ROBOTS WHO LISTEN • CAMPUSBOT • FACE WALKER

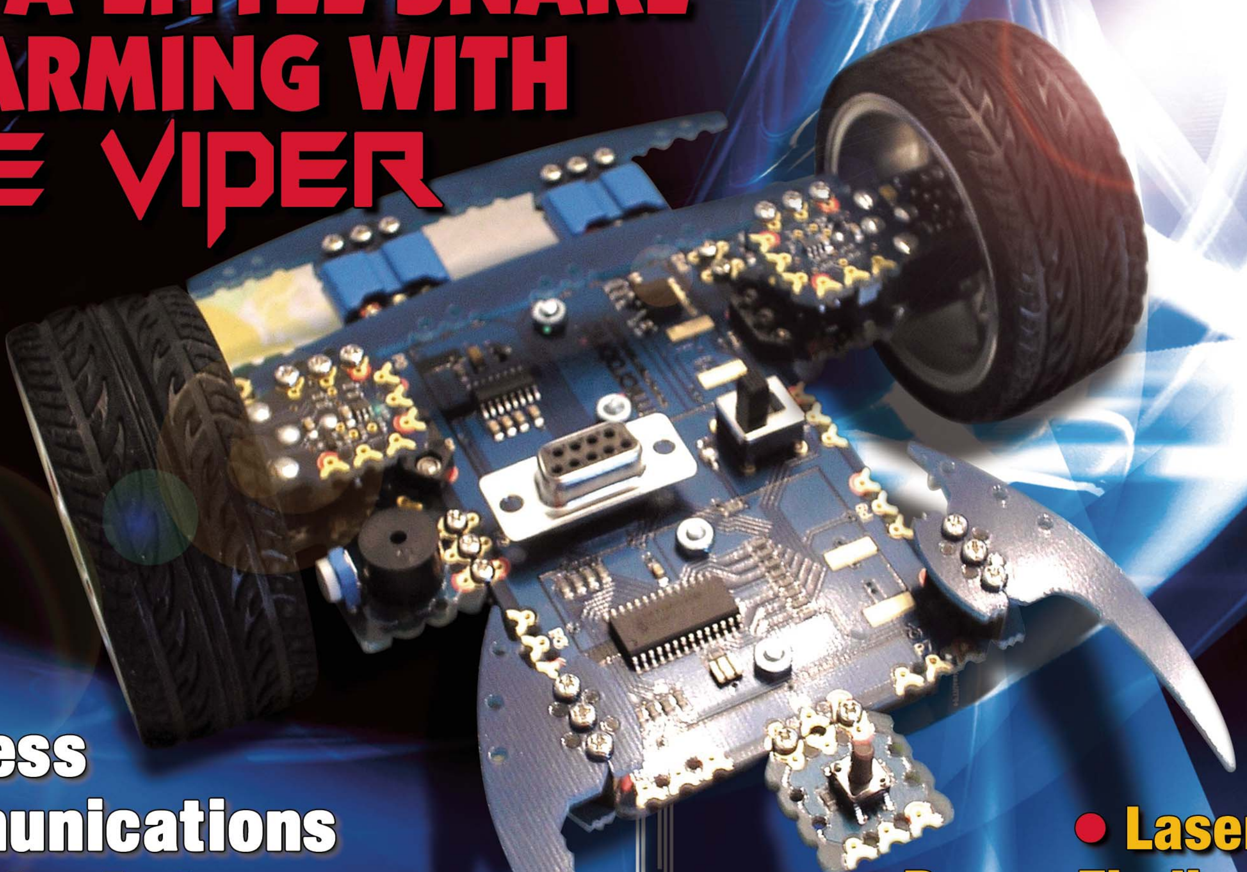
SERVO MAGAZINE

FOR THE ROBOT EXPERIMENTER

www.servomagazine.com

SEPTEMBER 2006

DO A LITTLE SNAKE CHARMING WITH THE VIPER



Wireless Communications For Tele-Autonomous Robots

• Laser Range Finding

• Designing A Seven Segment Counter

• Roboreptile Is Here

SERVO MAGAZINE

Columns

Departments

ENTER WITH CAUTION!

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

09.2006
VOL. 4 NO. 9

Features & Projects

Roboreptile is here!
See Page 34

Jammin' Bot Band
See Page 37

Mind / Iron

by Bryan Bergeron

My first exposure to practical robotics was repairing an analog autopilot in the belly of old cargo ship. An oil capacitor had ruptured, resulting in an under-damped feedback circuit controlling the hydraulics of two mattress-sized rudders. At the time, I didn't appreciate the challenges of seamlessly integrating electronics with mechanics. As an electronics technician, my focus was on debugging the defective feedback loop. That the circuit happened to control the movement of a few tons of steel was only momentarily interesting.

Decades later, with dozens of robotic projects under my belt — some successful, some blatant failures, but all learning experiences — I appreciate the engineering finesse behind any robotic or mechatronic device. I also find it fascinating how rapidly robotic principles have transformed my home, work, and leisure life. My inkjet printer, tape backup unit, and DVD player contain MCU-controlled servos and motors. The wireless, Hall-effect computer on my bike displays real-time and average speed, time, and distance traveled. My shop is filled with power tools that have processor-controlled speed, current sensing, and temperature cutoff. And, of course, the dozen or so MCUs in my car monitor hundreds of parameters, from the status of the antilock braking system to the rate of fuel injection. If the embedded accelerometer detects an impulse of sufficient amplitude, the air bag will hopefully deploy in time to save my life.

In the hospital where I spend some of my time, there are robot surgical assistants that occasionally make the national news. And there are the less well-known animatronic patients that look and respond just like real patients. Their chests rise and fall with each breath, their eyes respond to light, and the pulsations of fluid-filled tubes can be felt just beneath foam rubber skin. More significantly for the physicians and medical students honing their craft, the simulated cardiovascular systems respond appropriately to anesthetics and other medications.

Less well-known, but critical to providing quality patient care are the hundreds of robotic devices concerned with routine tasks that range from pumping fluids into patients, to focusing the beams of various forms of radiation for imaging and therapy. In the research buildings adjacent to the hospital, surgeons perfect their techniques using tele-operated orthoscopic instruments. But stop and ask any of the hospital staff in the halls if they've seen a robot lately, and you're likely to get a blank stare.

Robotics — like AI and other initially over-hyped technologies — has quietly become absorbed in everyday products and devices. The pervasiveness of robotics is invisible to the casual observer. This is, in part, because a digital camera with auto-everything doesn't fulfill our expectations of what we've come to expect from exposure to *Lost in Space* or *I, Robot*. Most of us have been conditioned to equate a robot with a humanoid created in our image. But that mindset is both empowering and limiting. It's empowering in that, given a concrete goal, someone will eventually succeed in creating a commercially or at least militarily viable humanoid robot. It's limiting because innovators and entrepreneurs may shy away from the more practical but less glamorous applications of robotics.

Many robotics enthusiasts dream of working in a federally funded laboratory or commercial R&D firm with the latest equipment and devices. However, as someone who straddles both worlds, I can say that enthusiasts often have the better deal. Although there is some satisfaction in working with a team on a government-funded multi-year project, enthusiasts have the freedom to pick and choose the technology, tools, and application areas that suit their current interests. This choice is facilitated by affordable and powerful computing power, publications such as this one, and communities of mentors and students supported by the Internet.

In addition to the articles, I subscribe to *SERVO* for the pictures of products

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, Pete Miles, David Geer, Jack Buffington, R. Steven Rainwater, Gordon McComb, Michael Simpson, Chris Cooper, Kevin Berry, Dave Calkins, Gerard Fonte, Bryan Bergeron, Paul Verhage, Evan Woolley, Bryce Woolley, Steven Kirk Nelson, Jeffrey Scholz, Charles Guan, Tim Wolter

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
sales@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

from advertisers. I especially scrutinize the robot-specific hardware and software offered by the niche vendors. Why? Because the products are the work of enthusiasts who have pushed their vision of robotics far enough to make a commercial product. In this respect, the advertisements represent a Darwinian selection of concepts, visions, and approaches to putting the theory of robotics into practice.

Of course, there's a place for the research journals on your bookshelf if you're going to keep up with the latest developments in AI algorithms or the physics behind sensor technology. However, robotics is a hands-on activity. Without practical implementation of theoretical concepts, the technology may never leave the confines of a lab. That's where the articles in *SERVO* come into play. Readers that take the initiative to actively experiment with the devices and algorithms discussed are rewarded with an intuitive grasp for robotics that can't be learned from passive reading.

In retrospect, one reason I wasn't impressed with the autopilot in the hot, oily belly of that cargo ship was because the black metal trunk housing the circuitry was far from awe-inspiring. Equally important was that is was just another subservient machine.

This will change with the next generation of robots that will work not only for, but with people. Imagine an affordable robotic wheelchair that can work with an elderly woman to help her decide if it's safe to cross a street can change her quality of life. Consider the value of a team of robotic firefighters that can work with a human firefighter to rescue people trapped by a fire while putting fewer firefighters at risk.

As you read through this issue of *SERVO*, pick one article and either apply it to your current project or use it as the basis of a new project. Hone your robotic intuition. It will serve you well, whether you're a student destined for one of those research labs or an enthusiast transforming your vision of the future into reality. **SV**

BIO-FEEDBACK

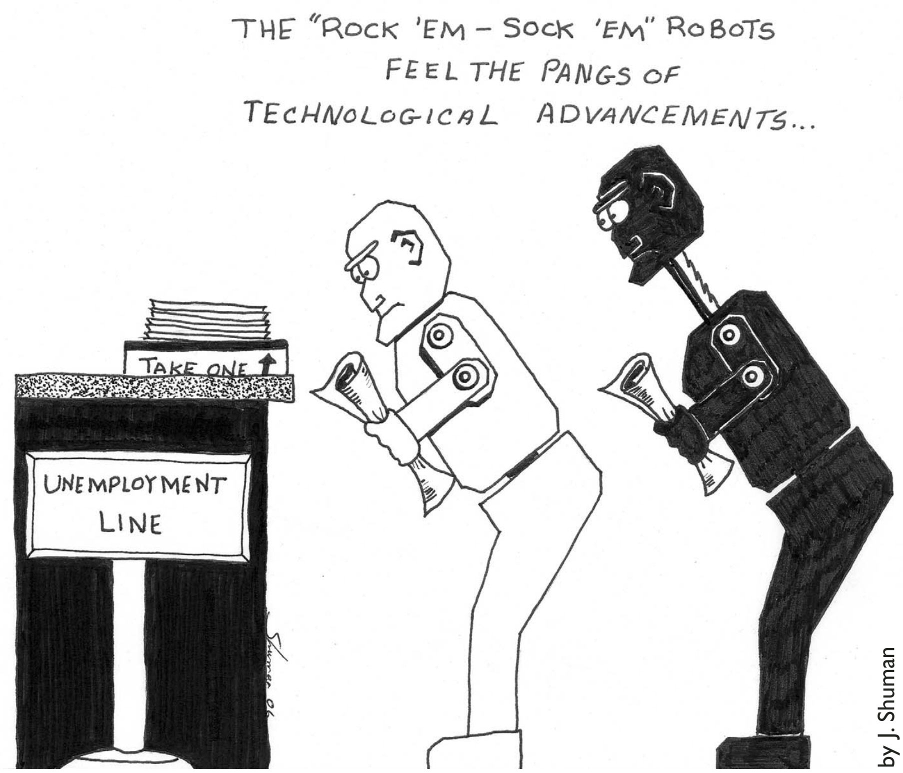
Dear SERVO:

Regarding last month's issue, with an article about H-bridges, and how to build your own ...

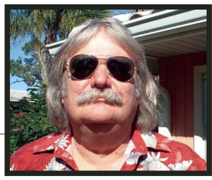
I have been designing mine for a long time in my few spare hours a week, and I have come to the conclusion that the best logic to control an H-bridge is exactly what is used in the L298 datasheet, and I just wanted to suggest that if you mention this datasheet, it has one flaw which is easily fixed.

The L298 has a separate enable pin and requires two PWM signals per motor, but if you have limited PWM pins on your processor like myself (using the BasicX 24), all that's needed is an inverter on one of the inputs. If you use the two pins that are supposed to have PWM on them as a single pin for direction, the inverter will automatically hold the other line at the opposite logic level. So for example, forward is input high on pin 1, pin 2 is held automatically low, and PWM the enable line.

Gary Tolley



by J. Shuman



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

Automated Transfer Vehicle Passes Tests

Artist's impression of the Automated Transfer Vehicle docked with the International Space Station. Photo courtesy of ESA, D. Ducros.

If everything goes right, starting in 2007, the European Space Agency will initiate more or less yearly flights of its Automated Transfer Vehicle (ATV) to haul 7.5 metric ton payloads from a launch site in French Guyana to the International Space Station (ISS). After each launch, the ATV, referred to as "Jules Verne," will remain there as a pressurized and integral part of the ISS for up to six months. During its visit, astronauts will be able to access its contents while dressed in normal clothing, making it something like a huge pantry that will hold up to 840 kg of drinking water, 860 kg of propellant, and 100 kg of air. When its contents have been used up, it will become a celestial garbage can that can haul 6.5 metric tons of waste back into the Earth's atmosphere, where the ATV and contents will burn up.

The latest news about the ATV is that it has successfully passed several days of acoustic testing, conducted at the European Space Agency's test facilities in Joordwijk, Netherlands. This was necessary to ensure that it can withstand the stress of launch, which will expose it to an overall sound pressure of 144 dB with frequencies mainly in the range of 25 Hz to 5 kHz. Sensors attached to the ATV confirmed that it suffered no damage, so it appears to be on schedule. Details and progress reports are available at **www.esa.int**.

Three-Finger Gripper Introduced

The SCHUNK gripper offers a range of grasping configurations including (1) parallel grip, (2) central grip, (3) cylinder grip, and (4) large parallel grip. Photo courtesy of SCHUNK GmbH & Co. KG.

A slightly eerie, but apparently versatile manipulation device is the new, electrically operated Schunk Dextrous Hand (SDH) from SCHUNK, Inc. (**www.schunk-usa.com**), a subsidiary of SCHUNK GmbH & Co. KG. Introduced at Automatica 2006 in Munich, the gripper is equipped with three identical double-jointed fingers, two of which can switch positions for a greater variety of gripping applications. The SDH is fitted with six tactile sensors on the grip surfaces that register and relay local contact force information back to a controller, which enables object recognition and optimal functional parameters (e.g., a delicate, but secure grip). The base of the hand contains a controller that can store control strategies and act as a decentralized program module.

The SDH, which operates from a 24 V power supply, is particularly suitable for industrial environments, being dust- and waterproof. The hand can generate torques of up to 4.8 Nm in the proximal joint module and 2.1 Nm in the distal module, which roughly corresponds to the strength of a human hand. Although at this point you may be imagining it clamped around someone's throat, the company emphasizes that it is highly safe when it comes to human interaction. The hand has no corners or sharp edges, and if it encounters an unexpected obstacle, it will detect the increased power consumption within a few milliseconds and respond accordingly.

Microsoft Enters the Picture

If you were hoping that Microsoft would mind its own business and stay out of robotics, well, the news isn't good. At the RoboBusiness Conference and Exposition 2006, the company previewed a Windows®-based product for developing robotic applications in commercial, academic, and hobbyist environments, across a broad range of hardware. Called Microsoft Robotics Studio, it includes a visual programming tool for program creation and debug, and it also provides simulation of robotic applications using 3D models. It will allow users to access the robot's sensors and actuators with a web browser, and the package will be third-party expandable via added libraries and services. Both remote (PC-based) and autonomous operations can be developed using several programming languages, including Microsoft Visual Studio and Visual Studio Express, Javascript, Iron Python, and others. The product is in the community technology preview stage as of this writing, and the preview version is available for download at **msdn.microsoft.com/robotics**.

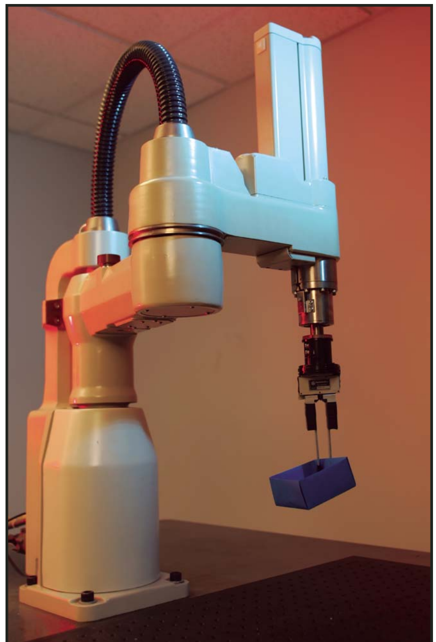
Robotic DVD/CD Publishing Introduced

The Aleratec Auto Publisher One™ provides unattended DVD/CD duplicating, publishing, and digital imaging. Photo courtesy of Alera Technologies, Inc.

In a move that looks like another headache for the major recording companies, Alera Technologies (**www.aleratec.com**) has introduced the DVD/CD Auto Publisher One — a robotic autoloading duplicator with a built-in inkjet printer. The fully enclosed machine offers a 75-disk capacity, a 16x recorder, and a 4800 dpi color photo quality disc printer. Production capacity is about 20 DVDs per hour, and it even comes with the company's Mastering, Recording, and Labeling Software Suite. All you have to do is run a USB 2.0 cable from your PC to the machine, and you'll be cranking out discs within minutes. The street price is estimated at about $3,000.

New Hall of Fame Inductees

Five robots have been inducted into Carnegie Mellon University's Robot Hall of Fame®, which was founded in 2003 as a tribute to both real-world and fictional robots that have advanced the concept of robotics. This year's inductees include Maria (the star of Fritz Lang's classic film, *Metropolis*), Gort (from the 1951 movie *The Day the Earth Stood Still*), David (the android from Steven Spielberg's *Artificial Intelligence: AI*), Sony's AIBO robot dog, and (back in the real world) the Selective Compliance Assembly Robot Arm (SCARA), which is a common, generic, and generally four-axis industrial arm that has been widely used for assembling consumer products. You can see them all at **www.robothalloffame.org**. **SV**



Among Carnegie Mellon's 2006 Hall of Fame inductees is SCARA, a widely used industrial arm. Photo courtesy of Carnegie Mellon.

This month, we have the pleasure of presenting the Viper from Microbric, a robot kit all the way from Australia. The Viper is a unique robotics kit with the distinction of being a "solderless construction set made for electronics enthusiasts," according to the manual. The Viper attains the paradoxical status of a solderless electronics kit by its innovative system of modules and "brics."

Each module is a clean electrical unit, with all the essentials you need for a variety of robotic designs — everything from LEDs and motors to bump sensors and infrared receivers. The kit also comes with blank modules — perfect for a snake charmer of the mechanical predilection.

This unique solderless electronics design, however, is a double-edged sword. While it may make the assembly arguably simpler and more approachable to a beginner at electronics — not to mention the clean and sleek look the lack of solder joints provides — it furnishes these advantages with the sacrifice of the stability and reliability of cleanly soldered joints. So, if you are all about old school, you may have to set your beloved iron to the side for this project. But overall, the assembly of this solderless kit is something any enthusiastic roboteer should be able to manage and appreciate.

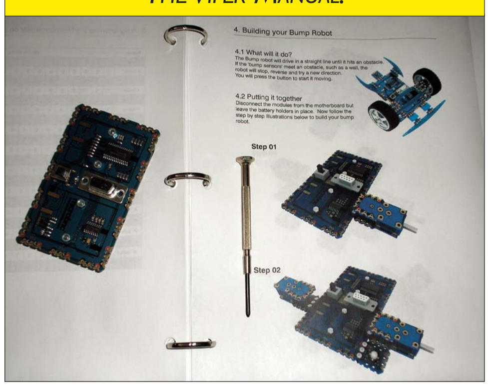
When we took our first look at the "brics" that give the Viper its unique modular nature, two questions ran through our heads: Can you actually build this thing with only two hands, and will it actually stay together? Fortunately, the answer to both of these questions is yes. The brics effectively join the modules with little plastic pins, and then a clever usage of nuts and screws provides a solid electrical connection. The kit itself comes with the only tool you need to assemble the Viper — a Phillips head screwdriver.

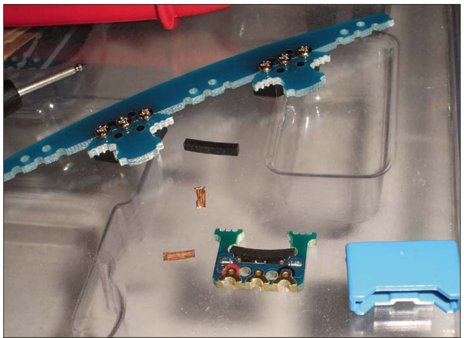
Snake Charming

The Viper is controlled by a Basic Atom, a microprocessor programmed in, as you could probably guess, Basic. The great thing about the Viper manual is that it gives a comprehensive walkthrough of the programming, detailing the commands associated with each module and then placing those commands in the context of a complete program.

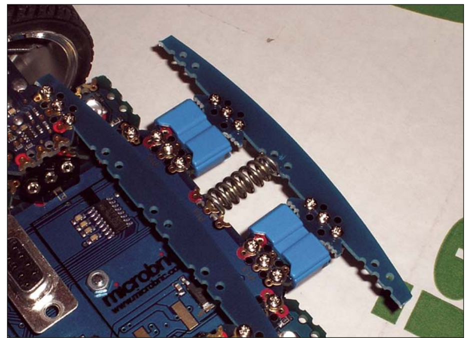
THE VIPER KIT FROM MICROBRIC.

THE VIPER MANUAL.





LET'S BUILD A BUMP ROBOT!



VIPER BUMP MOD.

The comprehensive manual also details several beginning builds — projects to get the tinkerer acquainted with the kit and the programming. These exercises — like hooking up a buzzer or programming the robot to turn its LEDs on and off — are certainly a good way for a novice to get their feet wet; but for more experienced builders, glancing at the accompanying programs is all the introduction necessary.

