taxes, if applicable.) They ship directly to you, using common carriers like UPS and FedEx, or for larger orders a trucking company.

Among the largest online wholesalers is DollarDays, which offers so many products, you can spend several days browsing their catalog, and still not see everything. Fortunately, their website is divided into categories, such as General Merchandise and Electronics. Just about everything is sold in case packs; the quantity in a case varies depending on the item, but a case usually contains anywhere from 6 to 144 items. For example, if you're interested in the "hot Lips Talking Radio" sold on DollarDays — yes, large red lips that move with the sound on the radio! — they can be yours, but only if you buy 50 of them at a time. Other merchandise, like tools, batteries, small personal electronics, and toys carry similar minimum case-lot requirements.

Competitors to DollarDays include DollarItemDirect, Oriental Trading Company, Price King, Kole Imports, and Sav-on Wholesale. You'll find many more on any good Internet search engine, but exercise caution. There are a number of companies that merely sell you "where to find it" booklets, and don't themselves sell any product. You can find most or all of the companies listed in these guides yourself, if you are willing to do the legwork.

The online wholesale business is highly competitive, and many of the companies do not list their prices until you register with them. Avoid those that require a payment or credit card details in order to register. There are plenty of others that do not impose a fee. Be especially mindful of minimum orders. Some online wholesalers have no minimum order, but may tack on a substantial handling charge to each order, thereby negating any savings over purchasing the goods at a regular outlet. Others have minimum orders ranging from $50 to $500 or more.

Online Distributors That Offer Quantity Discounts

Many online retailers and distributors offer attractive prices if you are willing to purchase in quantity. Some list their quantity purchase prices right on their web pages, but many others do not. If bulk pricing is not explicitly stated, look for more information on the company's discounting policies in a FAQ (Frequently Asked Questions) page or on their ordering pages. If you still can't find anything definitive, dash off an email, or call them on the phone.

A good example of an electronics distributor that sells at discount quantity pricing is Digi-Key. Their online catalog lists the quantity price breaks for many of the products they carry. The price breaks are typically at quantities of 25, 100, and 1,000. Buying 25 or even 100 of some electronic component is not always a daft proposition, especially if you can arrange a group purchase with your local robot user group or school.

For example, at the time of this writing, Digi-Key was selling the AVR ATmega 16 microcontroller for $6.56 in single quantities, but $4.12 in quantities of 25 or more — a savings of about $2.50 each. While your application you may not require 25 chips, if you need more than a couple, you may want to combine your order with someone else's, so you can enjoy the cost benefits.

As a final word, consider that wholesale has its place, but it's not the answer to everything. Wholesale almost always entails buying in quantity. You'll be wasting money if you don't need the higher quantity. If you can't share or resell the excess, it's better to buy the product in singles through a retailer, and look for savings elsewhere.

Sources

All Electronics
www.allelectronics.com
All Electronics is one of the primary sources in the US for new and used robotics components. Prices and selection are good. Several items available with deep quantity discounts.

Allied Electronics
www.alliedelec.com
Allied Electronics is a prime source for all electronics (components, chemicals, tools, you name it), in single or multiple quantities. Their prices are often lower than the competition's. Do note the minimum order amount.

Arrow Electronics, Inc.
www.arrow.com
Arrow Electronics distributes a full line of electronic components to industry. Products can be located by manufacturer, part number, or category.

Avnet, Inc. (Avnet Electronics)
www.avnet.com
Full line distributor with local offices worldwide. Refer to the website for locations.

BJ's Wholesale
www.bjs.com
Wholesale warehouse chain in the eastern US. Similar to Costco.

Costco Wholesale
www.costco.com
Costco is the world's largest wholesale warehouse retailer. They are open to the public, and require a membership fee (certain membership requirements may apply, and they have different membership levels).

Digi-Key
www.digikey.com
Digi-Key is one of the largest mail order retailers/distributors of electronic components in North

America. Discounts are often (but not always) available when buying 25, 100, or more.

DollarDays
www.dollardays.com

DollarDays is a leader in online auction, dollar store, and flea market wholesalers, DollarDays offers everything from baby bottles to radios to tools. All merchandise is sold in case quantities.

DollarItemDirect
www.dollaritemdirect.com

Purchase "dollar store" items by the case.

Future Electronics
www.futureelectronics.com

Large electronics distributor. Mostly for industry, but they will also sell to individuals.

Jameco Electronics
www.jameco.com

Full service general electronics mail order: resistors, capacitors, ICs, transistors, diodes, etc. Quantity discounts available for most products.

Kole Imports
www.koleimports.com

Kole is a primary wholesaler to dollar stores, flea markets, and other discount retailers. You must register to see all the prices, and there is a hefty minimum order amount. But the prices are excellent if you are looking to buy several dozen of something.