The manual, however, introduces the programming in this section in a way that would be helpful to everyone, in our opinion. Instead of just throwing fragments of code at you, the manual puts each small program into a clear flowchart — perfect for beginners, or even for programming pros used to programming in C that need a refresher in Basic.

In our experience, the Viper's flavor of Basic and the instructions in the manual on its usage are some of the most intuitive programming tools that we have come across for any kit. The sample programs are also meticulously commented and serve as perfect templates for your own custom programs. So, even if you thought Basic only referred to a pH over 7, or a Jackson/Travolta movie, then you should still be able to write something to get the Viper to slither around.

But, just in case you have some trouble, the BasicMicro IDE also contains a very helpful and easy-to-use debugging mode. The progress of the program can be visually mapped using a traveling green bar, which glides by working parts of the program and sticks on the problem areas. It's kind of like having your own personal Springer Spaniel, but instead of hunting, you're programming; instead of the dog, you have a green bar; and instead of ducks, you have syntax errors. However loose the analogy, the BasicMicro IDE included with the Viper kit comes with a cornucopia of helpful tools for programmers of all skill levels.

Untangling Jormungand

The Viper manual comes with instructions on how to build two basic Viper incarnations — the bump robot and the remote controlled robot for use with the remote control that comes in the kit. A video that comes with the CD also showcases a sumo version of the Viper, but that requires extra pieces not included in the starter kit.

We started with the more autonomously inclined bump robot. The building instructions in the Viper manual are unique in the sense that they are completely done with beautifully rendered 3D drawings. The images are a little on the dark side, but other than that they are perfectly detailed to lead you through the initial construction of your Viper — pictures are worth a thousand words, after all; or at least they will save you from uttering a few four letter ones in frustration.

The actual construction of the Viper is also well-suited to beginners. The innovative brics really make attaching the modules easy. And though the nuts and screws that hold the kit together are nearly of the maddeningly small variety, the brics are designed to hold the nuts while the module is fastened. It was a nice feeling to be able to build a small robot without having to wish for nimbler fingers. And there is one final thing about the Viper kit that gets our seal of approval — the tires. They smell like real tires! That's quite a rarity that we think speaks to the overall quality of the kit.

The bump robot detailed in the manual also has a corresponding program ready for downloading on the Viper disk. When we first loaded the bump robot program onto the Viper, we expected to see simple obstacle avoidance behavior by virtue of its front mounted bump sensors. What happened was more like the proverbial dog chasing its tail.

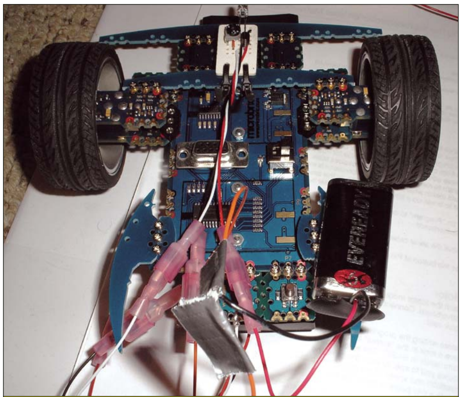
The problem was easy to identify. We were suspicious of them from the very start, and this erratic behavior only confirmed our suspicions. The bump sensors — they were fishy. The bump sensors actually came in pieces, and they had to be assembled along with the rest of the bump robot.

The bump sensors were constituted by two PCB bits, a mysterious piece of rubber, and a plastic casing. At first glance, you might be compelled to ask yourself "Where's the electrical connection?" The most unexpected answer to this burning inquiry is "in the rubber."

Supposedly, the rubber in the bump sensors was conductive, and when the bump sensor was pressed, the rubber would bridge the gap between two pads of the main PCB bit and give a reading. After some thorough investigation with a multimeter, we came to the conclusion that the rubber in our kit was not actually conductive. There was, however, an easy fix.

All of the Viper modules had to be punched out of a large PCB, and stuck onto the PCB, right where the bump sensor bits had to be punched out, were two small strips of metal. Maybe

VIPER MODULES.



VIPER POWER MODS.

these were the secret ingredients needed all along to concoct "conductive rubber," but after a bit of super glue and no hard feelings we did indeed have working bump sensors.

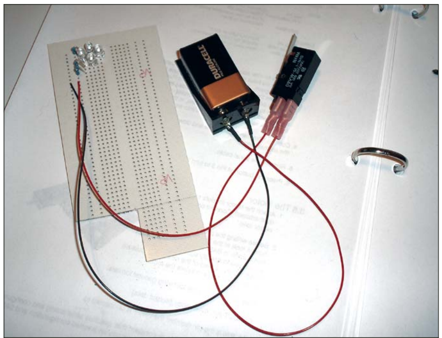
Now that we had the first incarnation of the Viper working, we needed a way to test it. In our experience, small robots like the Viper always love a good maze, so we constructed a simple maze for the Viper to slither through.

Slither In, Slither Out

A simple maze would be an effective way to test the ability of the stock robot with a stock program, the stock robot with a custom program, and a custom robot with a custom program. The first step on our hierarchy of complexity was to put the bump robot with the stock program through the maze.

The bump robot's performance certainly left room for improvement. It always got stuck in turn one of our simple U-shaped course. The program worked perfectly, but after being depressed the first time, the bump sensor would not return to a neutral position. The result was that the hapless bump robot would spin helplessly in the corner. Our idea for a quick tweak to fix the bumper was to spring load the mechanism, but even a beefy spring couldn't get the bumper to snap out of its depression.

We thought the Viper might be happier if we got rid of the bumper and gave it another program, so we scrapped the stubborn sensors and substituted the superfluous cipher for

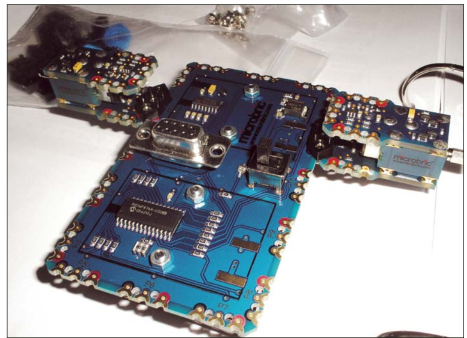

VIPER BEACON.

something more suitably savvy. We settled on a dead reckoning program, which did away with fiddly sensors altogether. We still used the stock robot — we just removed the uncooperative bumper and a push button that was surplus to requirements. The real change was in the program.

Of course, a dead reckoning program isn't that difficult. There's no sorting through sensory data or anything like that — just a series of directions. The Viper sample programs make a dead reckoning program even easier because they already come with subroutines like "forward," "spinright," and "spinleft." All we had to do was give the Viper the directions through the maze — and a simple U-shape meant simple directions.

It all sounds so easy, so a dead reckoning program should get the Viper through the maze perfectly, right? Not really. The problem with dead reckoning programs is that they are notoriously unreliable. The essential reason as to why this is the case is that a simple dead reckoning program is like a poorly conceived science experiment — there are too many confounding variables.

If the robot was placed in a slightly different place or at a slightly different angle, the destination could end up completely different. In the case of the Viper, even the slight movement that flipping the on/off switch creates could be enough to stymie the bot's attempt at solving the maze. Sometimes it still would complete the maze, and other times it would seal its own doom by running up on a wall right before the finish line. Sometimes it didn't even make it past the first turn. Overall, the dead reckoning snake was certainly more


IT'S A VIPER IN PROCESS ...

successful than its predecessor, but only slightly. There had to be a better way.

Snake Eyes

And there was. A fusion of sensor input and preprogrammed directions seemed like the best, albeit most complicated way, to reliably solve the maze. The problem was that the main sensor in the stock Viper kit was the fiddly bump sensor. There were other sensors available for the Viper like light sensors that could be used for line following, but we're all about hacking, not paying shipping and handling. So, we would make our own sensors — some snake eyes.

Our initial inspiration for our snake eyes came from a past FIRST game. The FIRST 2004 game used infrared beacons at the beginning of the match to lead intrepid autonomous bots to pedestals with balls on them so they could score extra points. If an infrared beacon could help a big FIRST robot navigate a game field, it should also certainly be able to help the Viper negotiate our maze. This hack would furnish the Viper with two major additional parts — an infrared receiver module (it came with one for the remote, but we're all about custom components), and an infrared beacon.

The Unsteady Viper's Navigation Mod

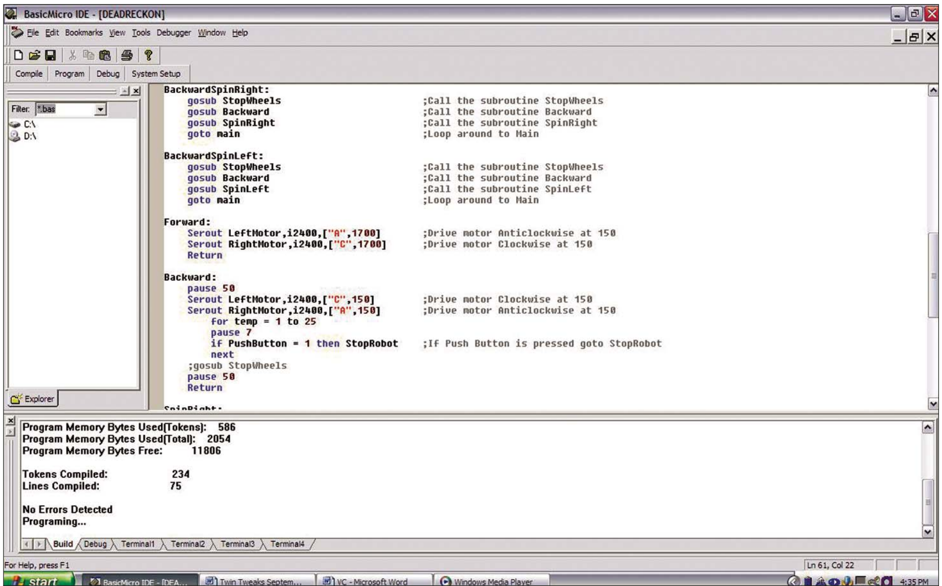
Our custom module was a simple circuit that was made up of three elements: a basic transistor, a phototransistor, and a 100K ohm potentiometer. As is turns out, our simple custom module could actually do double-duty as two different sensors. A simple adjustment of the potentiometer with a screw driver could turn the sensor from an infrared sensor to a dark sensor.

The infrared transmitter was also an elementary circuit. It was basically four parts — four infrared LEDs for the beacon itself, a nine volt battery, a few resistors to get the right voltage, and a switch that we wired in so we could turn the beacon on and off at our leisure. We grabbed a breadboard and some scrap PCB and we were good to go.

After mocking up both circuits on a breadboard and testing them to see that they worked, we wired them up for real on PCB bits. We cut down the receiver "module" so that it would approximate the size of the other Viper modules. It differed from the Viper modules in that, instead of using the brics for attachment, we had the wires that extended from the custom module end in connectors.

Their other corresponding halves were connected to wires that we soldered to one of the blank Viper modules, which could finally be fastened to the actual Viper via one of the brics. So, we did use the brics, but in an indirect matter. The PCB bit with the actual infrared receiver was simply tie-wrapped to the Viper itself. Our custom module was a bit bulkier that the Viper's stock modules, but it was still just as easy to connect and disconnect. Mission accomplished — that is, if the module worked.

Programming the Viper to use an infrared receiver was not difficult given our maze course. The simplest program we could think of would just have the Viper turn right when it saw the infrared beacon. Ideally, we would have several beacons to place around the maze, but one mobile beacon should also be enough — it would kind of be like a rudimentary remote control, in a sense. After downloading the program, we eagerly tested the Viper only to find out that it was just as unsteady and directionally challenged as a sidewinder trying to cross a balance beam.

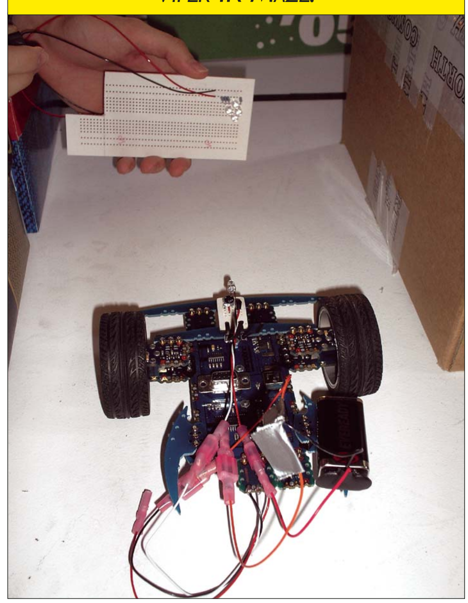


TIME TO START PROGRAMMING.

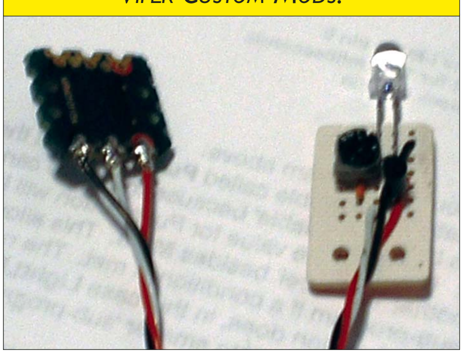
Why wouldn't our infrared receiver work? A quick multimeter diagnosis revealed that the Viper was always reading low from the receiver, no matter how much we messed with the potentiometer. That meant a problem with the receiver itself, and after studying our circuit again, we discovered our mistake. The sensors on the Viper all run on five volts, so when we hooked our custom module into the Viper, it also ran on five volts. We tested our module on nine volts and it worked fine, and that's because the breakdown voltage of the transistor was six volts. Too bad we weren't playing horseshoes — now we had to think of a way to trounce our transistor tribulations.

It may have been possible to find a transistor with an acceptable breakdown voltage, but we opted for something more accessible — a nine volt battery. An extra power source just for our custom module could give the transistor the voltage it needed, and with the potentiometer, we could make sure that only five volts were going back to the Viper itself. Unfortunately, this adjustment eliminated the capabil-

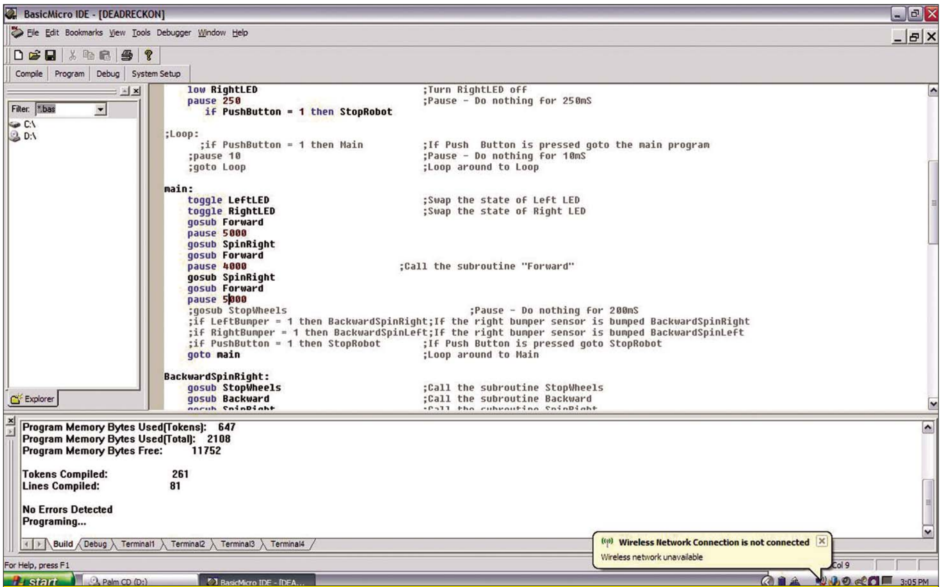
VIPER IR MAZE.



VIPER CUSTOM MODS.



MAZE TIME!



Yet, Some More Programming ...

ity of our sensor to be a dark sensor in addition to the infrared sensor, but that was okay because it really didn't make sense, anyway. Snakes are cold blooded, so why would our Viper go around hunting darkness? Things worked out for the best.

After our quick power addition, the infrared module did indeed work, and the Viper was able to complete the maze far more reliably than the bump robot or the dead reckoning bot. The moral of the story — for the best results when working with a robot, a balance of the mechanical side and programming side of the bot is needed. And what could be a better illustration of that than a robot getting stuck in a maze, and after a few modifications, a robot completing a maze?

V for Viper

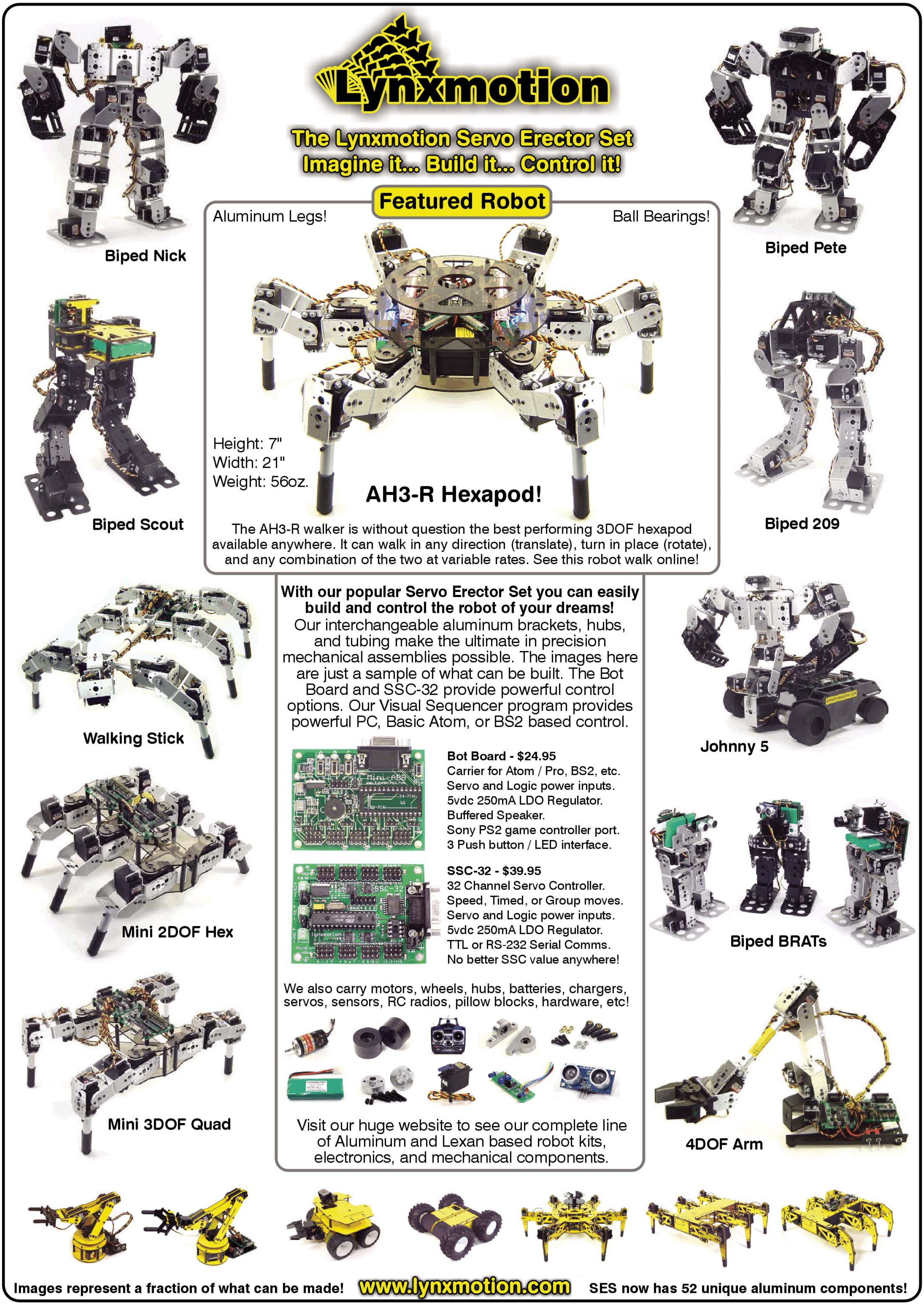
So, we were able to hack on an infrared sensor, but how do we think the Viper would take to hacking and expanding in general? Quite well, actually. A lot of robotics kits that intend for the builder to expand upon them come with special features to facilitate hacking. Past projects that we've worked on have had things like input/output ports unused by the stock robot. The Viper has that, but they aren't your ordinary ports. Other kits will have ports specifically for PWM inputs and the like, but the Viper just has a plethora of ports for modular attachments. This works great for the extra modules that you can order, but where does that leave the hacker? In a pretty good place, actually.

The Viper does come with extra blank modules, and even though the Viper is pumped up as a solderless kit, they are handy things to solder your own creations to. Of course, the way we made our custom module eliminated one side as a possible attachment point, but that was a sacrifice we were willing to make. We're sure electronic gurus out there could also cleverly build whatever custom sensor they want directly onto the blank module if their fingers were nimble enough, but tinkerers more at the not-quite-a-guru level could also easily use long wires and connectors like we did.

The Viper also has another valuable resource for hackers — a very well established Internet community. The Microbric website — where the Viper is prominently featured — includes an online forum where Viper users from all over can discuss the apparently very popular kit. Topics range from a basic discussion of the Viper kit to problems with programming to suggestions for a chassis. And if people aren't already talking about the problems that you've been having, go ahead and post a question. After just a little snooping on the site, I could see that questions were answered pretty efficiently, even if it was just a problem that one person was having.

Overall, the Viper is certainly a well-suited kit for electronics experimenters, no matter what their skill level. The Viper's electronic building blocks could be used for anything from an initial foray into robotics to rapid electronics prototyping to just messing around for fun. And remember, if things don't go so well and solder is like venom to your Viper, all you need is a solder sucker to restore the snake to its former solderless glory. **SV**

The Viper Forum.


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Featured Robot
Aluminum Legs!
Ball Bearings!
Height: 7"
Width: 21"
Weight: 56oz.
AH3-R Hexapod!
The AH3-R walker is without question the best performing 3DOF hexapod available anywhere. It can walk in any direction (translate), turn in place (rotate), and any combination of the two at variable rates. See this robot walk online!
Biped Nick
Biped Pete
Biped Scout
Biped 209
Walking Stick
Johnny 5
Mini 2DOF Hex
Biped BRATs
Mini 3DOF Quad
4DOF Arm
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.
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!
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Visit our huge website to see our complete line of Aluminum and Lexan based robot kits, electronics, and mechanical components.
Images represent a fraction of what can be made!
www.lynxmotion.com
SES now has 52 unique aluminum components!

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

roboto@servomagazine.com

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

ASK MR. ROBOTO

by **Pete Miles**

Q. I have some of those Sharp GP2Y0D340K object sensors and I can't figure out how to adjust the sensing range. The data sheet says it is adjustable from 10 to 60 cm, but the trip point always occurs around 15 inches. Can these sensors be adjusted? If so, how do you do it?

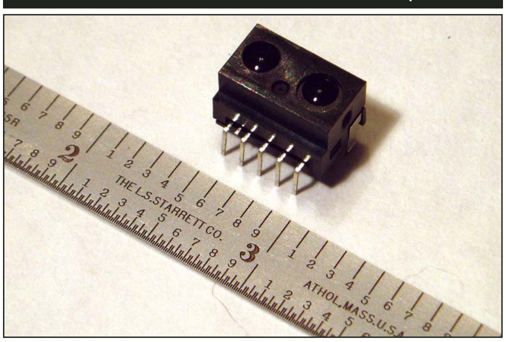
— Jason Reed

A. I have been wondering about this myself. I bought a set of them a couple years ago and never got around to actually using them. So, I decided to dive into this question and see if they can actually be adjusted.

I'll start with a little background information. These sensors are made by Sharp Electronics (**http://sharp-world.com**) and their datasheets can be downloaded from their website or where they were purchased. The housing for this sensor measures 0.59 inches wide by 0.38 inches high and 0.34 inches deep (15 mm x 9.6 mm x 8.7 mm). They require a five-volt power source and two additional components, a resistor and a capacitor. The output is zero volts when it detects an object, and is 4.7 volts when there is no object in its detection range. The normal detection range is 15.75 inches (40 cm). The data sheet does indicate that the range is adjustable from 3.9 inches to 23.6 inches (10 cm to 60 cm). But, it does not provide any information on how this is accomplished. One of the attractive points about this sensor is that it has an extremely fast response time — 6.4 ms — when compared to the other Sharp GP2xxxx class sensors which are at 38 ms. This faster response time allows for a more reliable detection of faster moving objects. Or it will enable your robot to move faster while having a higher confidence in detecting obstacles.

Figure 1 shows a photo of this sensor with a scale to show its relative small size. Figure 2 shows a simple schematic for testing this sensor. R1 and C1 are the only two components required to be used with the sensor. The transistor is only acting as an inverter so that when the sensor detects an object, the LED will turn on. When no object is detected, the LED would be off.

Figure 1. Photo of the Sharp GP2Y0D340K sensor with an inch scale for size comparison.



A point to note here is that the current draw from the sensor is not clearly defined in the data sheets. The data sheets talks about the average current draw with a 1 ohm resistor for R1. This can be misleading because it is the time average, not a peak current draw. The IR LED sends out a burst of 16 pulses at a 6.67 kHz frequency. For 18 µs during each time the IR LED is on, the current draw is approximately 300 mA when R1 is a 1 ohm resistor. When R1 is 2.2 ohms, the peak current draw drops to approximately 145 mA. This is important to know so that you can make sure that you use a power supply that is capable of supplying short pulses of current based on the sum of all of the sensors worst-case current draw. Otherwise, voltage drops due to high current draw from the sensors could have adverse effects on the rest of your electronics.

Modifying the sensor so that its detection range can be adjusted turns out to be a relatively simple process. Figure 3 shows a photo of the top of the sensor. You should notice that there is a small plastic tab inside a narrow slot. This tab is what needs to be moved (to the left or to the right) to change the sensor's detection range. But the immediate problem that you will run into is that this tab does not move. This is because the lens mount has been glued in place. Figure 4 shows a photo of the right side of the sensor. There is a small oval shaped hole in the side of the housing. When looking with a microscope, you will notice that a small drop of clear acrylic-like glue was placed in this hole. This glue is used to lock the lens in position. In order to change the detection range of this sensor, the glue spot needs to be removed.

The first step is to remove the front plastic lens mount from the sensor module. The adjustable lens mount is

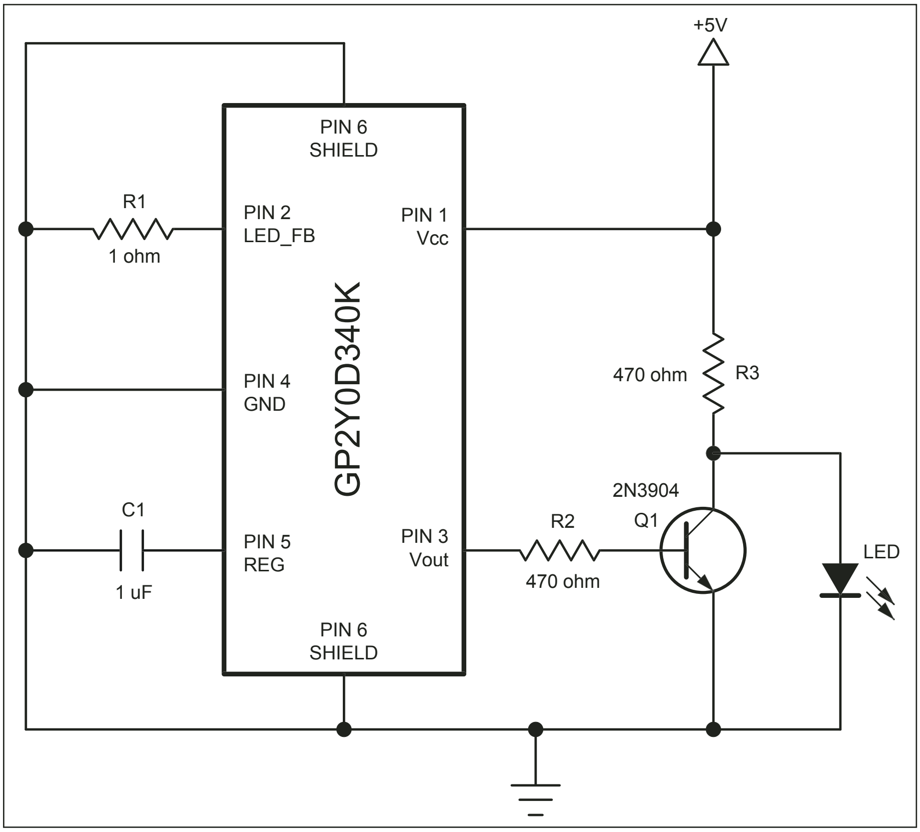


Figure 2. Schematic for testing the Sharp GP2Y0D340K sensor.

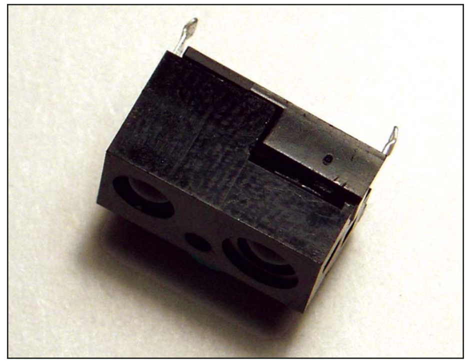


Figure 3. Closeup view of the Sharp Sensor showing the range adjusting sliding tab.

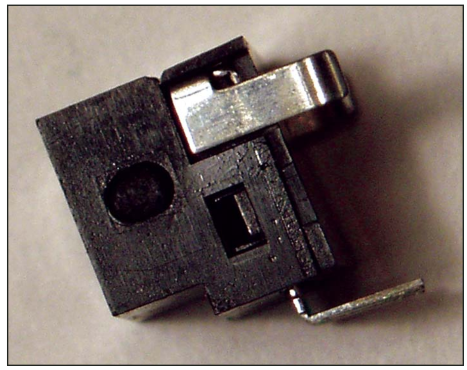


Figure 4. Side view of the GP2Y0D340K showing the hole where the lens mount is glued in position.

used to focus the reflected IR light onto the position detection sensor. The fixed lens covers the IR LED. Use a small jeweler's screwdriver and pry the tabs on the sides of the forward housing away from the hooks on the back side of the housing (see Figure 5). Figure 6 shows the forward lens housing and back housing removed from the sensor module. Figure 7 shows an interesting closeup view of the sensor module.

The next step is to use the jeweler's screwdriver to pry up the adjustable lens mount out of the forward housing. Figure 8 shows a photo of how this is done. You will want to use the screwdriver to work on both sides of the lens mount and slowly work (wiggle) the mount out. This will cause the glue on the side of the housing to break loose. The glue is a hard material and shatters/cracks when it finally breaks loose. When this happens, it will become easier to remove the lens mount. Figure 9 shows the lens mount removed from the forward lens housing.

The next step is to use a blade (such as an Xacto knife) to scrape off all of the glue remnants from the lens mount and housing. If any glue residue is left in place, it will be difficult to adjust the lens position. Then finally, the sensor is reassembled. Place the lens mount back in the housing with the small tab at the base of the mount inside the original slot that is above the lens mount hole (use Figure 8 as a reference). The lens mount should easily rotate left or right and the tab limits the full range of motion. Next, place the rear housing back on the rear of the internal sensor module. Then with the forward lens

Figure 5. Removing the front housing cover from the sensor.

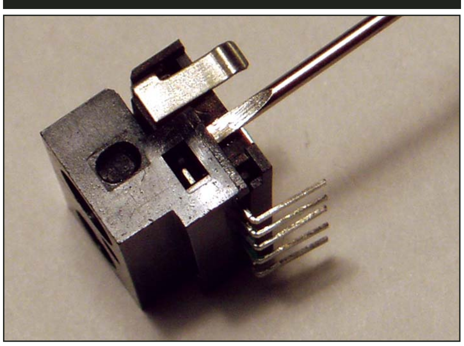


Figure 6. Optical sensor module removed from the forward lens housing and rear plastic housing mounts.

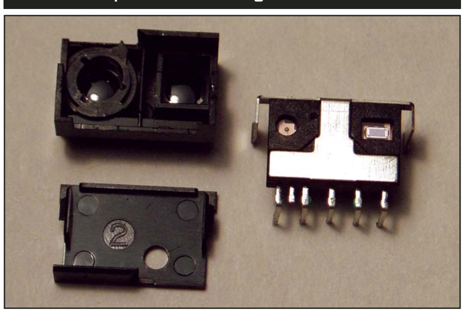
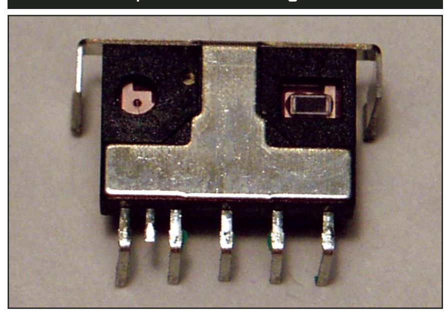


Figure 7. Closeup view of the front of the sensor module showing the IR LED and the position sensing detector.



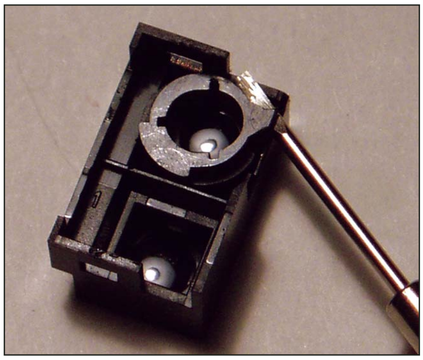


Figure 8. Using a small jeweler's screwdriver to slowly work the lens mount out of the forward lens housing.

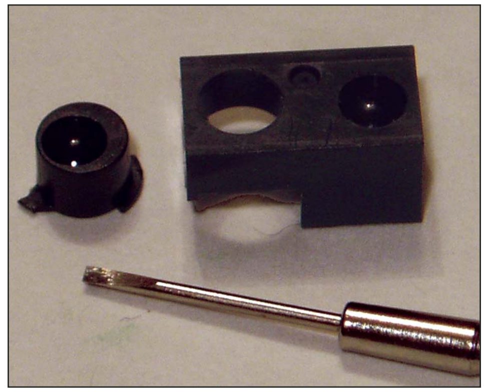


Figure 9. Adjustable lens mount removed from its housing.

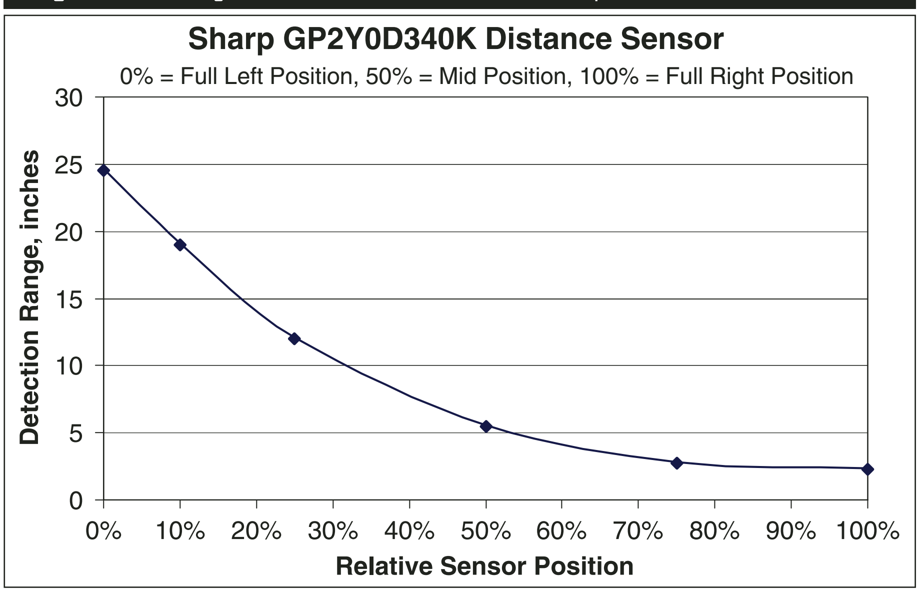
housing facing down, snap the sensor module and rear housing back into the forward lens housing. If the forward lens housing is not facing downward, then there is a chance that the lens mount might fall out of position and get jammed up against the sensor module.

At this point, you should be able to freely adjust the sensor's detection range by rotating the lens mount clockwise and counter-clockwise, and have the full detection range of the sensor.

With the sensors that I have, the detection range is not proportional with the rotational position of the lens. Figure 10 shows a plot of the detection range as a function of the sensor's detection lens position. 0% means that the tab position is all the way to the left side of the sensor. 50% means that the tab is centered in the middle of the slide slot. And, 100% means that the tab is pushed all the way to the right (towards the fixed lens). Figure 10 shows that the position of the lens and the detection range is not linear. The greater the detection range, the more sensitive the position of the sensor becomes.

At this point, you should have all the information needed to modify your sensors to detect an object anywhere between the 10 cm to 60 cm range. If you need to lock the sensor in a particular position, all you have to do is add a dab of glue in the oval hole on the side of the sensor. Ideally, you would want to use a semi-permanent glue that can be broken easily if needed. One suggestion would be red finger nail polish. It holds small things together just fine, will break off if pried apart, and the red will be easy to see. **SV**

Figure 10. Sensing distance as a function of rotational position of the detector lens.



Rubberbands and BALING WIRE

by Jack Buffington

Laser Range Finding

Last month, this column took a look at how to use the Taos TSL3301 linear image sensor. Adding vision capability to your robot can be exciting. This month, we'll cover a way that your robot can gather useful information about its environment using laser range finding.

This may sound extremely high tech and difficult to do but, in reality, it isn't so hard. There are only a few choices out there when you want to figure out how far you are from objects in your environment.

The old standard is to use sonar. You will often see university robots that have a big ring of gold disks around their perimeter. These are sonar transducers. Sonar is fairly reliable but can be tricked by soft surfaces and tends to round out details such as corners.

If you want to measure short distances, you can use the Sharp GP2D12 infrared range finders. They are good for distances up to 30 inches. Various things can trick them too such as too much ambient light. Another solution is to do stereoscopic range finding. This can be somewhat processor-intensive.

While laser range finding has its faults as well, it seems to be a blend of the best things of these types of sensors. Laser range finders can detect long distances and are able to detect crisp details such as the corners of rooms or possibly the biggest obstacle to mobile robots: chair and table legs.

How it Works

Let's take a look at how laser range finding works. Take a look at Figure 1 for a graphical look at how things are arranged. A laser and an imaging sensor are aligned so that the laser and camera are aimed in the same direction. After a certain distance, the camera will be able to see the laser dot. The farther from the range finder an object is, the higher the laser dot will be on the image sensor. It is really no more difficult than that.

In Figure 1, the gray lines represent the distance detected by each of the 102 pixels of the TSL3301. You can see that the precision that the range finder can resolve distances at is much higher if the object is close to the sensor. The farther that your object gets from your sensor, the less accurately your sensor will be able to resolve its position. This doesn't leave you with much accuracy using a sensor that only has 102 pixels. Fortunately for you, later in this column you will be shown how to greatly increase the precision that you can resolve longer distances at.

Let's look at what else you can do to increase the accuracy of your range finder. Figure 2 shows what happens if the camera is tilted downward. You might think that this would increase the accuracy of the sensor, but it doesn't. The only difference that this makes is to allow you to detect distances that are nearer to the range finder.

Increasing Accuracy

There are three things that you can do to increase the accuracy of your range finder at longer distances. The first

Figure 1. A diagram of how a laser range finder works.

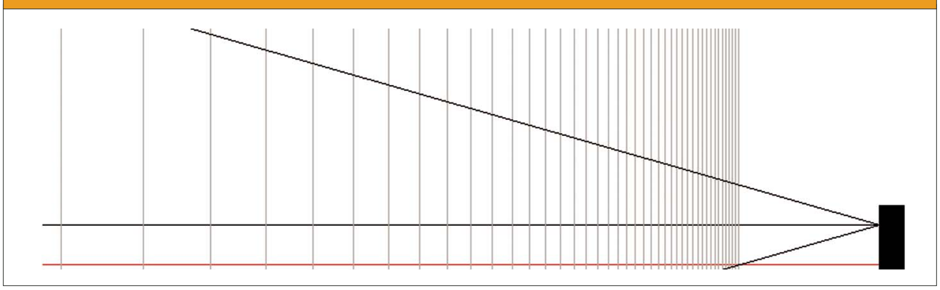
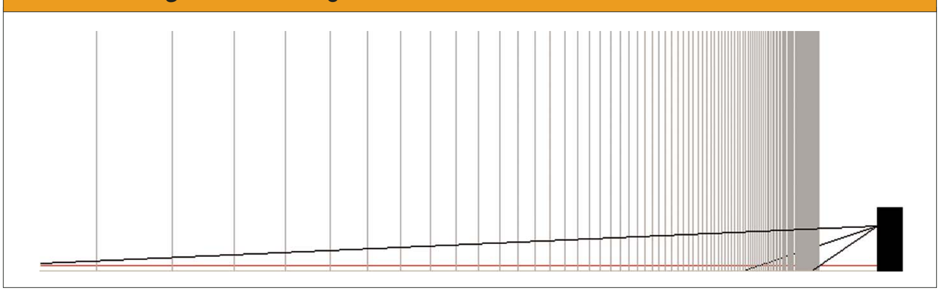


Figure 2. The range finder with the camera tilted downward.



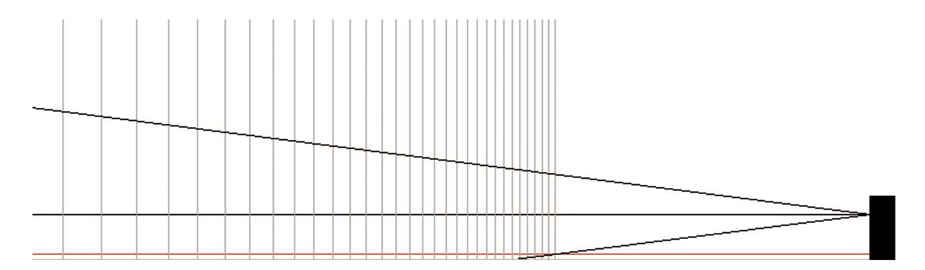


Figure 3. The range finder with a narrower field-of-view.

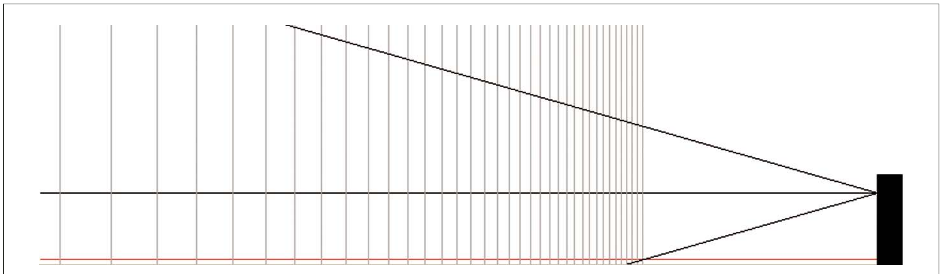
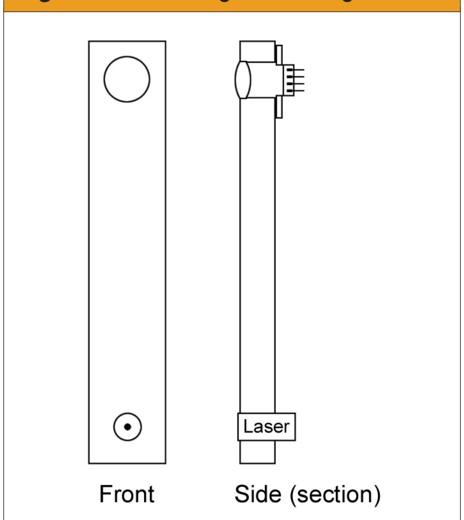


Figure 4. The range finder with more distance between the laser and image sensor.

is to use a sensor with more pixels. There is a linear relationship between how many pixels you have and how many distinct distances you can measure. Another way to increase your accuracy at distances is to decrease the field-of-view. Figure 3 shows the result of that. Of course, by doing that, you lose a lot of nearer distances that you could detect before. You can compensate by tilting the image sensor downward, though.