Mouser Electronics
www.mouser.com

Mouser Electronics is a mail order components distributor, providing a full line of products for industry and hobbyists. Many products available at a quantity discount.

Newark Electronics
www.newark.com

Prime-component distributor to business and industry; also caters to hobbyist market, but minimum orders may apply. Sells large quantities, when available.

Oriental Trading Company
www.orientaltrading.com

Oriental Trading Company serves the toy "premiums" business. Much of their product is inexpensive small toys for parties and party favors, kid crafts, schoolrooms, and fundraisers.

Price King
www.priceking.com

Price King is an online wholesale distributor, with a heavy-emphasis on household products. Catalog pages indicate merchandise that may be shipped via UPS (usually cheaper than a trucking company).

Sam's Club
www.samsclub.com

Wholesale membership club, operated by Wal-Mart.

Sav-on Wholesale
www.sav-on-wholesale.com

Internet-based wholesaler specializing in low-cost dollar store merchandise.

Star Restaurant Equipment and Supply
www.starkitchen.com

This LA-based wholesale retailer is a good example of the unusual merchandise you may find at this and similar restaurant equipment supply stores. **SV**

ABOUT THE AUTHOR

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

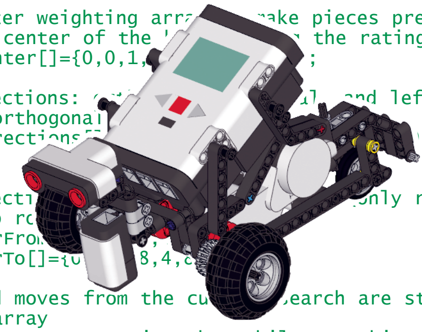
```
// castling bonuses
B8 castleRates[]={-40,-35,-30,0,5};

//center weighting arr    ake pieces prefer
//the center of the       the rating routine
B8 center[]={0,0,1,         ;

//directions:               and left/right
from orthogonal
B8 directions                  ,-10,1,-
1};

//directi                   only really for
bishop rc
B8 dirFro
B8 dirTo[]={               8,4,

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



A bi-monthly column for kids!

LESSONS FROM THE LABORATORY

NXT Robotics: Remote Control

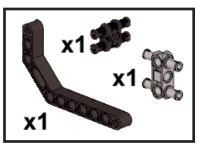
by James Isom

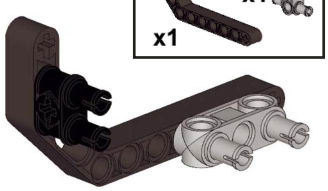
Here's a new NXT robot design I've been playing with. It's pretty flexible and fairly simple to build. Have fun building the GOBOT ROVER.

CHASSIS INSTRUCTIONS

STEP 1:

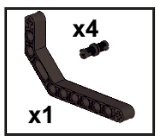
Parts:





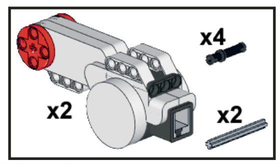
STEP 2:

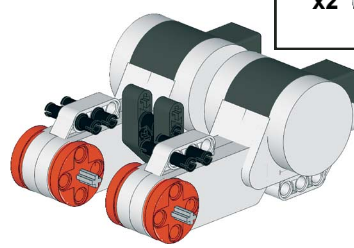
Parts:



STEP 3:

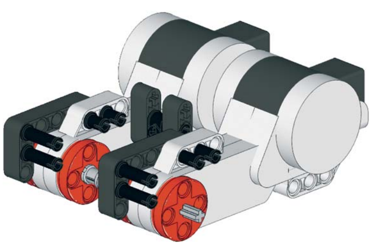
Parts:





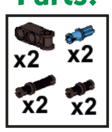
STEP 4:

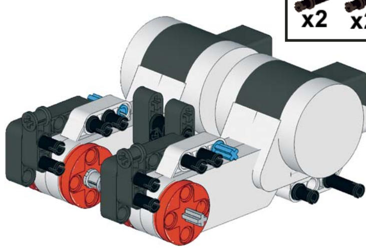
Parts:



STEP 5:

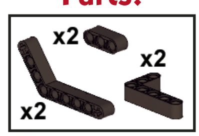
Parts:

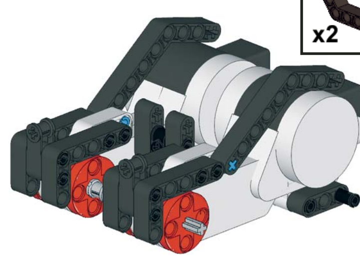




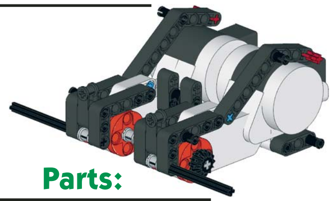
STEP 6:

Parts:

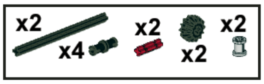




STEP 7:



Parts:

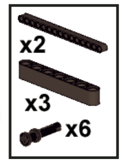


STEP 8:

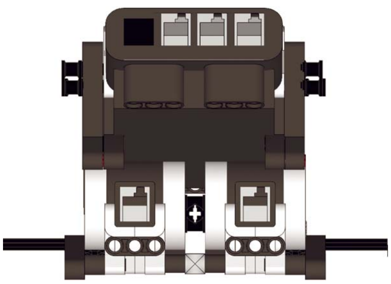
Parts:

STEP 9:

Parts:

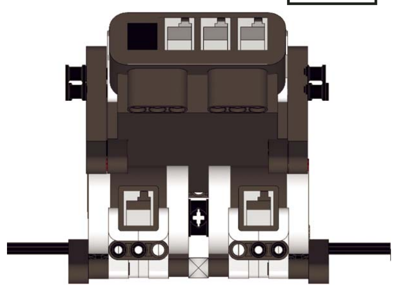


STEP 10:



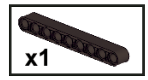
STEP 11:

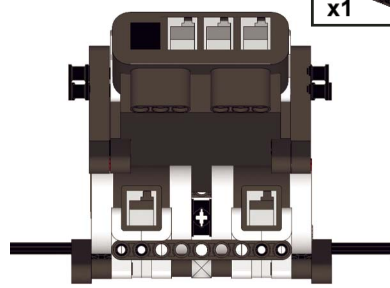
Parts:



STEP 12:

Parts:

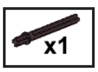




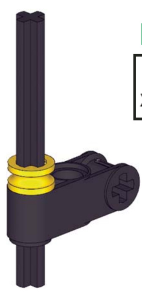