The last thing that you can do to increase your accuracy is to increase the distance between your image sensor and the laser. Figure 4 shows the result of doing that. Once again, tilting your image sensor can compensate for the loss of nearer measurements. There is a danger to increasing the distance between your image sensor and the laser that you should be aware of. When you make this increase, you also increase the likelihood that something closer to the range finder will obscure its view of the laser dot. This might seem like a non-issue, but it might be good to restrict the distance between the two to three or four inches.

Figure 5. A drawing of the range finder.



Rolling Your Own Range Finder

In a nutshell, that is all there is to making a laser range finder. Figures 1 through 4 were generated with a piece of software written for this column. You can download it from *SERVO*'s website (**www.servomagazine.com**) to help you figure out what the best setup for your application would be. Let's look at how you can actually make your own range finder. This design doesn't allow for tilting of the image sensor but works well, just the same. A piece of soft plastic was cut long enough to allow for the desired sensor-to-laser spacing. Measure this distance on the plastic and mark it. The lens hole was first drilled with a hole size that was too small for the lens to pass through, even by forcing it. Next, a second hole was drilled in the same location that was also smaller than the lens's diameter but big enough to allow you to force the lens into the plastic. Drill this hole only as deep as the thickness of the lens. This hole creates a 'force fit,' which will hold the lens in without the need for any glue.

You should experiment with the proper hole size for the force fit on another piece of plastic. Don't use anything that could damage the lens to force it in there. With the proper hole size, you should be able to insert it with your finger. Last month's column used a special lens setup that allowed you to focus, but this month, we are going to do away with that and figure out the proper distance between the lens and the sensor to get a good focus at most distances.

The laser used to make the test setup for this column was from a cheap laser pen that was found at a drug store. It was easy to mount because — like the lens — it had a circular profile. A hole was drilled that was just slightly too small to fit the end of the pen into. This allowed for a force fit of the laser, as well. Now we will figure out the proper focus distance.

This can be done by holding some tracing paper or other thin paper behind the lens in your range finder. Find the ideal distance and then search around for something that is the same thickness. Make sure to allow for the bit of clear plastic that covers the actual sensing silicon in the sensor chip.

In the test setup, another piece of plastic that was lying around was used. A slit was cut into this piece of plastic to allow light from the lens to hit the sensor but which blocks other ambient light. Before this was glued to the main body of the range finder, the sensor and small piece of plastic were placed in front of the lens and the focus was verified using the image viewing program that can be found on *SERVO*'s website.

Once everything is good to go, glue the plastic with the slit in it to the body of the range finder. You will now glue the image sensor on top of the plastic with the slit in it. This requires a bit of accuracy, so be careful. You want to make sure that the line of pixels points directly towards the laser. If your sensor is rotated, you probably won't be able to measure all distances. You will also want to make sure that the center pixel is lined up with the center of the lens.

Once you have the image sensor glued into place, you are almost there. Run the image-receiving program to view the results of your handiwork. If you are lucky, the laser will be visible at all distances. It is likely, though, that you won't see the laser at longer distances. You will need to rotate the laser from side to side so that it can be seen at farther distances. It is likely that this won't be very much of a rotation. Once you have it perfect, put some glue onto the laser to hold it in place and let it dry before proceeding.

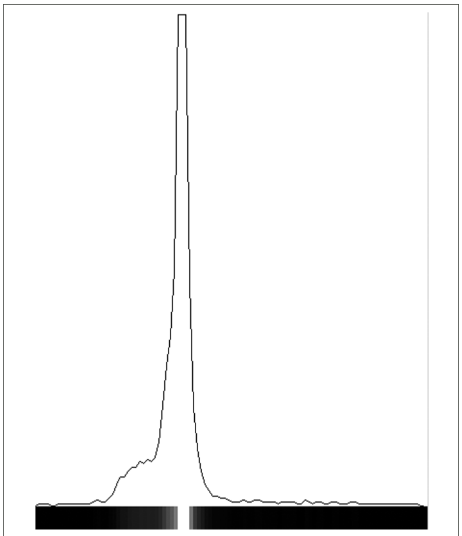


Figure 6. A brightness graph and returned image of a laser dot.

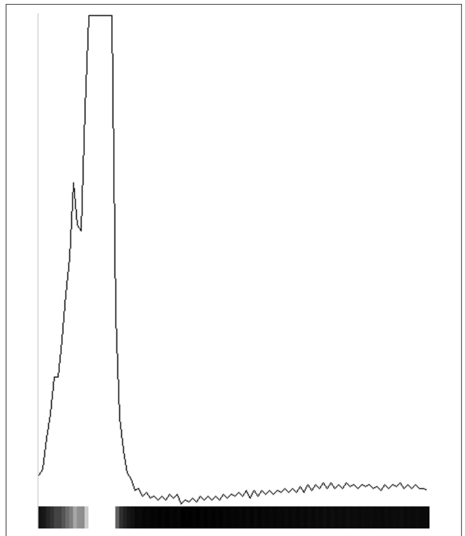


Figure 7. The laser dot for a short distance.

Ready, Set, Go

Okay, your sensor is now complete. Let's look at what it will take to get it going. The first thing that you will need to do is write some software to find the laser dots. This isn't as hard as you might think. Lasers are very bright so they make a very nice peak even if you turn the gain and exposure time way down. This makes everything else in the room very dark in comparison. This strategy works for most indoor situations quite nicely. Of course, it can't deal very well with extremely bright areas.

A strategy that you can employ in that sort of situation is to take two images in rapid succession. The first image would be with the laser turned on and the second image would be with the laser turned off. Subtract the second image from the first and the only thing left except for the internal noise of the sensor will be the laser dot.

It is very easy for a person to be able to tell where the laser dot is in Figure 6 but how can a computer figure out where the laser dot is? The strategy of simply looking for the brightest area is a pretty good one. If you look at Figure 6, you can see that the laser dot peaks for two pixels. If you took the first pixel that had the maximum value, then you would be fairly close to the actual measurement. Let's look at Figure 7 now. In this case, the distance being measured is short and the laser dot fills up more of the field-of-view. Also, since the focus is adjusted for distance measuring, the dot is a bit blurry. This results in a large peak.

Obviously, taking the first pixel that peaks first will give you a very inaccurate distance measurement. You can get close by using the middle pixel that has the peak value as your distance measurement.

Using the middle brightest pixel value can get you close to an accurate distance measurement but it doesn't get you all of the way there. In Figure 8, you can see that at longer distances, you don't have multiple pixels that peak. You don't even have a pixel that goes to full brightness. Out at these distances you are going to have terrible accuracy by just using the pixel that has the highest peak. This gives you a clear idea of where the laser is, but at longer distances, the accuracy falls off because of the longer distances between where a pixel's field-of-view intersects the laser dot. Let's look at a way that you can achieve sub-pixel resolution for where the laser dot really is so that you can make much more precise measurements at longer distances.

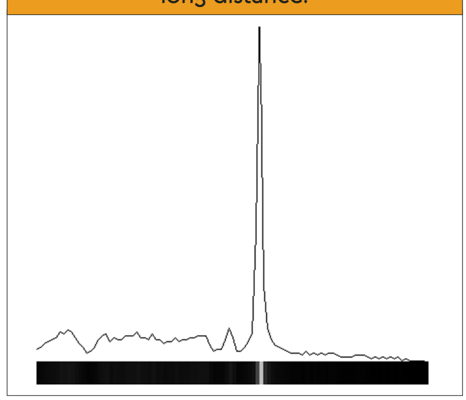
Figure 9. Pixels that are added to the 'total' variable are highlighted in yellow.

The Brightest Pixel of Them All

This strategy starts out like what was described above where you search for the brightest pixel value. If multiple pixels share the brightest value, then the first one is noted. Now that you have figured out where the first brightest pixel is located, you will create a variable called 'total' and load it with the brightness value of the pixel before the first peak pixel. Now add the peak pixel's brightness value to total.

For each successive pixel after the first peak pixel that has the same brightness value, add its value into total, as well. After the last peak pixel is added, add in the brightness value of the next pixel after the peak. Figure 9 shows the pixels that would be added for a narrow and wide peak.

Figure 8. The laser dot for a long distance.



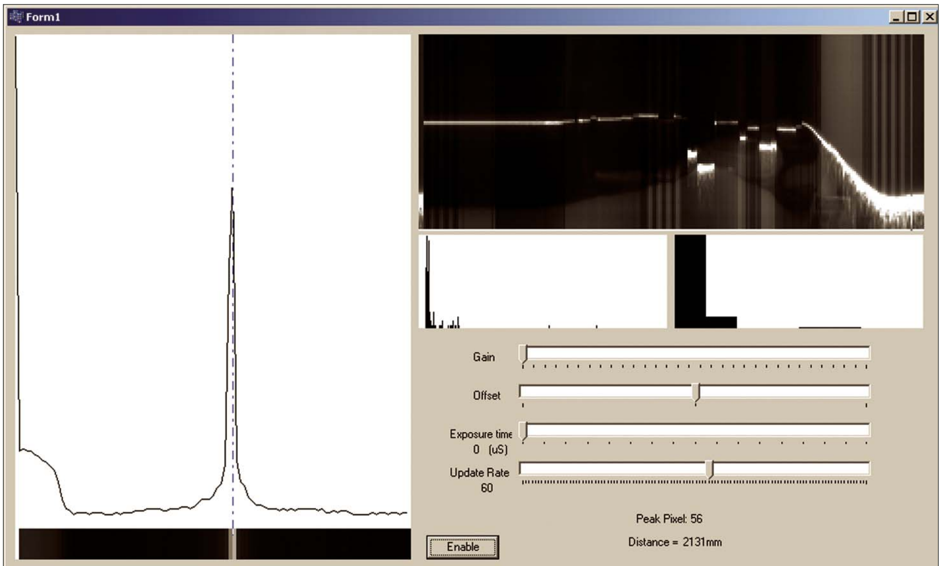
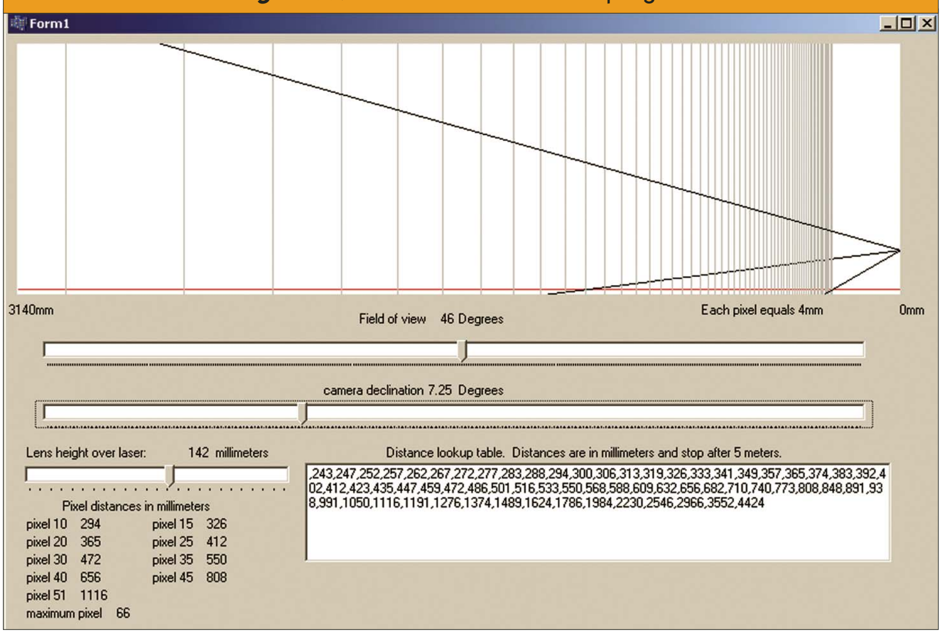


Figure 10. The updated image-receiving program — the vertical bands in the image are due to the flickering of a fluorescent light.

The total variable now contains the total of the values of the peak pixels and one pixel on either side of the peak. Divide this value by two. Now go through the pixels starting at the pixel before the peak and subtract its brightness value from the total variable. If the subtraction results in a negative number, then the center pixel is that pixel. Add that pixel value back into total. This calculation usually gets us the same pixel as what was described before. We can take that value and run it through a lookup table to get an approximate distance. Store that value in a variable called 'theDistance.'

Here is where this code differs from what was shown before: We will now find the difference in distances between this pixel and the next and store that in a variable called 'pixDif.' Multiply pixDif by the value in total. Now divide by the brightness value of the current pixel. Add this value to what is stored in theDistance. This will be a more accurate estimate of the distance than if you had simply looked up the value and had stopped there. In short, this chunk of code finds the center peak pixel and then looks at the pixels on either side of the peak to decide how much to adjust the center location based on their brightness values.

Figure 11. The distance calculator program.



Go the Distance

The last thing that you will need is the actual lookup table to figure out your distances. You can get a pretty good approximation with the software on *SERVO*'s website. The code in the lookup table generator gives approximately the correct distance. The range finder built for this column was within 20 millimeters at medium distances using a table generated with this program. Factors such as misaligned sensors or lens distortion will reduce the accuracy of the lookup table.

To find the best lookup table using the lookup table generator, you should first use the newer version of the image-receiving program presented last month. This program allows you to do other things such as create 2D pictures. It also shows a 256 and eight bin histogram of the image that it is currently processing. The important part right now is that it tells you which pixel has the peak value.

Clamp your range finder in a vise or otherwise position it so that you can adjust the distance to a target. Now run the image receiving program. Move your target until it has a peak pixel that is far away and is a multiple of five. Measure this distance in millimeters. If you only have a measuring tape that is in inches, then multiply the number of inches by 25.4 to get millimeters. Now move your target until your peak pixel is 10. Measure this distance. It wouldn't hurt to measure a couple other distances for pixels that are multiples of five, as well.

Now switch to the other program that draws a representation of the pixel distances. Set the lens height over the laser and the approximate field-of-view for your lens. On the bottom left of the screen is a list of pixel values and their distances. Play around with the camera declination and field-of-view until your largest distance and the distance to the tenth pixel match as closely as possible with what the program says. Put greater emphasis on getting the longer distance to be correct.

When you have a good match, copy

the lookup table from the bottom right side of the screen and use it in your program. If you happen to have Borland C++ Builder, you could copy this lookup table into the source code of the image-receiving program and see what distances your range finder is calculating.

The lookup table provided by the distance calculator program will give you values that are fairly close to reality but to get the highest precision, you should carefully aim your range finder at known distances and generate your own lookup table manually using the actual measurements. This will correct for misalignment and lens distortion.

Figure 12. Self portrait taken with the linear image sensor.

Conclusion

Having a laser range finder on your robot can allow you to dart from place to place at full speed without worrying too much about running into anything. If you were to mount it onto a hobby servo to swivel it around, you could quickly create a map of where your robot can and cannot go. Your robot could navigate from room to room fairly easily if it simply scanned each room for gaps in walls that were the correct width for a door. You could do this with a high degree of confidence since the distances reported by a laser range finder have a fairly high degree of accuracy using sub-pixel calculations, even at longer distances. What could you do with a laser range finder on your robot? SV

RESOURCES

Jameco Electronics
www.jameco.com

Mouser Electronics
www.mouser.com
Sells the TSL3301 chip.

Edmund Optics
www.edmundoptics.com/US/
Sells lenses.

Custom Computer Services, Inc.
www.ccsinfo.com
Sells the C compiler used for the PIC code on *SERVO*'s website.

Borland
www.borland.com/us
Sells the C++ compiler used for the PC code on *SERVO*'s website.

Spark Fun Electronics
www.sparkfun.com
Sells a laser module that you could use for your range finder.

COMBAT ZONE

Featured This Month

PARTICIPATION

Welding Safety

● by Steven Kirk Nelson, Team Kiss

Welding is really safe for the most part. Welding is also fun. But you always have to pay attention to what you are doing and the environment you are doing it in.

What to Do

● Always wear a welding helmet or goggles with the correct shade of lens.

● Wear protective clothing made from 100% cotton or leather or Nomex welding jackets and pants.

● Always wear welding gloves made from leather or Nomex.

● Remove anything that can burn or catch fire from your welding area.

● Remove all gas cans and fuel tanks, solvents or paint cans from your welding area.

● Remove all electrical cords, power tools, and circuit boards from your welding area.

● Keep all high pressure cylinders

ALWAYS WEAR EYE PROTECTION!
Recommended Helmet Shade Levels for Welding or Cutting

Type	Shade Level (1)(2)
Gas welding	4-8
Gas cutting	3-6
Plasma cutting	5
Arc welding	10-14
MIG welding	10-13
TIG welding	11-14

(1) The low shade number is good up to about 1/4 inch thick material.

(2) There is a basic rule of thumb that says you should always pick a shade level that allows you to see the welding puddle without noticeable eye strain, but try different lens shades because each person is different. I like an 11 or 12 shade lens for arc or MIG welding steel.

Willy the Welder added some extra protection to the welding glove on his left hand using a reflective pad. Photo courtesy of Steven Kirk Nelson and Pete Maxham.

Pyrotechnic Pete figures he's okay, since he's just doing a tack weld. Photo courtesy of Steven Kirk Nelson and Pete Maxham.

in the upright position and chain them to a stand or something solid and stable.

● Always remove the regulator and install the safety cap before transporting cylinders.

● Communicate with people and other critters or children around you before you strike an arc.

● Keep an ABC fire extinguisher and a steel bucket full of water handy.

● Totally clean your metal before welding. Oil, dirt, paint, and galvanizing contaminates welds and produces toxic gasses.

● Provide good ventilation in your welding area.

● Move the project to a position that is comfortable and easy for you to run a bead on.

What Not to Do

● Weld without protecting your eyes. (Even tack welding for an instant without a proper lens can sunburn your eyes.)

● Weld without covering all your skin and fur with proper protective clothing.

● Wear polyester, nylon, or any other oil-based clothing material.

● Lift or carry high pressure cylinders by their valve or regulator.

● Weld on gas tanks, high pressure cylinders, oil tanks, or anything that has had flammable chemicals in it.

● Drip hot steel onto your torch hoses. (Hose fires are bad.)

● Panic if you start a fire. (Put the fire out while it is still small.)

● Try fighting a big fire. (If an extinguisher or five gallons of water won't put it out quickly, leave the shop and call 911).

● Breath the toxic smoke. (The breathable air is usually *better* on the floor. Crawl out of the shop quickly if the air gets bad.)

If you have any questions about welding or want to learn how to weld, take a welding class. Most junior colleges or adult education programs offer them. SV

"Build Safe, Build Mean, Build Strong"

Steven adds some bonus wisdom along with his buddy Mat Maxham (Team Plumb Crazy). We were just comparing burn scars last weekend. We've both done it wrong several times. Many of our bots have left a lasting impression on our frail human bodies. It's easy to be safe when you're not in a hurry. The period of time required to inspect a recent weld, using you bare hands, is directly proportional to the heat applied to the weld. The hotter it is, the shorter the inspection time. Pass on what you have learned.

Troubleshooting Radio Problems

● by Jeffrey Scholz

Alright, you wired up your robot, screwed down the lid, set it on blocks, turned it on, and the robot's wheels start twitching. You tweak the trims, but the problem isn't solved. After fully extending your antenna, the glitching decreases, but doesn't fully stop. You can tolerate a little uncontrolled movement, but when you put the robot in the arena, the jerking goes through the roof, and the robot drives all over the place.

The aforementioned scenario is one most combat robot builders face at some point. This is called "reception issues." Reception issues result from a fault within the transmission of the signal during its conversion to a pulse-width format in the receiver. What can you do to fix this problem? Well, here are some of the causes and the solutions.

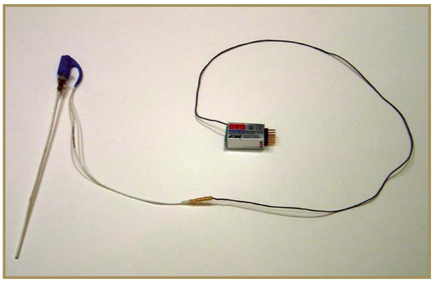
Causes

There are three things that cause reception issues: a weak signal, a corrupted signal, or a bad radio system.

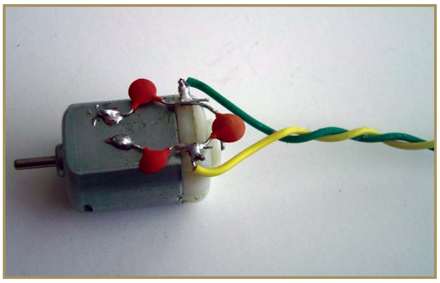
Weak Signal

● Transmitter battery is low. Before you do anything else, fully charge the battery. Also, remember that the battery's performance fades over time, and may need to be replaced.

● The transmitter and receiver are



A Deans antenna attached to the receiver's antenna.



Capacitors installed and wires twisted. The foil has been left out.

too far apart. Keep in mind these should never be over 85 feet away from you. Very few arenas are greater than 60' x 60', so if you are testing your robot from 200 feet away, that's overkill; 100 feet for "bigger bots" is a good target. For insects, aim for an excessive minimum of 20 feet for a comfortable safety margin.

● The antennas are oriented incorrectly. If you could see radio signals, you would observe a doughnut expanding and traveling outward 90 degrees from the transmitting antenna. The receiver accepts the signal best when both antennas are parallel. As such, both antennas should never be pointed at each other, but rather kept vertical to ensure they are always aligned correctly.

Unfortunately, the antennas attached to the receivers are often too long to be placed vertically inside a robot, and often need to be coiled around something. What is the best way to hold your antenna without letting it just run all over the place in your robot? You may have seen people wrap the antenna around a drinking straw; this works, but a better way is to thread the antenna in and out of corrugated cardboard.

Remember not to cut the receiver's antenna; the antenna's length is relative to the radio's wavelength. To remedy a clipped antenna, attach a Deans whip (**www.robotmarketplace.com/marketplace_rc.html**).

● Conductive surfaces are attenuating with the signal. Ever had the experience where you are in the car listening to the radio and when you pass under power lines or a tunnel, the radio emits static? The receiver will experience something similar if it's encased in metal, or if it's in a metal arena. Replace the robot's lid with a non-conductive material such as polycarbonate to give an unobstructed path for the radio signals. Lastly, you can put a Deans antenna on as mentioned earlier. This will dramatically improve reception from inside a metal box. Just remember not to mount the Deans to any conductive material, such as metal or carbon fiber. Likewise, keep the transmitter's antenna away from conductive surfaces.

Faulty Signal

Common causes of corrupted communication:x

● Brushes in electric motors cause interference. When sparks jump from the brushes to the commutator, EMI (electromagnetic interference) is produced. You can alleviate the interference by a) switching to brushless motors; b) installing .01 µF to .1 µF capacitors as shown in the photo; c) twisting the wires from the motor; and d) putting aluminum foil over the plastic brush housing to "foil" the EMI.

● Long wires. These act like antennas themselves and send faulty signals all over the system. Keep them as short as practical.

● ICE (internal combustion engines) and ingitors. Place metal around the sparking components.

Weak Radio System

Possible faults in the radio system:

● Voltage sagging from motors starting up starves the receiver of power. If your reception goes bad during a weapon spin-up, this is the cause. The solution is to a) use a larger battery; b) reduce the load on the motors; or c) use a separate battery for the radio.

● The receiver died. During an impact, the microscopic wires in the integrated circuits may break. Or, the jolt can temporarily short out a voltage to the receiver, frying its components. The only way to fix this problem is to get a new receiver and to shock mount it with foam or rubber.

● The frequency crystal broke. These things are delicate; swap in a different one. Also, if the frequency crystal is not making a good connection in its socket, the system will work intermittently. Hold it steady with a removable adhesive.

● Non-radio components are creating issues. Doublecheck all your connections and electronics parts, such as the ESC and the gyro.

● I've tried everything; I am still not getting good reception. Well, here's the most expensive answer: Upgrade to a pulse-code modulation (PCM) or 2.4 GHz system. PCM is a lot more resistant to EMI than FM, but 2.4 GHz is (for all practical purposes) immune. Google up "Spektrum Radio" for a variety of sources selling 2.4 GHz radios. The Robot Marketplace sells "bot friendly" (tournament legal) Spektrum receivers. SV

Jeffrey Scholz is a high school sophomore who started in autonomous robotics in 2003, and combat robots in mid 2005. More information about Jeffrey and his robots is on his website at **www.freewebs.com/teamhammerbros**

BUILDING BASIC INSECT BOTS — A GUIDE TO GETTING STARTED

● by Kevin M. Berry

There are lots of big name builders out there who like to compete in small weight classes. It's also a natural entry point for people excited about the sport, but not wanting to start with a large investment. This article is intended to be a handy reference guide to get a new builder started, or help a veteran moving into small bots. Specifically, we'll address parts and tips for a 150 gm, one-pound, or three-pound combat bot — typically lumped into the category "insects."

Because of the incredible array of products and performance, it is very difficult to make a comparison based on that alone. There's also the difference between buying new or used parts. Here, I've tried to level things by comparing equivalent equipment, balancing new and used prices, and just generally being a fair judge.

The First Question ...

Is always, "isn't there a kit out there?" The answer is yes, I'm happy to say. Three vendors currently have kits available.

Inertia Labs (**www.inertia-labs.com**) has a milled aluminum chassis, motors, wheels, top armor, screws, receiver, crystal, and a LiPoly battery for around $150. You'll need to buy a speed controller, battery charger, and radio transmitter. An ESC from another source (more later) will run about $80, charger/supply around $100, and Inertia Labs sells a nice basic GWS transmitter for $59, so a total "out the door" price for a running antweight would be $390 plus shipping, including reusable "tools" like the Tx and LiPoly charging setup.

Besides selling the Inertia Labs chassis, The Robot Marketplace (**www.robotcombat.com**) has a couple of packages of their own to get builders started. Their Basic package includes a motor/gearbox combo, an ESC, a Laser 4 Tx/Rx set with crystals, wheels, LiPoly battery, and charger for $238 plus shipping. The builder needs to add a power supply for the charger, chassis, and armor. This very basic set would run under $300.

A more competitive setup is their Advanced Antweight Package, which goes for $306 if you supply your own power supply to the charger ($381 with a top notch power supply). It includes upgraded gearmotors, wheels, and hubs, a bigger ESC, the Tx/Rx set, LiPoly battery, and a carbon fiber sample pack which would be helpful in crafting the chassis and armor.

Team Think Tank markets their VDD kit through the Marketplace. The kit — which is a spinning saw blade weapon platform — uses a unique combination of carbon rods, Kevlar thread, and CA glue to build a light, strong frame. Included is a motor/gearbox combo, wheels, and hubs, all for $100. The builder provides the ESC, battery, radio, and charger. So adding $240 to $300 to cover what's needed, we get a kit price of around $250. The charm of this kit, though, is in the weapon — usually a miniature saw blade or custom cut spinner. To add their weapon kit, figure on another $125 or so.

I've never built one of these kits, but I've sure fought against a lot of them. I think, in general, the sport is moving beyond the Tamiya motor/gearboxes, with state-of-the-art being in higher end gearmotors like we'll talk about later. So, while I think all of these are a reasonable "out of the box" solution, I'd tend towards the Inertia Labs or Robot MarketPlace Advanced package if I was recommending something to a new builder. However, part of what drew me into the sport was the creative aspect, so I'd probably move right to the individual component level with lots of advice. Which is, exactly how I did it.

So How About a Parts List?

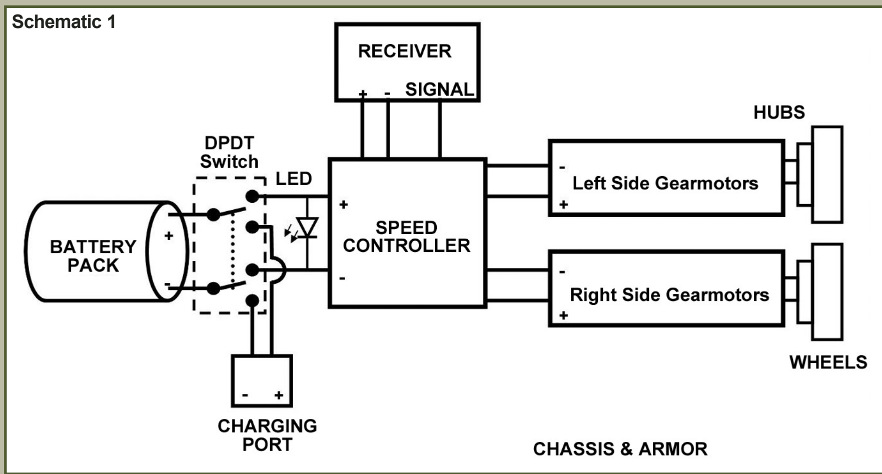
In the August '05 issue of *SERVO* I wrote a design/build report on a beetleweight of mine called "John Henry." I won't repeat the calculations and design tradeoff's here, but will suggest near equivalent parts to build your own "kit."

We'll use Schematic 1 as our source for a parts list.

Batteries

This is one of the toughest items to compare. Either NiMH or LiPoly batteries are "current state" in most insects. Each has their own advantages, and each needs its own type of charger. To determine the battery size ("capacity") and number of cells ("voltage") required, you need to get into the design/tradeoff circle where you'll spend a lot of your time.

Higher voltage (more cells) means higher motor speed. Bigger cells (more current) can mean higher torque. But this increases weight.



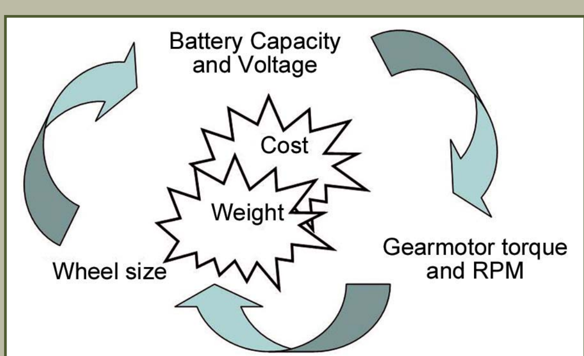
Bigger wheels travel faster (less RPM needed from motor) but reduce torque (pushing power). This vicious cycle, to me, is the most fun in designing a bot. Obviously, it can't be a free-for-all. Somewhere, you pick a wheel size (based on robot configuration), the number and size of motor/gearboxes, and some percentage of your weight budget for batteries. And then, after endless repetition on custom-made spreadsheets, you notice your solution is going to cost you hundreds of dollars. Back to the tradeoffs!

For most builders, I recommend buying professionally-made packs. It's possible to buy cells and solder your own, but (having been down this road) it's frustrating and difficult. Plus, ALL battery related combat failures I've had were on "roll-your-own" packs.

There are tons of places like hobby and R/C stores to buy packs from, but both the Robot Marketplace and Robotic Power Solutions (**www.battlepack.com**) are run by combat folks, and have a nice selection. You'll need a second pack, because often at events, there's not time to recharge between every round. (For estimating purposes, though, we'll stick with one pack to even things out.) For NiMH you'll spend $20 to $40, and for LiPoly $30 to $70. For a good, mid-priced charging setup — which usually means you need a separate 12V power supply — you'll spend around $100.

Speed Controller

There are several good choices out there, all roughly equivalent (note the term "roughly") for insect controllers. The SOZbots (**www.sozbots.com**), Barello (**www.barello.net**), and Scorpion lines (**www.robotpower.com**) are all battle tested, with a long history of success. Another choice, which again I've decided isn't worth the aggravation, is to "hack" the boards out of servos as ESCs. While the 150 gm class may seem to make this worthwhile, there are enough proven systems out there to choose from. Remember our dreaded "wheel of tradeoffs?" Well, motor current and supply voltage are factors in selecting your speed contoller. Better run back to the spreadsheet to make sure we're still okay. Assuming you're designing a fairly standard insect bot, figure $65 to $80 for a new controller.



Radio Systems

New builders often buy a prepackaged "Flight Kit" from one of the major R/C vendors like JR, Futaba, Hitec, Airtronics, or GWS. In that case, your receiver choice may already be made. In selecting a radio control system, you'll need to consider the number of radio channels you need, since it takes two for just driving the bot. If you have weapons, lifters, etc., you'll need more.

I recommend four for a starter set, but it's easy to get into designs needing five or six channels quickly. Also the type and frequency of transmission needs to be decided. For folks that are going to stay in the insect classes for a while, I recommend a four- or six-channel, FM, 75 MHz, PPM type. If there's a good chance the builder will be moving into the bigger classes, then PCM type radios become necessary, but, of course, are more expensive. A minimum transmitter setup will be about $60-$70 not including the receiver, but there are so many parts and options available in flight packs, that price comparison is difficult.

All the above vendors supply receivers. However, I should mention the GWS Nano receiver, which is becoming very popular. Microbotparts (**www.microbotparts.com**) also makes a very lightweight unit. For between $30 and $60, you can pick up a new, high quality receiver. Note: Not all Tx and Rx will work with each other. Do some research to make sure your system will be compatible.

Motor/Gearboxes

There are many, many choices in this area. I mentioned earlier that the class has moved beyond the plastic, "Tamiya" type gearboxes. Pretty much the current state for gearmotors fall into two midrange brands: Copal and Banebots. Some also use Sanyo and Solarbotics motors. Some builders use the more expensive Maxon motors. Here, we'll stick to the midrange technology.

Look for all metal gears and consider supporting the output shaft of the motor either at the gearbox, the outside of the wheel, or both. The Banebots motors will run under $15, and the Copals between $20 and $30.

Wheels and Hubs

Again, many choices here, most from the R/C folks. Robot Sumo has also generated some great wheels and tires. The "default" option for many is the Dave Brown Lite Flite series. Others tap into the incredible array of products from online stores and hobby shops. To attach wheels to motors, there are prop adaptors in all sizes, some custom hubs (available on Robot Marketplace), and endless other creative ways to attach wheels.

Figure about $8-$10 per wheel/hub as a good starting point.

Oh, by the way — you've now selected your batteries, speed controller, gear-motors, and wheels, Do the numbers still line up? Did you calculate for two motors when sizing batteries, then chose to go with four wheel drive later? (Oh, my, it just never ends.)

Chassis and Armor

No way to compare this one. Chassis range from CNC milled masterpieces to plastic boxes. I will say, however, that today's insect arena is a vicious place, and titanium, Kevlar honeycomb, or carbon fiber are not overkill.

The way I think about this is to picture my bot getting hit by a gaso-line powered garden edger, tossed four feet in the air onto concrete, and repeat. For three long minutes. Components must be mounted securely and be well protected.

Final Thoughts

There are lots of options available, but you can see that getting started in robotics doesn't have to cost you an arm and ... a servo!

The total for our DIY "kit" is $150-$300 for the bot, and $200-$300 more for charger setup and transmitter. This is without chassis or armor, and all those "little extras" that run up the cost of any project.

This has only been an overview, and every topic, product and vendor is subject to much debate in the combat community. Still, having built and fought many insects over the years, I think it's a fair representation of the current "state-of-the-art." SV

EVENTS

RESULTS — June 12th - July 10th

ROBOGAMES

RoboGames 2006 was held June 16-18 at the classic Fort Mason venue in San Francisco, CA. Presented by Combots, over 150 combat bots registered and, according to all involved, the fighting was fierce — even by today's standards. The international flavor was highlighted with Canada winning the SHW class, and Brazil placing 1st and 3rd in MW and BW. Visit **www.robogames.net** for all the details. Results are as follows:

- *Superheavyweight* — 1st: "Ziggy," flipper, CM Robotics; 2nd: "Sewer Snake," Plumbcrazy, lifter.

- *Heavyweight* — 1st: "Original Sin," Pirhana, wedge; 2nd: "Brutality," Demolition, wedge; 3rd: "SJ," Blackroot, lifter.

- *Middleweights* — 1st: "Stewie," Lunch Money, Drum; 2nd: "Ice Cube," Toad, plow; 3rd: "Touro," RioBotz, Drum.

- *Lightweight* — 1st: "Son of Whacky Compass," Hawg, bar; 2nd: "Death by Monkeys," Death By Monkeys, wedge; 3rd: "Hexy Jr," WhoopAss, flipper.

- *Featherweight* — 1st: "Killabyte," Robotic Death Company, full body spinner; 2nd: "Gnome Portal," Robotic Hobbies, hammer; 3rd: "BOT-6:00," Cerebral Machines, wedge.

- *Hobbyweight* — 1st: "Darkblade," Sawzall, spinner; 2nd: "Bullet," Target Practice, wedge; 3rd: "Lil Shocker," SMC, wedge.

- *Beetleweight* — 1st: "Mini Touro," RioBotz, drum; 2nd: "Itsa?," Bad Bot, spinner; 3rd: "Titanium Chipmunk," Slackers United beater.

- *Antweight* — 1st: "MC Pee Pants," Fatcats, drum; 2nd: "Switchblade," Sawzall, beater; 3rd: "Team DMV."

- *Fairyweight* — 1st: "Microdrive," Misfit, wedge; 2nd: "Change of Heart," Misfit, wedge; 3rd: "VD," Fatcats, saw.

SHW Ziggy sends HW Sewer Snake into low Earth orbit at RoboGames 2006.

The Robot Fighting League's South East Championships
Battle Beach
RFL

Battle Beach Lite 3 was held June 24, in conjunction with the city of Ormond Beach, FL, Hurricane Preparedness Seminar. About 20 bots participated, with all the usual SECR-Florida culprits supplying the logistics. Results are as follows:

- *UK Ants* — 1st: "Electric Eye," Cerberus, Lifter; 2nd: "Strike Terror," Team V, Pusher; 3rd: "Skeeter from Hell," Team V, Pusher.

- *Antweights* — 1st: "Ultimate Ultimatum," Overvolted Robots, Undercutter; 2nd: "Zeromancer," AG Robotics, Flipper; 3rd: "Ant from Hell," Team V, Verticle Spinner.

- *Beetleweights* — 1st: "R.M.D. (Ron Must Die)," Team V, Dustpan; 2nd: "Ron," Overvolted Robots, Dustpan/Saw/Karate Chop Action; 3rd: "Beetle from Hell," Team V, Undercutter. **SV**

Saw Blade vs. Saw Blade beetleweight at Battle Beach IIte, as top ranked Veteran Ron pins newcomer Playgerist.

TECHNICAL KNOWLEDGE

Thoughts on Low-profile Combat Robots

● by Charles Guan, Team Test Bot

In the modern robot fighting arena, there are many things that can flip your 'bot over. Spinning weapons seem to get more powerful by the day, and cross-arena tosses are not uncommon with flipping and ramming robots. There's a pretty good chance that your robot will end up on its head at least once. If you can't recover from it, your match is probably over.

There are also numerous ways to counter being flipped. For instance, if your weapon swings hard enough to knock or "gyro-dance" the robot back over, then the match can continue. Some builders, however, design their robots to reduce the chance of this in the first place by building them very low to the ground.

The term "low profile" is rather subjective and broad. Generally, it describes a robot whose height is very small compared to its other dimensions or footprint. Some say these robots sport a "pizza box" look.

I am a fan of this type of design for a few reasons:

- Having the center of gravity very close to the ground inside a large footprint means a higher chance of the robot staying stable. This is helpful for powerful weapon platforms which must withstand their own impact reactions. Many fighting robots have flipped themselves over on hits.

- Strong chassis are easily constructed from readily available bar stock. Metals, plastics and, of course, wood, if you are so inclined, all come in long beams that can be attached together quickly.

- The frame and armor can be integrated into one unit. Since the robot has little vertical dimension, this can translate into thicker armor as there is not as much area to cover. This can be especially advantageous in the modern fighting robot world of high-energy spinning weapons.

- Beauty may be in the eye of the beholder, but I like the sleek and clean design of a flat robot.

There are things that need to be considered when you build low. These issues make "lowering the bar" a challenge in itself. As a case study, I will use my 12-pound Hobbyweight class robot, Test Bot, a 1.5 inch tall "pizza box" with two inch wheels and a four-bar linkage based lifting arm, which is seen in Figure 1. However, since I have never built a fighting robot larger than the 12 lb class, it will have to be my only example.

When the lower limits of height are pushed, more exotic options need to be looked at to do simple things, such as move. To keep the price low while searching for drive motors (so that I could afford it), I had to shy away from luxuries such as rare-earth magnets or completely custom gearboxes. Instead, I settled on something I had done before — creating a barebones mount for the lowly cordless drill motor, ubiquitous in small robots.

Figure 2 shows the result of this modification. The motor and gearbox are normally housed within a plastic shell that aligns the parts and offers structural integrity. The plastic shell had to be done away with for this design, as it would be significantly thicker than the bot. So using plastic and metal bar stock, I created a "sand-

FIGURE 1. Test Bot, a one foot square Hobbyweight class combat robot which is two inches tall at the wheels, armed with a four-bar lifter.

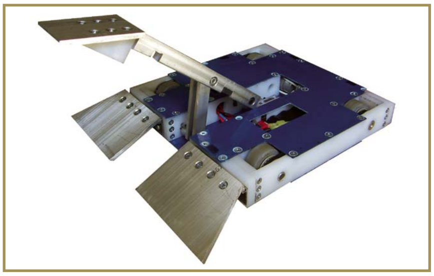
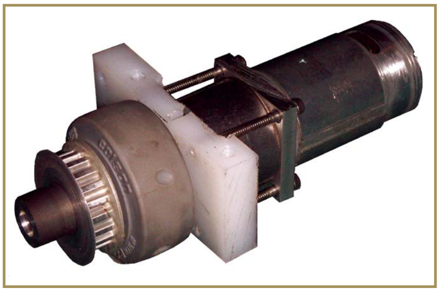


FIGURE 2. A modified drill gearbox on a mount that will be integrated into the chassis.



wich mount" that aligned the two parts using drilled dimples and holes, with long cap screws holding the entire thing together. The UHMW polyethylene plastic (which the bot is made out of) is also used as a bearing material. What resulted was a compact drive solution at a minimum of 1.5 inches square. The drive hub was turned on a lathe by a very helpful friend. For wheels, I chose two inch diameter Colson Caster rubber wheels from another robot; also available for a few dollars each from places such as the almighty McMaster-Carr (**www.mcmaster.com**) and The Robot Marketplace (**www.robotmarketplace.com**).

Of course, stock solutions exist that are much stronger than something I rigged up in the garage.

- Team Whyachi (**www.teamwhyachi.com**) offers 1.5" and 1.8" square profile planetary gearmotors, and wheels to go with them.

- BaneBots (**www.banebots.com**) has a whole line of small gearmotors that are suitable for a flat robot. They even have 1" diameter gearboxes. I have yet to attempt a robot lower than 1.5". But maybe soon!