CASTER INSTRUCTIONS

STEP 1:

Parts:



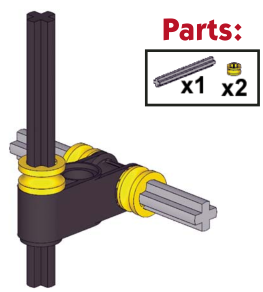
STEP 2:



Parts:

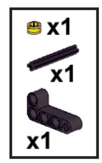
STEP 3:

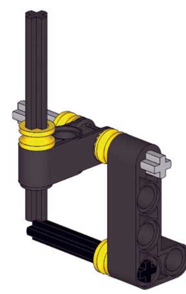
Parts:



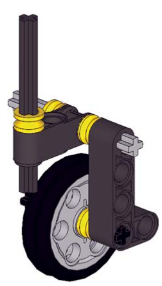
STEP 4:

Parts:

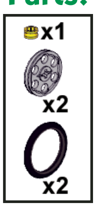




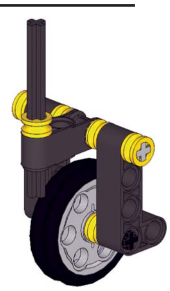
STEP 5:



Parts:

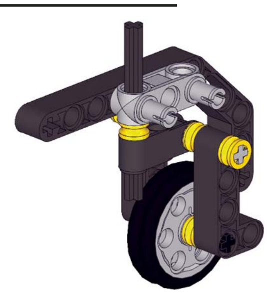


STEP 6:

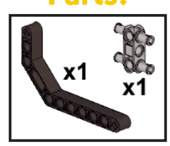


Parts:

STEP 7:

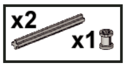


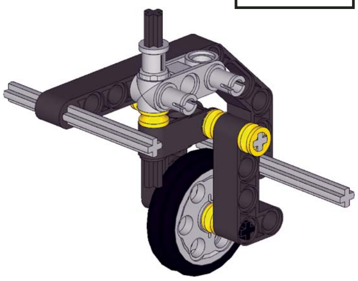
Parts:



STEP 8:

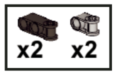
Parts:

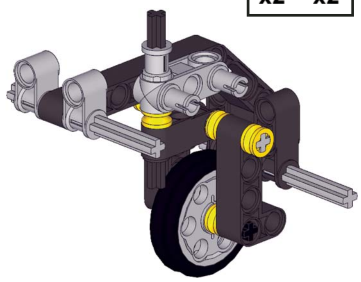




STEP 9:

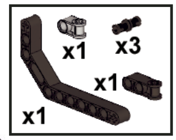
Parts:

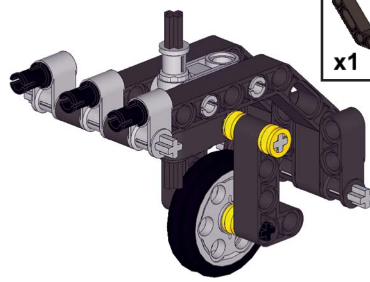




STEP 10:

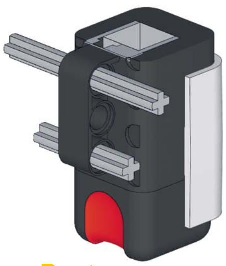
Parts:



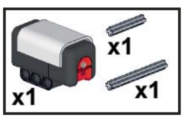


LIGHT SENSOR INSTRUCTIONS

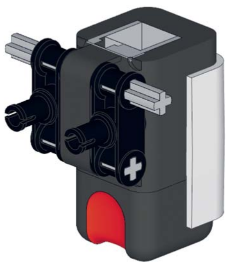
STEP 1:



Parts:



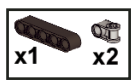
STEP 2:



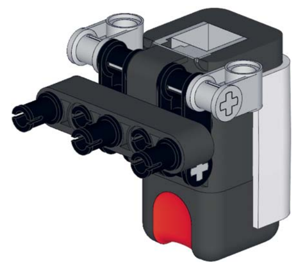
Parts:

STEP 3:

Parts:



STEP 4:

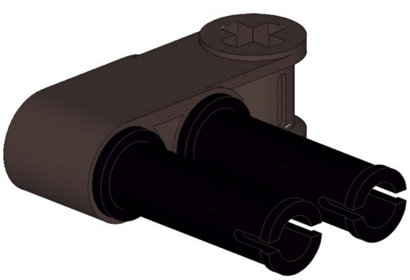


Parts:

ULTRASONIC SENSOR INSTRUCTIONS

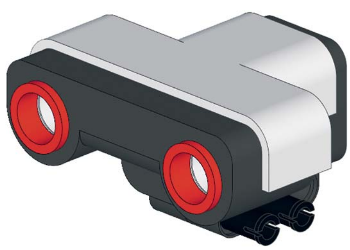
STEP 1:

Parts:



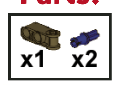
STEP 2:

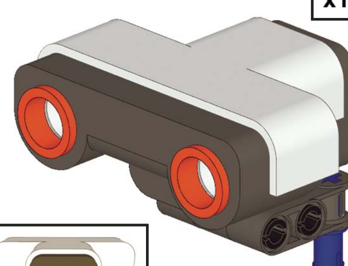
Parts:

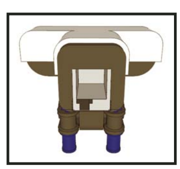


STEP 3:

Parts:



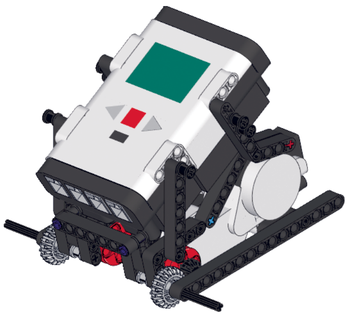




BACK

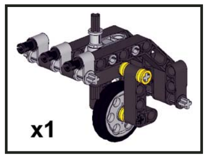
FINAL ASSEMBLY INSTRUCTIONS

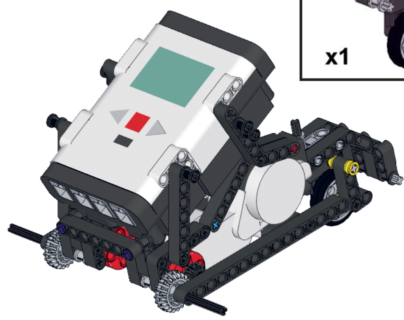
STEP 1:



STEP 2:

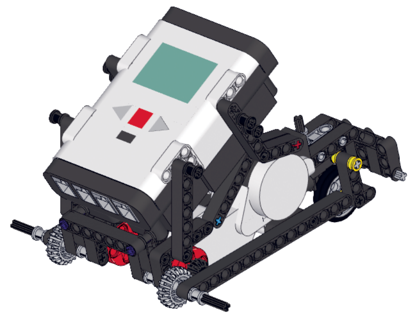
Parts:





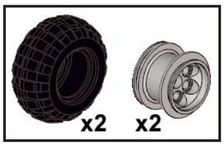
STEP 3:

Parts:



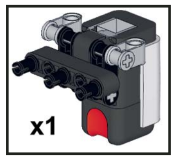
STEP 4:

Parts:



STEP 5:

Parts:



STEP 6:

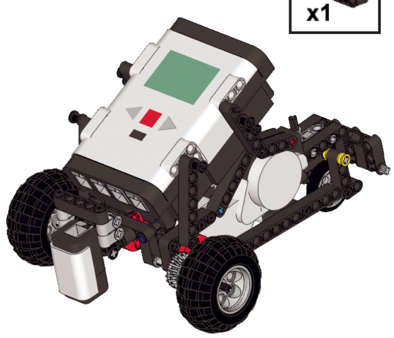
Parts:

STEP 7:

Parts:

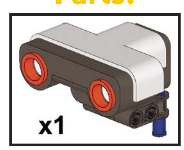
STEP 8:

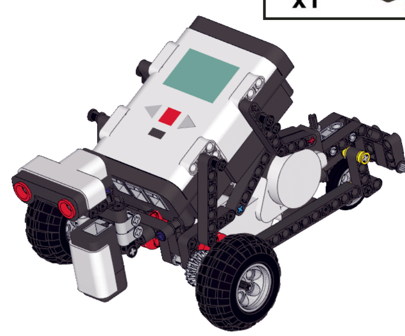
Parts:



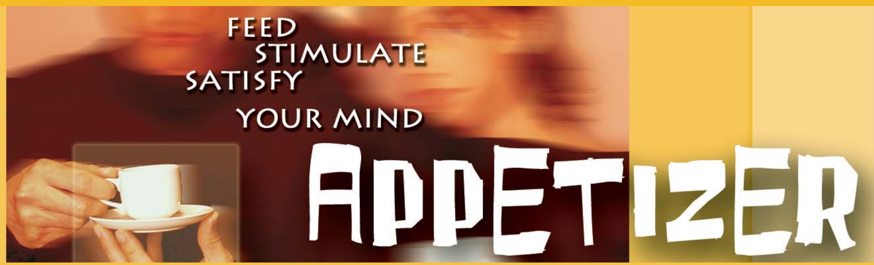
STEP 9:

Parts:





The GOBOT ROVER can be configured in a variety of ways for various activities. Check out some of the options at **www.LEGOedwest.com** in the Resources section under Tutorials. Have fun! **SV**



Precisely What Your Robot Needs

by Chris Harriman

If I offered you a choice of making your autonomous robot either 100% accurate or 100% precise, which would you choose? Precision, I hope.

A quick trip to **Wikipedia.com** will tell you that accuracy is the degree of veracity while precision is the degree of reproducibility. What does that mean and why should you care? Well, if you're a target shooter or a robotics guy with an upcoming autonomous maze event, this will apply to you.

Consider a rifle shooting at a paper target (see Figure 1). Accuracy would be the ability of hitting the center of the target while precision would be the ability to reproduce the same shot while aiming at the same spot. A rifle can have its sites adjusted to correct for poor accuracy, but there is no adjustment to correct for poor precision.