Both these solutions are low-cost. If you are extremely ambitious or have a huge shop, you can try making your own gearboxes to suit a completely custom design.

For the chassis, I went with the usual "square pizza box" type design using 1.5" wide, 0.5" thick UHMW polyethylene from McMaster-Carr. I am also a UHMW enthutiast due to its features — light, easily worked with, cheap, and high impact strength. Since I used white UHMW, the robot looks even more like a pizza box. The bars had holes drilled where things needed to be, as well as indentations and channels cut out where the motors would sit. The top and bottom plates were made of a fiberglass-epoxy composite called Garolite; same stuff used in high-end circuit boards. At 1/16" thick per side, the robot was still invertable with 2" wheels.

The four-bar lifter was the most challenging part of this build. Four-bars have been used successfully on robots in the past, such as the famous BattleBots contestant Biohazard (which, I might add, is also designed extremely low and was a huge inspiration to me). I had never built a mechanism like this before, and it took several cardboard mock-ups and nights playing with Autodesk Inventor to settle on the design. The arm folds to 1.5" and measures 8" at full swing. This is where the trouble in the design was.

Unfortunately, in order to pack the arm into such a low space, I had to make the linkages fold nearly horizontal. This meant the mechanism was at a "toggle" point — where mechanical advantage is nearly zero when being driven. I didn't realize this until it was all done — on applying power, the arm locked up completely. Score one for bad linkage design. In order to remedy this, I had to redrill mounting holes lower in the chassis and higher in the arm itself to make the mechanism not lock up every time.

The remainder of the build process was not too unique. Batteries presented another issue, as only a few cells with meaningful capacity exist that are under 1.5" standing vertically. Those that do exist were a tad out of my price range — this bot was a complete budget build. I settled for cheap 3,800 mAH Sub-C cells, times 10 for 12 volts, that were an absolute steal at under $4 each. They were probably not matched or high performance, but I wasn't aiming to build an R/C race car anyway. As the cells had to lay flat, they took up a significant amount of space.

Better choices for batteries exist in the form of lithium-polymer cells, or "Lipolies." I have never personally touched a lithium battery, but they are renowned for their high energy density and low weight and volume. A large Lipo pack would have done wonders for Test Bot instead of having one-third of the interior volume taken up by batteries. Lithium batteries also exist in very prismatic and flat forms, perfect for low interior height.

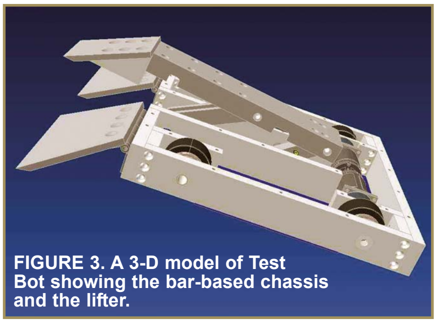


FIGURE 3. A 3-D model of Test Bot showing the bar-based chassis and the lifter.

Some final thoughts and considerations on low profile robots:

- *Ground clearance*. This is an issue in any weight class, not just Hobbyweights. Test Bot has 3/16" of clearance on a good day, and arena floor features can make a world of difference. One essential task is to countersink everything that sticks out on the bottom and is responsible for holding the chassis together. I made the mistake of using button-head screws on a previous robot, and it would hang up on the floor after only a few feet of movement.

- *Top and bottom armor*. You can armor the sides with as thick of material as you want, but the broad flat pizza box designs are especially vulnerable to overhead weapons.

- *Speed*. Likely not much of an issue with a wider selection of drive motors, but smaller wheels obviously mean less distance traveled per motor revolution. I was stuck with high-reduction drill gearboxes for Test Bot, which meant it moved at a very slow pace. Make sure plenty of speed and torque calculations go into the design phase!

- *Appearance and function*. This arti-

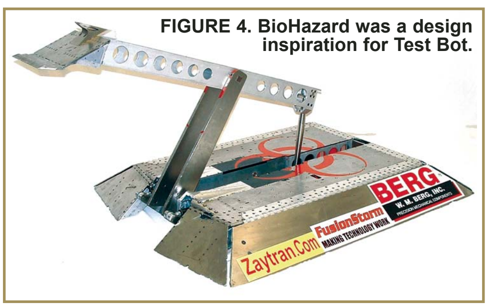


FIGURE 4. BioHazard was a design inspiration for Test Bot.

cle was very provincial in that I focused on my own pizza box design. The possiblities can be very expansive. For instance, weapons need not be contained entirely in the chassis. It especially does not need to be square.

If I didn't build Test Bot on a very low-profile budget, I likely could have reduced the design height even more. However, at that level, the question of practicality becomes rather hard to answer. I will probably stop my designs at 1.5" thick since I now have a relatively proven system for them. If a bot was to be built specifically to test the boundaries of flatness, then things can get pretty exciting quickly. **SV**

PRODUCT REVIEW — NPC 2212 Gearmotor

● by Tim Wolter

National Power Chair has earned its reputation as the premier supplier of gear motors for robotic combat. But the experience of most builders is skewed towards the heavier end of their product line.

Recently we bought a set of their NPC 2212 gear motors with the notion of building something in the lighter weight classes. The 2212 lists at 5.1 pounds, although it is actually a bit less after you trim off some extra drive shaft.

We ended up building a 30 pounder, mostly to fill out this weight class at Mechwars 9, and to test some anti-spinner ideas we had been kicking around. "Arbor Mortae" had sufficient speed, and could push opponents around fairly well. The armor, made of green wood and Kevlar fabric, took a respectable number of hits before it gave way, after which things got ugly.

On "autopsy," we found one 2212 had a badly bent shaft and that the brass output gear had sustained some stripping. I am told the gear was made of brass for noise reduction, hardly a high priority in robotic combat. But realistically, the damage would have been less if we hadn't trimmed off the extra weight. It would have supported the far end of the drive shaft with a bearing.

So, the best application for this motor would seem to be the 60 pound weight class.

The 2212 is rated for 12 volts, but Norm Domholt at NPC says you could probably run them at 24, although it might shorten their life expectancy. At a reasonable $155, they look to be a good motor for the entry level builder. **SV**

Tim and Karl Wolter build robots of all sizes. They favor comic relief machines, and have pioneered the "weaponization" of Spam, Christmas Fruit Cakes, and Barbie Jeeps.

EVENTS

UPCOMING — September and October

The Texas Cup — September 9th, Carrolton, TX. Presented by Southwestern Alliance of Robotic Combat.

Classes from 150 grams up to 120 pounds. Venue is Mike's Hobby Shop (**www.mikeshobbyshop.com**). Spectator admission: $2.00, limited seating. VIP passes required for restricted area overlooking arena. Registration limited to 16 bots in each class. Prizes: First and Second place only. Medallions will be awarded. Sponsorship certificates will be awarded. Format: Standard double elimination, all classes. This is a 2006 qualifier for the RFL Nationals. See **www.robotrebelion.net** for more details.

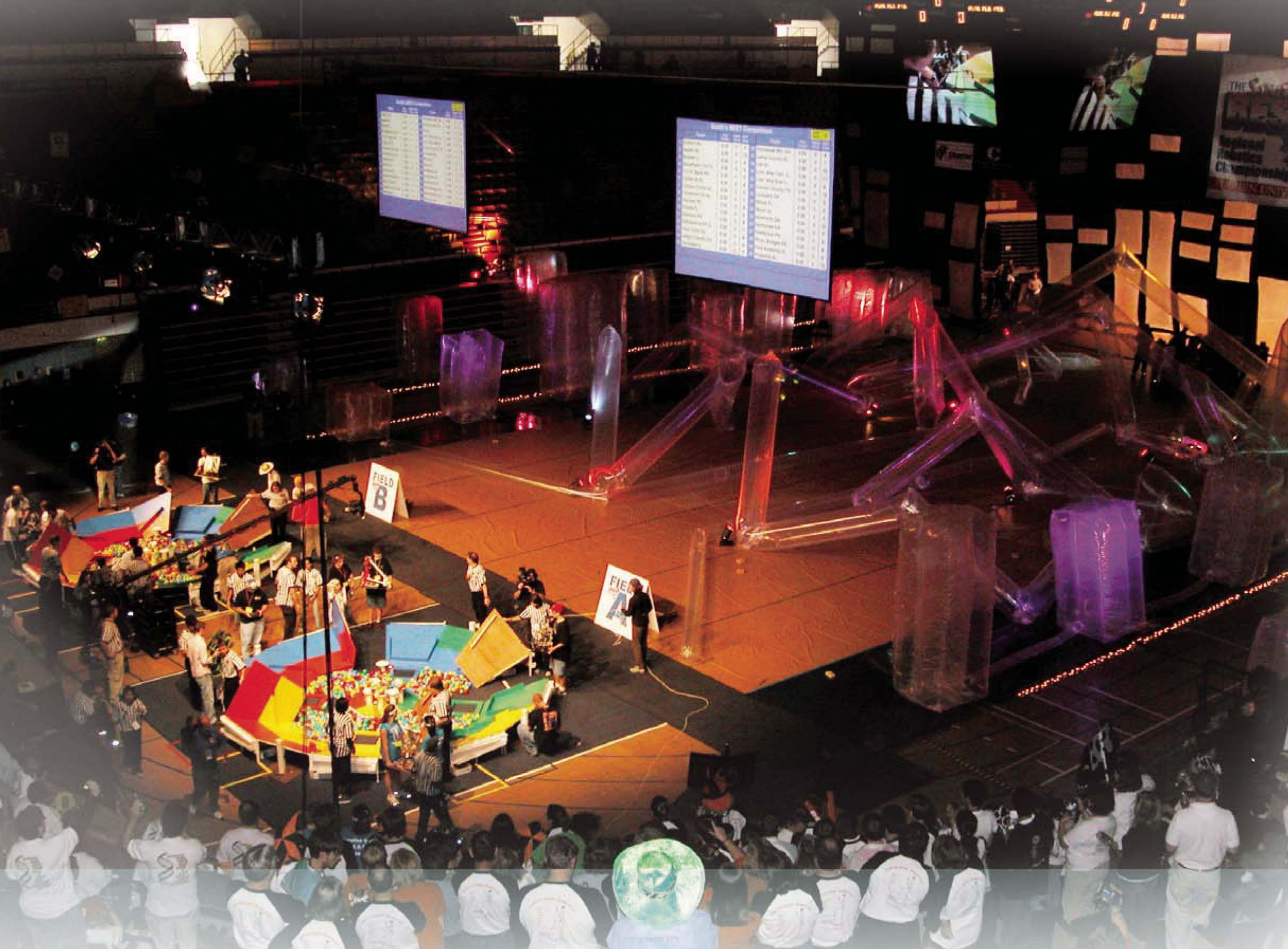
Fall Whyachi House of Robotic Entertainment 2006 — September 16-17, Dorchester, WI. Presented by WHRE.

No pit passes, no limits on pit members, no fee for spectators, all entry fees put into prizes and cash for competitors. Just fighting robots. Floor is epoxy painted cement with traction compound. The arena is 13.5' x 24'.

Halloween Robot Terror — October 28, Gilroy, CA. Presented by California Insect Bots

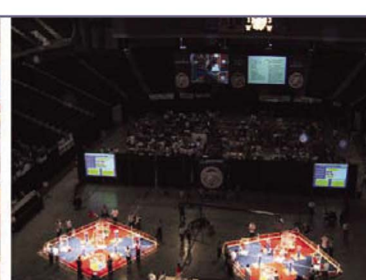
Venue is Hobby World, 6901 Monterey Rd., Gilroy, CA 95020. Bot Gauntlet Baron Dave says "This is open to Fleas, Ants, and Beetles. The only rule I have for the bot costume contest is you must use the bot you brought to fight with. You will NOT be fighting with your bot costume on your bot.

You are welcome if any builders or team mates want to wear a costume to this event, but please remember to make it safe for anyone that's working on the bots. Weigh-in starts at 10:00 am and fighting starts around noon. It costs $20 per bot with prizes for first, second, and third place in each weight class. For fight rules, go to **www.sacbots.com/eventrules.html** **SV**









NEW PRODUCTS

CONSUMER ROBOTS

Roboreptile

WowWee Robotics™ has set loose the ultimate herpetological species ... Roboreptile™. A highly interactive, seeing, and hearing cyber-reptilian, Roboreptile is one of the most sophisticated, fast, and animated creatures to come out of the WowWee workshop. WowWee is best known for manufacturing technologically advanced blockbuster hits such as Roboraptor™, Robopet™, Robosapien™, and Robosapien V2™.

With his low menacing stride and striking animations, Roboreptile is an impressive mix of great mobility, multi-sensory technology, and a fiery personality. Watch as the 2-1/4 feet long adventurous beast awakens with a roar, whips his long tail, and springs into action.

Equipped with a 28-function remote, Roboreptile has direct control functions — free roam, program, and guard mode capabilities. Incorporating a complex array of sensors and advanced artificial intelligence, this futuristic reptile achieves new levels of awareness. Roboreptile's acute vision, touch, and stereo sound sensors allow him to roam his environment autonomously and avoid obstacles, and make him a formidable protector when in guard mode.

With his highly flexible neck, Roboreptile will scan his environment with Infrared Vision Sensors for any "prey." His vision sensors enable him to detect movement and avoid obstacles. Roboreptile is also equipped with a touch sensor on his back, which allows him to respond to human interaction by performing a short animation. Roboreptile relies on his sharp sonic sensors, located on either side of his head, just behind the jaw. The sonic sensors enable him to detect sharp, loud sounds such as a clap. When he hears a sound, he will turn and run towards it.

Roboreptile's realistic biomorphic movements and cutting edge dynamics will fool your senses into believing that you are looking at the real thing. This impressive technology enables Roboreptile to cycle through four different gaits. He has the ability to run and walk on all four feet then switch to a bipedal mode and attack on his two hind feet. With the press of a button, Roboreptile will jump, and he turns on a dime to surprise his adversaries. His tail whipping action will defend him against any enemy.

Once powered on, Roboreptile's default state is hungry, aggressive, and active. Using his keen senses, he will start to explore his environment, attacking, roaring, or moving away from anything he sees or hears. Approach with caution as this reptile knows how to bite. Activate the "Feed" button on the controller and he will track down the food signal. Once his appetite is satisfied, Roboreptile becomes a bit more passive and lethargic; this is the time to tame your beast by placing his hood over his eyes. While hooded, Roboreptile will become docile and subdued; remove the hood and you better stay on your toes.

Using the direct control function, you can take command of Roboreptile. Have him perform multiple realistic actions such as snapping, running, jumping, and whipping his tail. Enter the Program Mode, and program Roboreptile to perform a series of 20 different movements or animations.

Place Roboreptile in guard mode, and he will protect your area from unwanted visitors. Standing on his hind legs, Roboreptile's vision sensors and sonic sensors keep him alert. He will respond to a sound or movement by either letting out a big roar, or performing a program that you entered.

Easy to use, Roboreptile is fully functional right out of the box, no assembly required. Equipped with volume control and a demo mode, all functions are handled by an easy-to-use remote control. With six AA batteries (not included) for the Roboreptile and three AA batteries (not included) for the remote, you can enjoy continuous entertainment.

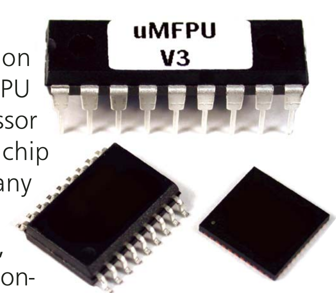
Roboreptile (ages 8 years and up) will be available nationwide this fall for an approximate retail price of $120.

For further information, please contact:

WowWee Ltd.	Website: **www.wowwee.com**

CONTROLLERS & PROCESSORS

uM-FPU V3 Floating Point Coprocessor



Micromega Corporation has released the uM-FPU V3 Floating Point Coprocessor chip. The uM-FPU V3 chip interfaces to virtually any microcontroller using an SPI interface or I^{2}C interface, making it ideal for microcontroller applications requiring floating point math, including sensor readings, robotic control, GPS, data transformations, and other embedded control applications.

The uM-FPU V3 chip supports 32-bit IEEE 754 compatible floating point and 32-bit integer operations. The new chip is 10 to 20 times faster than previous versions for all instructions, and up to 70 times faster for advanced instructions. New instructions provide support for faster data transfer, matrix operations, multiply and accumulate, unit conversions, and string handling. Two 12-bit A/D channels are provided that can be triggered manually, by external input, or from a built-in timer. A/D values can be read as raw values or automatically scaled to floating point values. Local data storage has been expanded to include 128 general-purpose registers, eight temporary registers, 256 EEPROM registers, and a 256 byte instruction pipeline.

An Integrated Development Environment (IDE) makes it easy to create, debug, and test floating point code. The IDE code generator takes traditional math expressions and automatically produces uM-FPU V3 code targeted for any one of the many microcontrollers and compilers supported. The IDE also supports code debugging and programming user-defined functions.

User-defined functions can be stored in Flash using the IDE, or stored in EEPROM at run-time. Nested calls and conditional execution are supported. User-defined functions can provide significant speed improvements and reduce code space on the microcontroller.

The uM-FPU V3 is RoHS compliant and operates from a 2.7V, 3.3V, or 5V supply with power saving modes available. SPI interface speeds up to 15 MHz and I^2C interface speeds up to 400 kHz are supported.

The chip is available in an 18-pin DIP, SOIC-18, or QFN-44 package at a single unit price of $19.95 with volume discounts available.

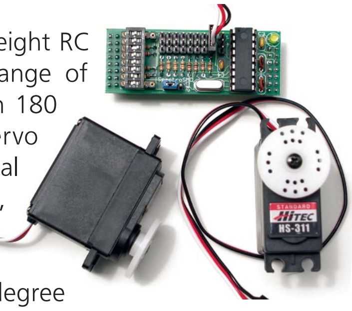
For further information, please contact:

Micromega Corp.
1664 St. Lawrence Ave.
Kingston, ONT K7L 4V1
Canada
Tel: **613 • 547 • 5193**
Website: **www.micromegacorp.com**

MOTOR CONTROLLERS

SpectroSMC

Spectro Technologies, Inc., announces the addition of SpectroSMC to its line of smart plug-in device modules. SpectroSMC is an eight channel servo motion controller that provides accurate positional control of up to eight RC servos through their full range of motion, typically more than 180 degrees. While most servo controllers give a positional resolution of only 255 steps, SpectroSMC has a resolution of better than 1,200 steps for the same 180-degree range, which yields a much greater positional accuracy. Additionally, SpectroSMC has two channels that can be used specifically with servos modified for continuous rotation and is therefore ideal for robot drive systems.



There are more than 18 standard RC servo commands for speed, range, home position, group, and individual moves. Additional commands are available for controlling the left and right side continuous rotation servos when used as part of a robot drive system. These commands allow the user to easily control the direction of a robot by individually commanding the servos to rotate forward, rotate backward, and/or stop.

Building brains for your robot or any other PIC-based project is made easier when using the SpectroBUS development system. Using the SpectroBUS development board, simply "plug-in" the functionality that you need — RS232 serial communications with a PC, LCD display driver for any size LCD up to 80 characters, keypad decoder driver for matrix keypads up to 16 keys, and the new eight channel servo motion controller. Use the proto boards to build custom circuitry and then just plug them in. With the SpectroBUS development system, when you are done using the system for one project, you can dismantle and re-configure it for your next project.

For further information, please contact:

Spectro Technologies, Inc. Website: **www.spectrotech.net**

Robotics Showcase

by Dave Calkins

CampusBot

Spanish Robots Show Flair at Spain's Biggest Geek Fest

One of my favorite TV shows is "Iron Chef." I'm not a big foodie, but I love watching two cooks from different cultures take the same ingredients, and each will make something entirely different from the other. All of the components are the same, and yet at the end, you get dishes that not only taste different, but look different and are prepared uniquely from each other.

What Does This Have To Do With Robots, You Ask?

Spain. Or more specifically, Spanish robot builders. They have the same servos, microchips, and sensors that US robot builders do, and yet while attending CampusBot in Valencia Spain this July, I saw robots that were very different from the ones that US builders typically make. They had line followers, snake robots, hexapods, and wheeled robots.

"Well duh — so do we!" you might be thinking. Ah, but my friend, the Spanish robots are not twins to the US bots. US robots are very good, as are Spanish robots. Much as a Spanish paella and American stew are both good. But they are still quite distinct from each other — even though they may use the same ingredients.

Take hexapods, for example. In the US, most hexapods are all skinny legs and quick moving beasties. Spanish hexapods, such as the one CampusBot organizer Alejandro Alonso Puig made, have shorter legs and full body enclosures. It could still scurry around quickly, scare the bejesus out of people who aren't expecting robot spiders to be crawling around the floor, and was just as good as the ones we make. But yet ... it's different. Somehow it moves a bit unusually to my American eyes.

Quantity of ingredients also varied from a US show. In the US, you'd expect to see about 95% wheeled robots and 5% walkers (bipeds, quadrupeds, and hexapods.) In Spain, it's about 30/70. Just as they use fewer veggies in their cooking and more meat (not a bad thing, just ... different), you find more walking robots. I was quite surprised at the number of quadropods. Big ones, skinny ones, bots with two servos and bots with 20 servos. I must have seen more homemade quadropods in three days in Spain than I have in the last two years in the US. Nice ones, too.

In one of the best examples of dirt-cheap walkers, one student built a quadropod out of eight, low-end servos, a single sonar, and a PIC. This little guy could really move — something you're more likely to see in a hexapod with its inherent dual-tripod balancing than in a dual-bipod. But it cruised around the floor of the venue like a sneaky spider soldier. A real treat!

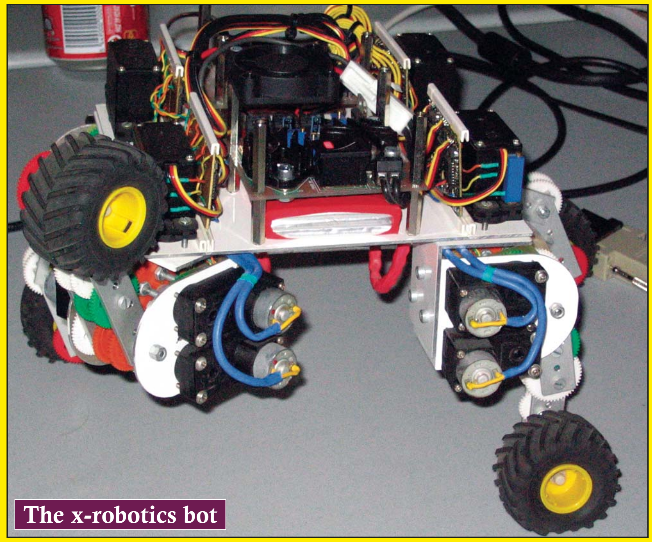
Snakes aren't the kind of cuisine most people look for when ordering, but the specialty of the house was Juan González-Gómez's amazing servo-driven snake bot. All snake robots I've ever seen — even Gavin Miller's amazing bots — cheat. They use wheels. They replicate a snake's motion, be it sinusoidal, caterpillar, or side-winding, but always with wheels on the bottom to eliminate friction and help the bot along. Gomez, however, perfected a system that most closely replicates how snakes really move. There are no wheels on his robots. Just his own servo housings.

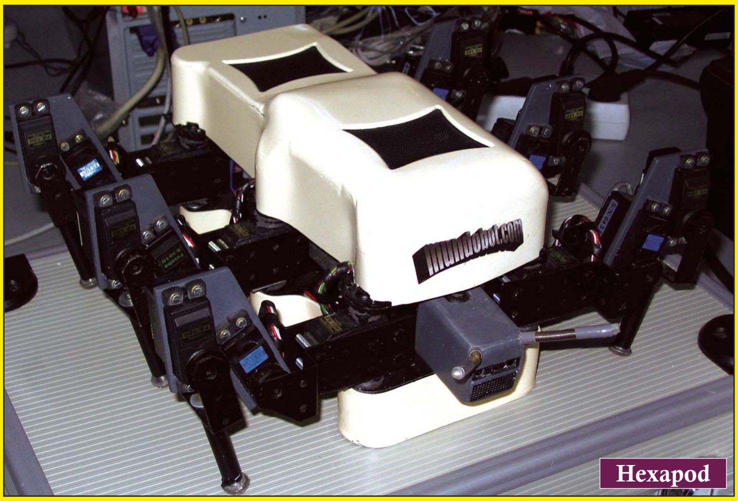
Watching a snake robot skitter across the floor is always cool. But when you pick up Juan's bot and realize that it's got no wheels and can still move the same way any snake can, you're truly awed. Even more inspiring is the fact that his bots are totally modular. You can have as few as two modules or as many as 256 — good for both garter snakes and anacondas!

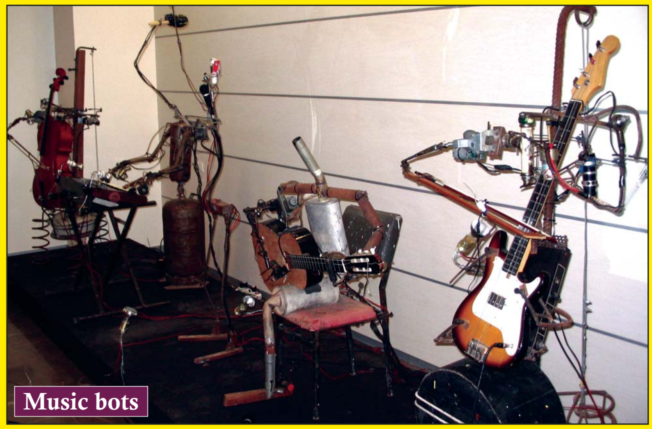
Innovative motion solutions weren't limited to snake bots and walkers. In the past seven years, I've seen a lot of cool robots, but one of the coolest I've ever encountered is the x-robot. Good wheeled robots have some kind of suspension or shock absorption. They might even have independent suspensions — but I've never seen anything like the x-robotics bot. As shown in the photo, each wheel is on a completely independent leg. A leg with three degrees of freedom!

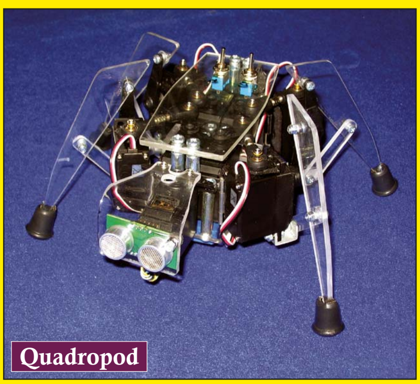
When going through a tight spot, the bot can bring the wheels in closely under the robot. If it needs to go over a rock, it can stretch them out wide. It can rotate them and do a 90-degree shift in vector for an instant shift in direction. It can raise each leg independently to go over rough terrain, even giving the robot knees to allow

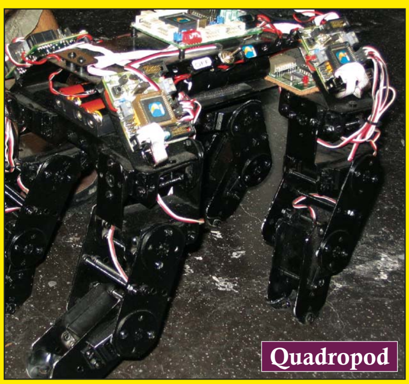
> **“Spanish hexapods … have shorter legs and full body enclosures. It could still scurry around quickly, scare the bejesus out of people who aren’t expecting robot spiders to be crawling around the floor, and was just as good as the ones we make. But yet … it’s different.”**

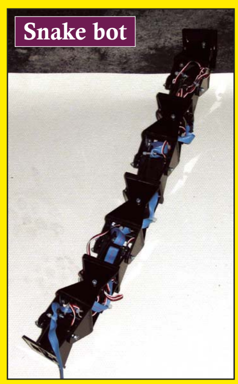

The x-robotics bot


Hexapod


Music bots


Quadropod


Quadropod


Snake bot


R2-D2

for further mobility. The legs can even go over the body of the robot for full invertability. I’ve never seen such a clever marriage of a walker and a wheeled bot. While it was only one foot square, it was probably the most agile robot in existence. This is the kind of bot we should be looking to for the next generation of space explorers!

Speaking of space robots, the most famous robot in the galaxy made a guest appearance at CampusBot. R2-D2 opened the festivities along with Stephen Hawking! Just like us, the Spanish robot building community holds a special place in its heart for R2 — who probably inspired more robot builders than any other robot. R2 wasn’t there to compete, however. Being retired from the movie biz, he just wanted to hang out and spend some quality time on the sunny beaches of Valencia!

An interesting thing about Spanish cooks … erm, robot builders — a good number are women! While many robot builders in the US are men, the percentage is still much too low (hear that girls!). In Spain however, at least 30% of robot builders at the show were women. And like their US counterparts, they build robots that are different from the styles that men build. It was quite refreshing to see so many gals at the show — and not for the usual reason!

One of the highlights of the trip was the robot music group. Spare auto parts and miscellaneous electronics had been salvaged and cobbled together to make a five-piece band. This was no static art piece, but a functional set of robots that played real instruments. The star was the guitarist (just like human bands), who could strum and play slide guitar. A pretty amazing feat for a musician with no head and a muffler for a body!

In America, we usually sit down to Saturday dinner at about 5 pm. In Spain, the restaurants don't' even open until 9; 11 pm is typically a good dinner hour. And so it goes with robot shows. If you plan on going to bed at 9 pm, you're going to miss all the fun! CampusBot was a week-long party, but was generally deserted until about 2 pm. As the sun started its return journey across the sky, builders slowly showed up, but the crowd wouldn't get solid until about 8 or 9. And then, it went all night! No overhead lamps were needed — all the light the participants needed was provided by their own computers. It's a magical site to see several thousand computer monitors lighting up a hall. (Total attendance of participants for CampusParty was 5,700! Although only about 5-10% of that were robot builders, the rest were programmers, gamers, and hackers.)

Another difference in American cooking and Spanish cooking is simmer time. While many of the robots there were built over the course of years and were ready to go on arrival, many of the robots were made at the event. Not because the builders procrastinated, but because that was the point. Most builders brought a big box of parts and saw what they could come up with over the course of the week. A whole competition was based on what you could whip up in five days at the show.

My favorite built-on-site project was "Spanish Tetsujin." No, you didn't have to lift weight — you had to go blind! Much like Luke Skywalker putting on the blast shield for light saber training, a group of builders made a paper mache helmet that completely covered the face of the wearer. But inside the helmet were speakers, connected to a sonar array. Attendees got to put on the helmet and learn what it's like to be a robot. You had to navigate your way out of a maze using only the sonic feedback of a salvaged PC speaker. If you've never tried to navigate by sonar, let me tell you — robots have it hard! It's far more difficult than you'd imagine. I've promised to be far more kind to all my bots from now on.

One flavor remained the same in Spain as it does in America — sportsmanship and cooperation. If any builder had a problem, 10 others immediately showed up to lend a hand. That's the thing that always sticks in my mind about the robot community — no matter what country I'm in, no matter what kind of robot event it is, and no matter how old the participants are, the camaraderie and positive attitude always remains the same. And that's the best spice of all! SV

IN PART 1, WE BUILT THE BASE FOR FACEWALKER. I PROVIDED YOU WITH A SIMPLE PROGRAM THAT ALLOWED YOU TO TEST THE ASSEMBLY. ONE OF THE PROBLEMS YOU HAVE WITH A WALKER OF THIS NATURE IS THAT YOU NEED A GREAT DEAL OF CONTROL POINTS TO REALLY PUSH THE ROBOT TO ITS LIMITS.

by Michael Simpson

FACEWALKER

Part 2 — The Wireless Connection

I originally attempted to use an RC radio to control the FaceWalker, but this approach presented multiple problems:

1. Ground-based radios with more than two channels are very expensive.

2. They are slow to interface as you must poll each channel individually.

3. The transmitters are not very compact.

4. The transmitter's battery requirements can be greater than the device you are trying to control.

While I was researching other options, I came across the PS2 controller. I have never owned a PS2 so it never occurred to me that this would be a viable option.

The PS2 controller has two full position analog joy sticks, and 14 additional buttons all within reach of your fingers while you are operating the joysticks.

You can pick up a wired controller for as little as $5.95 and a wireless for $24.95. Figure 1 shows a very popular wireless model called the Predator by Pelican.

The Predator runs on two

FIGURE 1

FIGURE 2

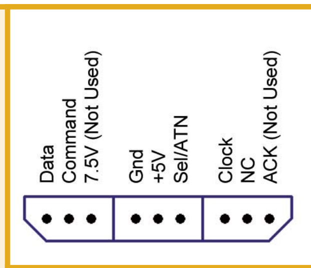


FIGURE 3

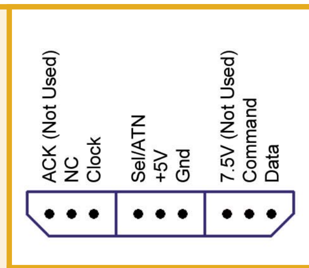


FIGURE 4

AA batteries which will last several hours. The wireless controllers operate on the 2.4 GHz band. You lock a transmitter to the receiver so that you can operate multiple transmitters at the same time. I have a WiFi wireless network and Bluetooth and never had any problems using the Predator. While these wireless controllers are rated at 30 feet, I have used mine at ranges well over 100 feet outdoors.

The receiver module shown in Figure 2 is small and operates on 5V so it is well suited for use with microcontrollers.

Figure 3 shows a pin diagram of a PS2 controller. The only connections needed are the Gnd, +5 for power, and the Data, Command, ATN, and Clock for communications.

You may connect directly to these pins, if you like. I prefer using a PS2 extension cable that has been cut in half. The extension cable will cost you $5-$10 and will give you a cool way of connecting various controllers to your application. The pins on the extension cable are reversed for obvious reasons, as shown in Figure 4.

The PS2 interface uses a SPI interface. There are four signal pins, as well as Gnd and power needed for this interface. The SEL and CMD signals are used to place the controller into the correct mode. The Data signal lead is used to send or receive data, depending upon the command. The Clock signal is used to clock the bits in or out.

To interface to a PS2 controller, you need to use a microcontroller. For the FaceWalker, we will be using a DiosPro microcontroller. I am not going to go into the protocol details as they are outside the scope of this article. I created two DiosPro functions called PS2Init and PS2Read. These functions will take care of handling the various timing requirements of different controller types, as well as the protocol needed for the actual hardware interface.

Interface Construction

Before we get started with the controller interface assembly, let's take a quick look at Schematic 1. We will be using a DiosPro chip to communicate with the PS2 controller. To make the construction as easy as possible, I used a Dios Carrier 1 board. A couple of LEDs were added for status. The remaining connections are connected to the SSC32 board's RS232 interface and the SSC32's regulated side of the logic power.

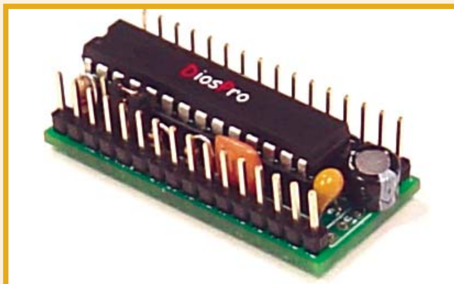
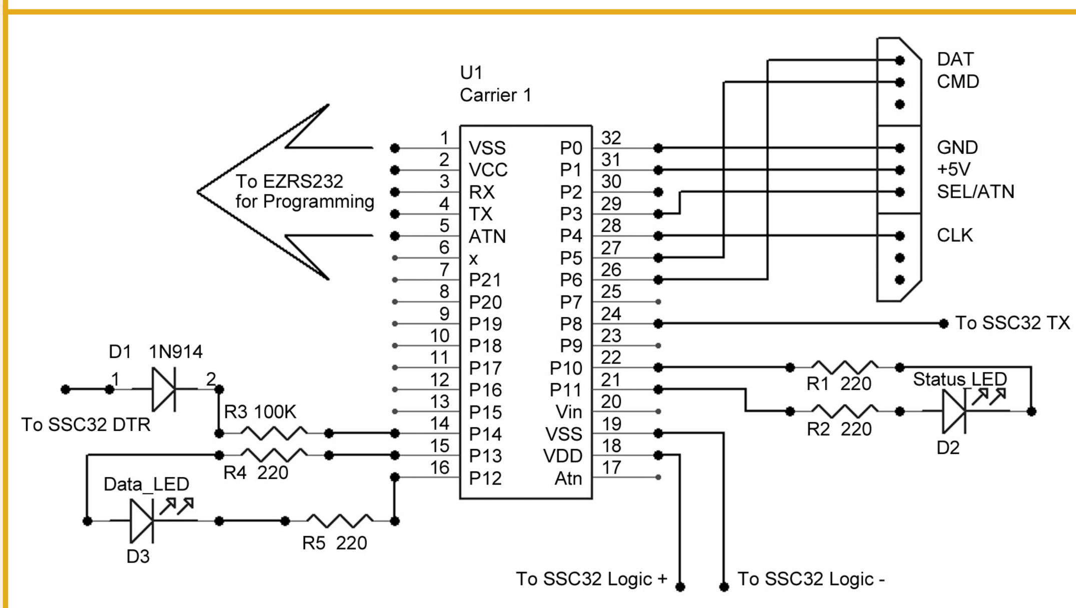


FIGURE 5

Let's start by assembling the Dios Carrier 1 board. The kit comes with an assembly manual which we will follow until we get to Step 7. In Step 7, install the headers on the top of the board as shown in Figure 5.

Once you have the carrier built, install the DiosPro chip and attach a small piece of foam tape to the bottom of the board. Attach the board to the EH3-R base, as

SCHEMATIC 1



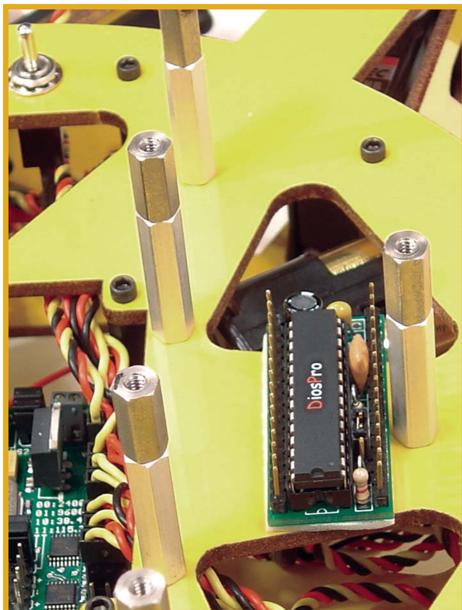


FIGURE 6

FIGURE 7

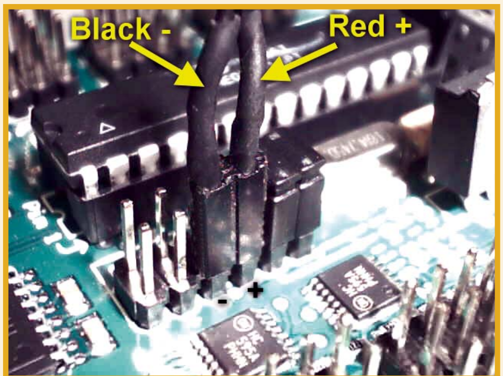


FIGURE 8

shown in Figure 6.

Center the carrier against the standoff. For orientation, notice the switches in the upper left side of Figure 6. Notice also that the notch in the chip is facing down.

In order to supply power to the Dios, we need to build the small two-conductor connector shown in Figure 7. Start by cutting two 5" wires. Since this is going to be a power connector, you need to make each conductor a different color. In this case, I used red and black. Next, cut two two-pin sections from the 36-pin female header. Cut four pieces of 1/16" heat shrink 1/2" long.

Start assembling the connector by soldering a wire to each of the pins on one of the two-pin headers. Slip a heat shrink section onto the wire and slide it over the solder joint and heat. Now slip one of the two remaining heat shrink sections over each wire. Then solder the wires to the remaining two-pin header. Slide the heat shrink down over the joints and apply heat.

Connect one end of the connector to the plus and negative header pins on the SSC32 shown in Figure 8. This header is located next to the baud jumpers. Connect the other end of the connector to the plus and negative header pins on the Dios Carrier 1 as shown in Figure 9.

In order to create the status indicators, you will need the components shown in Figure 10.

- Red LED
- Green LED
- Four 220 ohm resistors
- Four female plugs
- Two 3" pieces of 1/8" heat shrink

Take the red LED and clip the leads so that they are 3/8" long. Clip resister leads to about 1/4" as shown in Figure 11. Solder one end of the resistors into the open end of a female plug as shown. Solder the other end to the LED as shown in Figure 12.

Cut just enough of the 1/8" heat shrink to cover the complete lead and slip it over the complete plug/resistor assembly and heat as shown in Figure 13. Repeat the same procedures for the green LED.

Insert the green status LED onto the Port 10 and Port 11 header pins as shown in Figure 14. Make sure the flat side of the LED is facing left. Insert the red data LED onto the Port 13 and Port 14 header pins as shown in Figure 15. Make sure the flat side of the LED is facing right.