Just like a rifle, things that can affect the accuracy of a robot are usually easily adjusted; misaligned drive wheels or bad pulse width calibration in your code would be good examples. Poor precision, on the other hand, is usually not so easily corrected. Here is a quick list of the most common causes of poor precision in small robots:

FIGURE 2

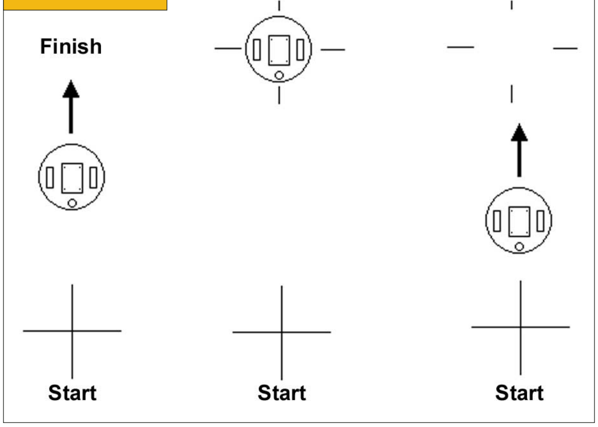


FIGURE 1

1) *Using doublesided tape or Velcro to mount your drive motors*. Your motors will move during acceleration and deceleration injecting their own random steering input.

Fix: Solid-mount your motors with screws or even glue, if need be. Get the flex out!

2) *Using a plastic chassis*. Plastic can allow flex during acceleration and deceleration. Also, many plastics will expand or contract as much as 5% with as little as a 5 to 20 degree change in temperature.

Fix: Use metal or thicker plastic.

3) *Wheel slip*. Not only can foam wheels slip at the point of contact with the surface, but some can also slip between the tire and the rim.

Fix: Use the correct tire material for the given surface and/or glue the tire to the rim.

4) *Poor idler wheel design*. An idler wheel that wiggles (even a little) while your robot is moving straight is injecting random steering input into your drive system.

Fix: Use a fixed idler such as a nylon post or ball. A roll-on ball idler may work, as well.

5) *Poor electrical connections*. Poor electrical connections can provide random amounts of current to your drive motors or feedback from your encoders.

Fix: Use good soldering practices; securely mount all your robot's electronic components.

Test Your Bot

To perform a precision test on

your robot, mark its starting position with tape, then have it travel in a straight line for a set period of time, say 10 feet or so. Mark its stopping point with tape. Now, put your robot back into the same starting position and have it repeat the same move for the same period of time and record its new stopping position, relative to the first one. Make sure to place the marker tape in an area that the robot's wheels won't run it over; that may mess up your results. Repeat this test five to 10 times, marking the difference in the stopping position from the first stopping position (see Figure 2).

This will give you a good idea of your robot's precision, just like the rifle target. The closer together your robot's stopping points are, the better its precision.

Hopefully, these quick tips will help put your robot in the winner's circle — the *exact center* of the winner's circle. SV

Then and NOW

ROBOTICS EDUCATION

by Tom Carroll

People have often mentioned to me just how much robot education has changed over the years. I've been in the field long enough to see the many changes and the new levels that academia now encompasses in robotics classes. I've taken my share of classes in various aspects of robot technology and even taught a few university level classes, but this has only shown me just how much I don't know about robotics, or maybe just how much more there is to know. The overall field certainly has changed over the years.

Needless to say, it is the power of computers, or should I say *microcontrollers* that are available to experimenters and students today that have changed the directions of classroom instruction. But what about the desires of students, you may ask? What new directions are they looking to in their quest for knowledge in this fairly new field?

Just who are these students? A vice president of production whose CEO has just told him to "implement robots in our factory to keep up with competition, increase production, and lower costs?" A young engineer-to-be in college? The brightest guy on a production line who has just been given the responsibility of operating a robot? A home experimenter who has a thousand ideas for a robot but has no idea where to start?

The Beginning of the Robotics Age

Before the 60s, robots were the staples of "B" movies and there were just a few experimenters like Grey Walter and his robot tortoise whom I wrote about in the October '05 issue of *SERVO*. Figure 1 shows a replica of his simple, but first-of-its-kind 'intelligent' robot. When George Devol and Joe Engelberger developed the first industrial robot in the early 60s, suddenly it was the "robotics age" and magazines and TV shows plastered us with images of these crude, tank turret-like machines welding and painting cars.

Other machines were snaking their metal arms into presses to retrieve hot metal forgings. Unions may have been a bit worried about workers being *replaced*, but the workers welcomed their metallic brethren to assist them with unpleasant and dangerous tasks. Robots were now a reality and not just strange creations running around the floors of university labs. Very few people knew anything about them, as there were no classes available to educate the rest of the world about how to build one or what they consisted of.

Robots had minimal 'intelligence'

Figure 1. Grey Walter's Tortoise.

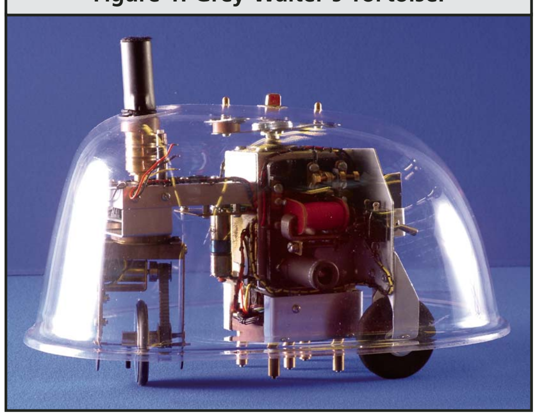


Figure 2. Hero 1 Robot.

Figure 3. Hero 2000.

as computers of those days were huge room-filling machines and early robot 'innards' could hardly be called a computer. The early Unimate was controlled step-by-step by commands stored on a magnetic drum and inputted by 'skilled machinists,' rather than engineers or people skilled in computer technology.

The only definition for a robot in those days was that it "... is a reprogrammable, multi-functional manipulator designed to move material, parts, tools, or specialized devices, through various programmed motions for the performance of a variety of tasks" ... certainly not what we experimenters think of as robots today. Microprocessors were still a decade away and affordable 'computers' still further in the future. How was one to learn about the new technology?

Early robotics instruction was performed by the robot manufacturers themselves and was tied closely to computer-aided manufacturing, the 'CAM' in CAD-CAM. These were people with previous experience in CNC machines of the 1950s — computer numerical controlled tools in a factory. Factories soon progressed from punched paper-tape instructions for CNC machine tools to factory automation with robots placed in work cells along a production line. Instructors and students alike soon realized that they needed to know about the entire manufacturing process, not just a single operation cell.

Education requirements increased dramatically, as did interest from other segments of society that felt that they could implement robotics somehow. Community colleges began to offer robotics classes that centered on installation, setup, and programming of the more popular robots such as the Unimation series and the larger Cincinnati Milacron machines.

Upper division university classes began to center on basic robot design, kinematics, and dynamics of continuous path robot motion, specialized end effectors, and control algorithms. Carnegie Mellon University and MIT began a series of 'hands-on' courses that were extremely popular with students and among the most thorough robotics courses in the world. This is fine and dandy for those lucky enough to be near one of these schools, but what was and is available for the rest of us?

Heath Introduces Robotics Courses

Back in the 80s, the Heath Company — makers of the many types of Heath kits — took the initiative to design and market a robot kit that was tied to an extensive robotics educational series of courses. I was fortunate to be given a course and a completed Hero 1 robot by the Heath representative in Southern California to "evaluate." I was quite impressed with the detail and completeness of the course materials and I learned quite a bit about all facets of robotics. I had to give the robot back (rats!).

Our Southern California Robotics Society meetings were then held in the Norwalk Public Library and Heath managed to convince quite a few of our members to part with their hard-earned cash for the original kit and course (Figure 2) The Hero 1 had a fairly flimsy arm but the continuous path movements that could be achieved by the robot were the important aspect of the course: the payload capacity. The little Hero Jr. had no arm and seemed to

Figure 4. Teachmover1.

Figure 5. Rhino xr3.

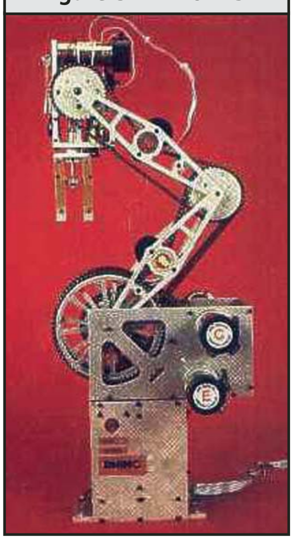
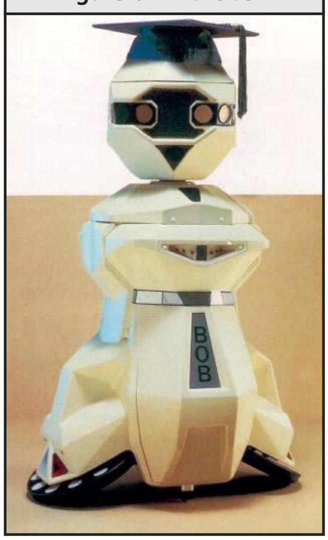


Figure 6. Androbot.



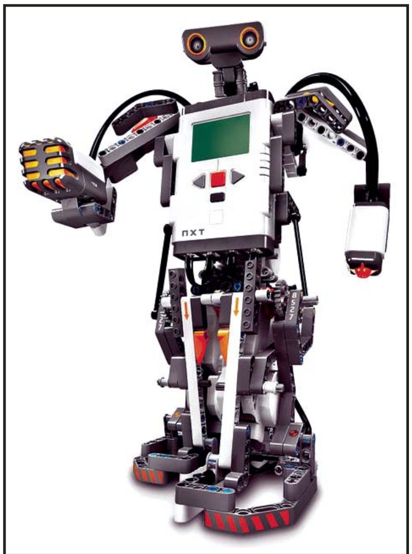


Figure 7. LEGO Mindstorms NXT.


Figure 8. Robotix class at Orcas Island School.

be more of a programmable toy than a multi-functional robot.

As the *SCRS* grew and we changed our name to the *Robotics Society of Southern California* by the late 80s, Heath evolved also and came out with a greatly-improved educational robot — the Hero 2000 (Figure 3). Heath generously loaned me one of these remarkable machines for a short while along with an improved course. I still have both of the course series that I completed and still refer to them to this day. They truly contained a lot of useful information about the field of robotics.

As you can see from the figure, the Hero 2000 has an arm that is not only more robust, but designed more like an industrial counterpart — important for teaching robotics in those days. Other 'educational' robots of the 80s — the Microbot Teachmover 1 (Figure 4) and the Rhino Robotics Rhino XR3 (Figure 5) — were fashioned after the 'typical' industrial robot, with the Androbot TOPO and BOB series (Figure 6) following a more 'youthful' form factor.

The few Androbot BOBs (BOB: Brains on Board) that were sold had three Intel 8088 processors and a hefty three megabytes of RAM — pretty good for those days. TOPO was RF-linked to an Apple II computer and used the Logo language, much like the Tasman Turtle, another 'educational' robot of the mid 80s. After all, the only jobs in robotics in the late 80s and early 90s were in industrial positions or a few positions designing experimenter's robots such as the machines mentioned above. There were no battlefield or police robots, no Roombas or Scoobas, no Aibos or I-Cybies.

Robotics in the New Millennium

Today, robotics touches all of us in some way or another. I've lightly touched upon the many types of robots in my series of columns, but there are so many more applications available that no school or course can cover them all. A recent NCIS episode on TV featured an autonomous Hummer that was programmed incorrectly and became a killer. Sounds a bit like the early 'B' movies that featured robots as malevolent creatures.

Robots are still a mystery to most people though they welcome their services with open arms. The I-Robot ads show people commenting "I love robots." We need to know more about the new machines available on the market.

Beginning Robotics — LEGO Mindstorms and OWI Kits

The first of the newer kits were made by LEGO, the company that made the plastic stick-together blocks for kids. The LEGO Discovery and Mindstorms kits proved to be an overwhelming success to tens of thousands of kids and adults alike. Some have complained, "those plastic blocks are just kid's toys." "You don't have any versatility in construction." "They're too cute to have any value teaching anybody above elementary grades." Well, these people have never been to one of the thousands of annual competitions across the nation where complex creations made of special shaped LEGO blocks, sophisticated sensors, gearmotors and powerful RTX microcontrollers beat custom designed machines (Figure 7). They may be made of brightly-colored plastic, but they are powerful machines and an excellent way to introduce robotics to any student.

Another series of robots that has been around since the 80s is the small creations by OWI. These kits cost from under \$10 to \$70+ and offer the handy tinkerer — young or old — a way to make a functional robot for low cost.

Figure 9. Robotix Kit.

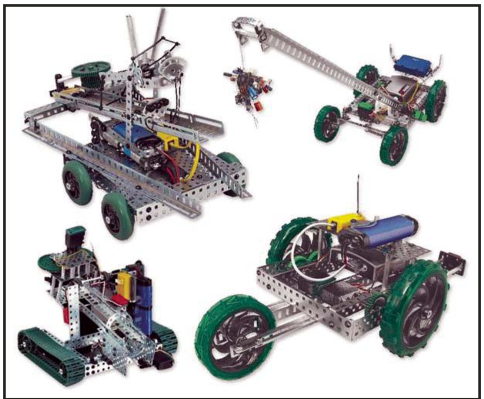


Figure 10. Vex Robotics Starter Kit.

Robotix and Vex Robotics Systems

One of the newer robotics educational systems is Robotix. This series of construction kits was originally designed for kids and included all sorts of motorized 'space' things that moved and could be programmed. They later began to develop a great curriculum to go with the kits and it was rapidly accepted in many schools across the country.

The Robotix Teacher's Guide — *The Science of Robotics* — is a thorough introduction to assist teachers of all school levels. I was fortunate to be able to watch a class being taught at my local Orcas Island School using Robotix (Figure 8).

The Boeing Museum of Flight in Seattle travels to many rural schools such as Orcas to assist teachers in this new science of robotics with a large supply of these kits. I was amazed at how all levels of students easily assembled and programmed their individual robots.

Another new kit series recently introduced is the Vex Robotics series, first introduced by RadioShack. Vex reminds me of a combination of the LEGO kits mixed with the old Erector Set beams, but with a unique creativity potential for the student. The first $300 kit sold fairly well but RadioShack soon stopped handling it.

Dean Kamen and FIRST

FIRST is a non-profit organization founded by Dean Kamen who brought us the Segway Transporter. An acronym For Inspiration and Recognition of Science and Technology, Dean feels it is his greatest contribution to society, though there are many who are alive or ambulatory due to his many innovative medical devices and stair-climbing wheel chairs and feel those are his greatest inventions.

FIRST seeks to inspire an appreciation of science and technology in young people through a series of robotics events such as the FIRST Robotics Competition (begun in 1992); the FIRST Vex Challenge (begun in 2005) for high school ages; the FIRST LEGO League for younger kids; and the Junior FIRST LEGO League for six to nine year olds.

Kamen's organizations utilize LEGO, Vex, and a very high quality (and a bit pricey) robot kit for the high schoolers. In last year's contests, over 1,180 high school teams competed. Over 15 million Google hits on "First Robotics" proves the success of Dean's organization. He deservedly should be proud of this accomplishment.

Robotics education has proven to be the 'carrot' to entice students into fields of science and engineering. Rhode Island has offered a variation of the FIRST competition in all 67 of its public high schools to give students a head start, and other states are seriously considering similar programs. One to two hundred dollar Robosapiens and Roboraptors are flying off the shelves of Linens and Things, of all places, and Roombas in Target Stores are prime gifts.

Consumers want robots and people of all ages want to know more about non-industrial robots. Go to the Internet and plow through millions of links to find the books and courses applicable to your needs. Go to Dan Kara's site at **www.robonexus.com** to see what his company, Robotics Trends, has compiled. There is more available for the potential student than you'd ever imagine or I could ever cover in a single article. **SV**

ABOUT THE AUTHOR

Retired from Rockwell and NASA projects, Tom Carroll is a space robotics engineer. He's authored numerous articles on combat robots, lives on an island in the Pacific Northwest with his wife, Sue, and enjoys robotics, kayaking, and hiking. He can be reached at: TWCarroll@aol.com

Advertiser Index