FIGURE 9

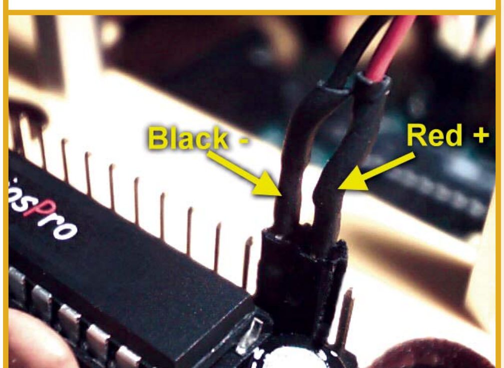


FIGURE 10

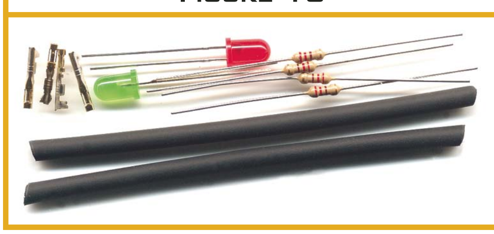


FIGURE 11

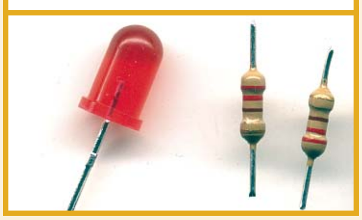
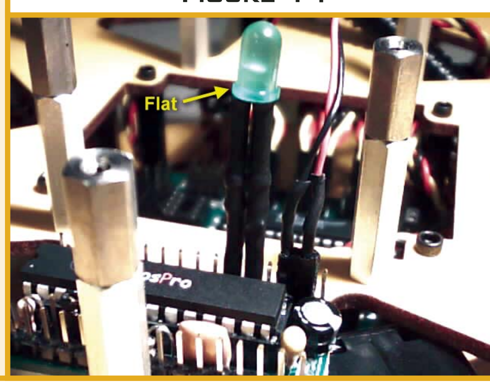


FIGURE 12

FIGURE 13

FIGURE 14



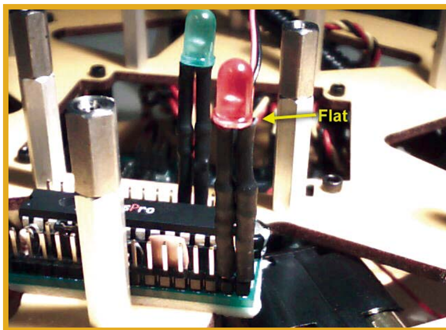


FIGURE 15

FIGURE 16

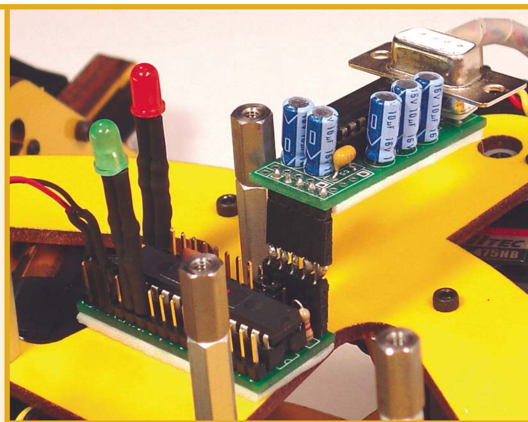


FIGURE 17

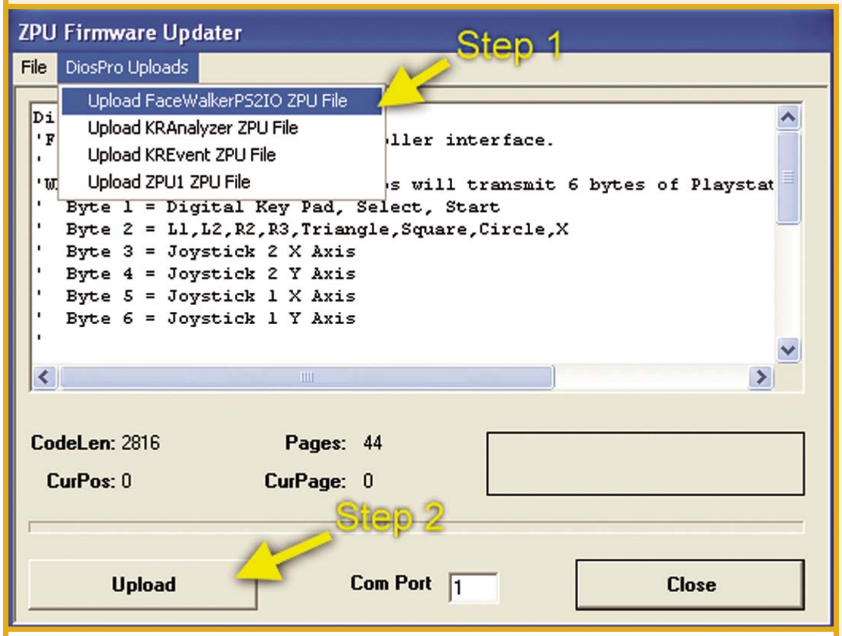


FIGURE 18

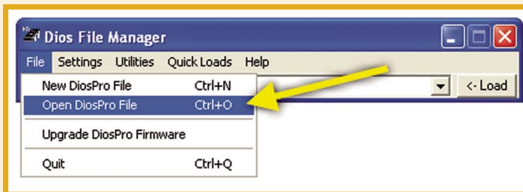


FIGURE 19

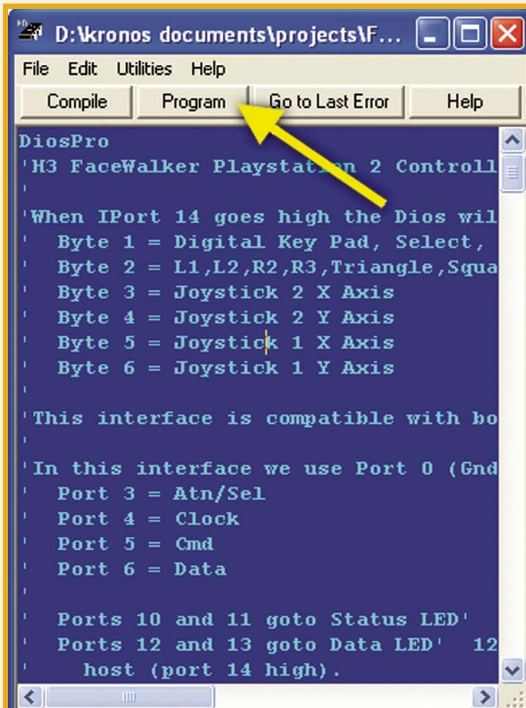


FIGURE 20

The Dios comes programmed with a test program that toggles all the I/O ports. If everything is connected properly and we provide power to the SSC32, the status LEDs should begin to flash.

Once we have a powered Dios, we can program the PS2 controller firmware into the chip. In order to do this, we need to create a small adapter by cutting two five-pin sections from the 36-pin female header and solder them together as shown in Figure 16.

With the small adapter, we can plug an EZRS232 board into the Dios carrier as shown in Figure 17. This will allow us to upload the firmware to the chip. There are a couple of ways to do this. You could download the free Dios software from the Kronos Robotics website or you can use the ZPU Firmware Updater form that is part of ZeusPro.

If you plan on making modifications to the firmware, you should use the Dios software as this will give you access to the source code. If you decide to use the

FIGURE 21

FIGURE 22

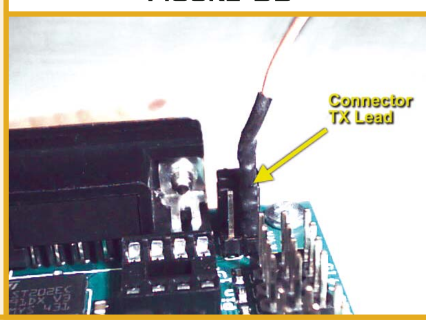
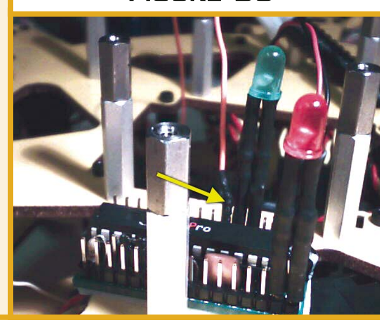


FIGURE 23



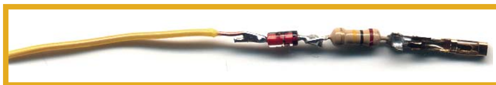


FIGURE 24

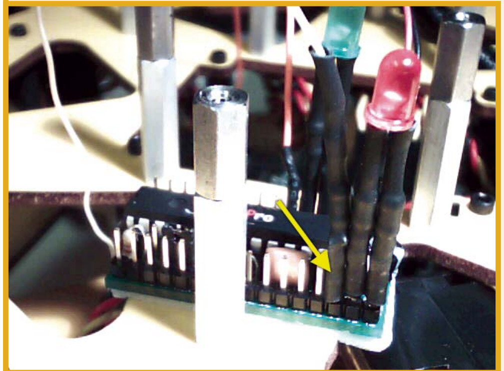


FIGURE 25

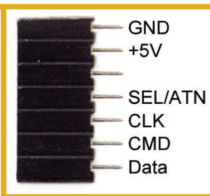


FIGURE 26

FIGURE 27

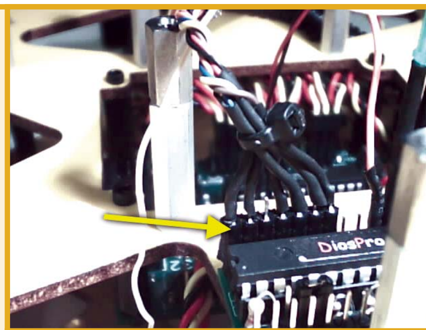


FIGURE 28

ZeusPro method, the Upload menu lists the file as FaceWalkerPS2IO. You will need to power up the SSC32 which, in turn, will power the Dios carrier. Connect a nine-pin cable between the EZRS232 board and your PC, and program.

Open the ZPU FirmWare Updater form and select Upload FaceWalkerPS2IO ZPU file as shown in Figure 18. The file will load and present you with some basic information about the file. Click the Upload button also shown in Figure 18.

If you want to program the Dios directly, you will need to load the Dios Software (free from the Kronos Robotics website) and select the Open DiosPro File option from the Dios File Manager, as shown in Figure 19. You will be presented with a file dialog. Point to the location where you placed the downloaded FaceWalker support files and load the FaceWalker PS2IO.txt file. A window will open up with the source file. At this point, you can make any changes or just program the code into the chip by hitting the Program button shown in Figure 20.

It is important to note that any program created for the DiosPro chips can be written to a ZPU file so that they may be uploaded with ZeusPRo software, as well. By simply placing the ZPU file in the Zeus firmware directory, they will show up on the menu.

Now that the chip is programmed, you should have noticed that the LEDs have started to blink. This is because they have not detected a controller and the firmware is in an error state.

It is time for you to make the final connections so that you can communicate with the firmware you just loaded into the chip. Let's start by creating a small jumper. You will need a piece of hookup wire about 6" long, two female plugs, and two pieces of 1/6" heat shrink about 3/4" long. Solder the wire to the open end of the female plugs, then slip the heat shrink over the ends, as shown in Figure 21.

If you have not done so already, remove the TX shunt (jumper) from the SSC32 board and slip one end of the jumper cable you just made onto the header pin closest to the edge of the board, as shown in Figure 22. Slip the other end of this jumper cable onto the header pin next to the green LED on the Dios Carrier, as shown in Figure 23. This is Port 9 on the Dios. What this does is connect the Dios TX UART to the TTL side of the SSC32 RS232 driver.

Now we need to connect the DTR on the SSC32 to the Dios. Back in Part 1 of this series, I had you connect a wire to the back of the board. This is DTR as it comes directly from the host. Since this is a raw RS232 level signal, we have to condition it a bit.

Connect a 100K resistor to the open end of a female plug, then connect the cathode side (band) of a 1N914 diode to the other end of the resistor. Now connect the free end of your DTR wire to the anode side of the diode.

If this was as confusing for you to read as it was for me to write, then you had better take a look at Figure 24. Cut off enough 1/8" heat shrink to cover the female plug and all the solder connections and heat.

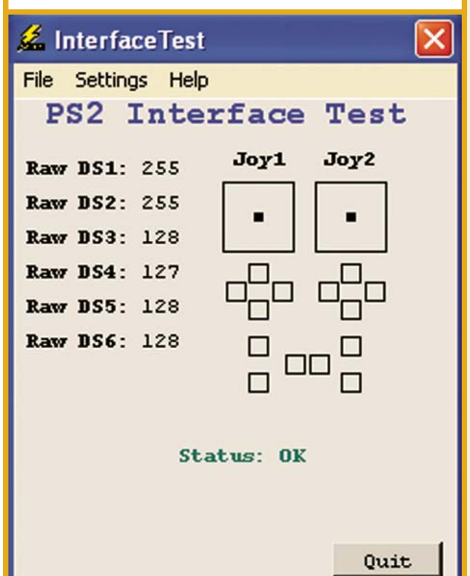
Plug this connector onto the header next to the red LED, as shown in Figure 25. This DTR signal connected to the Dios gives us a way to signal the Dios that we want a new set of PC2 controller values.

We have come a long way, but it is time for our final connection. For this, you will need a PS2 controller extension cable, some heat shrink, and a seven-pin female header.

- Cut the end of the extension cable that mates to your PS2 controller. You want about 12" of cable. Don't confuse this with the end that plugs into the PS2.

- Strip about 2" of the plastic

FIGURE 29



insulator back revealing the small colored wires.

• Trim about 1/8" off each of these wires.

• Take a multi-meter and, using Figure 4, determine which wire is mated to each pin.

• Using Figure 26, connect the various wires to the header. Slip a 1" piece of 1/6" heat shrink over each wire before you solder it in place. Once a wire is soldered, pull the heat shrink over the connection and heat.

When all the wires have been connected, you should end up with the cable assembly shown in Figure 27. I secured the ends with a small tie wrap. Now you can connect the female header to the header pins on the Dios carrier, as shown in Figure 28. Notice that the first two pins are the Gnd and +5V. These mate up with Ports 0 and 1 on the carrier. Port 0 is indicated with the yellow arrow in Figure 28.

Testing

Plug in a controller to the extension cable. Attach a serial cable from your PC to the SSC32 board. *Do not apply power to the servos at this time.* I have included a test program that talks to the Dios chip.

The InterfaceTest program shown in Figure 29 will allow you to test the two joysticks and all the buttons. Start up the InterfaceTest program and turn on the Logic Power to the SSC32.

When the power is applied, you should see the LEDs flash, then the green status LED should light solid. The red LED will flash on each time the program asks for a set of data. If you don't get the Status OK label, then it's time to start troubleshooting all your connections.

What's Next?

In the next and final article of this series, I will show you how to mount a Pocket PC and interface it to your FaceWalker. We will add some final touches to the construction. I will go into detail about the software so you can customize it to meet your own needs.

All the source code, as well as project updates, can be downloaded from the Kronos Robotics website at **www.kronosrobotics.com/Projects/FaceWalker.shtml** SV

PARTS LIST

Parts are available from Kronos Robotics

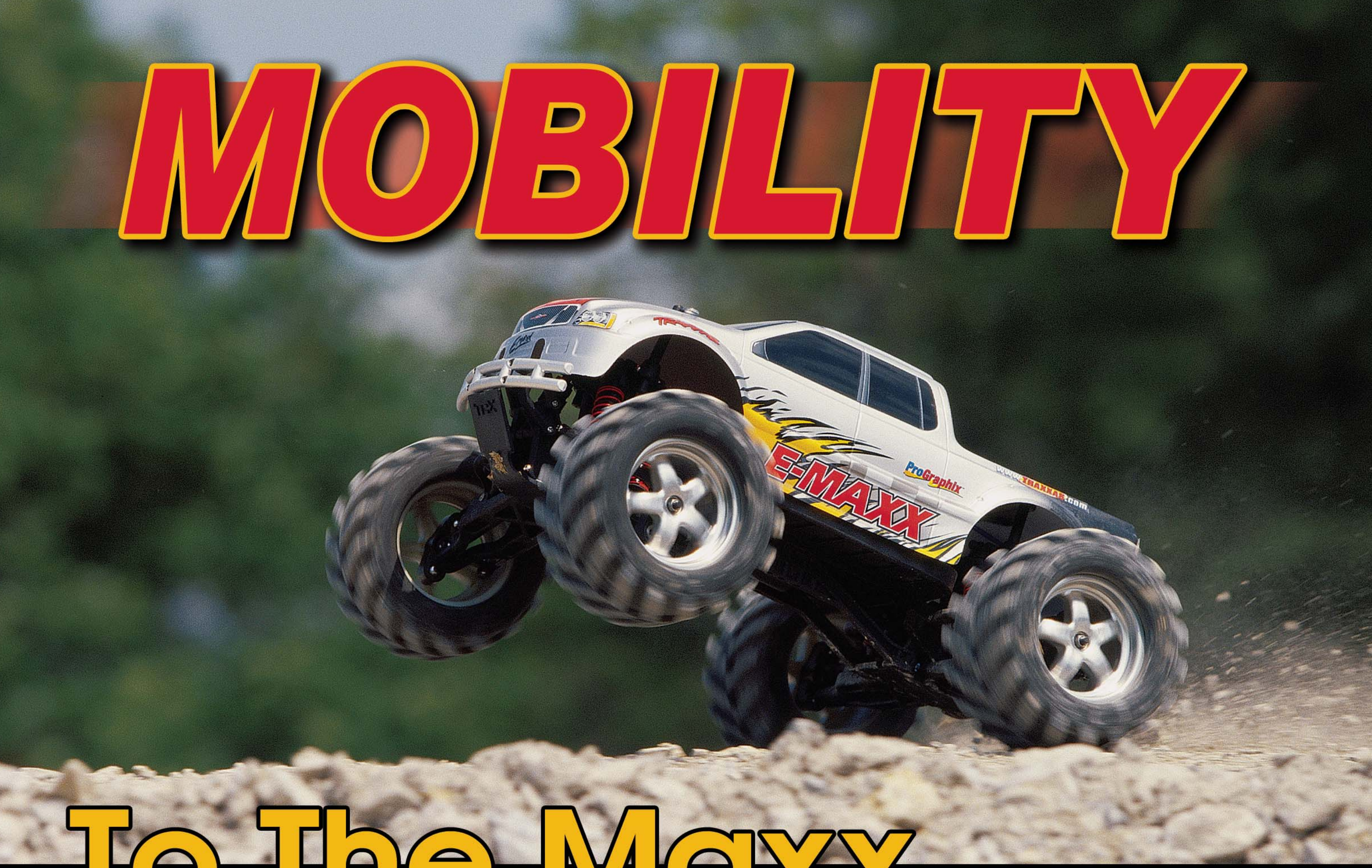
Item	Part No.
• Dios Carrier 1	#16170
• Dios Pro 28	#16429
• EZRs232	#16167
• 36-pin Female Header	#16291
• Two 3" Pieces of 1/16" Heat Shrink	#16287
• Two 3" Pieces of 1/8" Heat Shrink	#16288
• 10 Female Plugs	#16261
• Red LED	#16234
• Green LED	#16235
• 4, 220 ohm Resistors	#16188
• 100K Resistor	#16195
• 1N914 Diode	#16134
• Dios Compiler (Free Download) **www.kronosrobotics.com/downloads/DiosSetup.exe**	

Kronos Robotics — **www.kronosrobotics.com**
KRMicros — **www.krmicros.com**
ZeusPro — **www.krmicros.com/Development/ZeusPro/ZeusPro.htm**

Miscellaneous

• Double stick foam tape. Two 1" pieces should do it. Any department store or home center will carry this. A popular brand name is 3M.
• Pelican Predator Wireless PS2 Controller.
• Hookup Wire 22-26 Ga. Different colors will be helpful.

MOBILITY



To The Maxx

PART 6 — Overcoming Obstacles

by Chris Cooper

For more information on this product, visit **www.machinebus.com/emaxx**

Last month, we discussed an overview of the Global Positioning System (GPS) and implemented basic GPS navigation. We examined how NMEA messages output from the GPS receiver could be used for vehicle navigation. The E-Maxx, using a combination of the GPS receiver and the digital compass, navigated a predefined route of waypoints programmed into the receiver.

Up until now, we have been careful to test drive the E-Maxx only in wide-open areas. It's a precaution taken to prevent it from careening into obstacles as it travels along its heading, oblivious to what lay ahead. When we complete this installment, obstacles will no longer be a problem.

In this article, we will describe a variety of ranging sensors and their strengths and weaknesses. We will cover sensor choices and ideal placements to provide essential information about the surrounding environment. We will add code to read the input from the sensors and use it to implement obstacle detection and avoidance. The E-Maxx will then be able to detect objects en route, change direction to avoid a collision, and get back on track towards the next waypoint.

A Range of Ranging Sensors

There are three common types of ranging sensors applicable for use on the E-Maxx, each with its own advantages and disadvantages. Refer to Table 1.

Infrared

Infrared range sensors work using triangulation. These are the sensors at restaurants that flush the toilet for you usually after you have stepped away.

Photo Above: The E-Maxx RC monster truck makes an excellent robotics base. *Photo courtesy of Traxxas.*

TABLE 1		
	Pros	**Cons**
Infrared	Low cost. Narrow focus cone.	Dark, shiny, small objects cause inconsistent readings. Nonlinear. Subject to infrared radiation (IR) interference.
Ultrasonic	Detects small objects. Not affected by color. Easy to use.	Speed of sound varies. Large focus cone. Object angles affect readings.
Laser	Highly accurate.	Expensive. Scans in a plane.

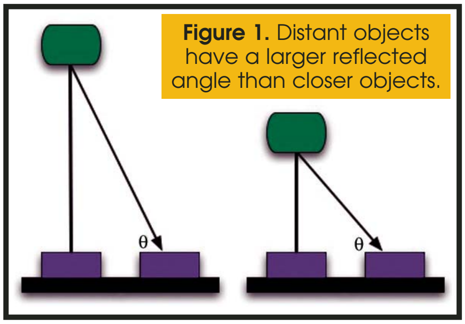


Figure 1. Distant objects have a larger reflected angle than closer objects.

The emitter sends a pulse of light in the infrared spectrum (1 mm–750 nm wavelength). The receiver contains a linear CCD array and measures the angle of the reflected beam. The distance of the object is proportional to the reflected angle.

Additionally, the frequency of the IR beam is modulated. Modulation allows the sensor to work in various lighting conditions with little or no effect from ambient light. Some factors that do affect the readings on infrared sensors are the reflectivity, shape, and size of the object. If the object absorbs the emitter beam, the object is never detected. Similarly, the shape of the object can cause the emitter beam to be reflected away from the IR receiver, and if the object is small, the narrow five-degree IR beam may not be able to detect it. Infrared has a range of about 10–80 cm.

Ultrasonic

Ultrasonic distance sensors operate at the speed of sound — 1,130 feet/second. To measure a distance, a timer triggers a piezoelectric transmitter. The piezoelectric material in the transmitter oscillates when a voltage is applied, generating ultrasonic sound vibrations. The transmitter emits a series of ultrasonic pulses. A piezoelectric receiver then detects the reflection of the pulses, converts it to a voltage, and stops the timer. The time measured is then divided by two and multiplied with the speed of sound. The result is the distance between the sensor and the object in front of it.

Since the piezoelectric effect is reversible, any transmitter can also be a receiver, and any receiver can be a transmitter. In fact, some ultrasonic distance sensors — like the Max Sonar EZ1 — reduce their size by combining the transmitter and receiver into a single piezoelectric transceiver.

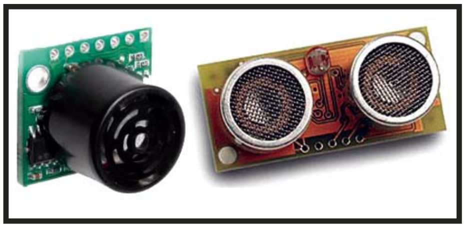


Figure 2. Ultrasonic sensors using one and two piezoelectric transducers.

Laser Range Finders

Laser range finders are sensors that measure the distance to objects by timing the reflection of an emitted laser pulse. Also known as LIDAR (Laser Imaging Detection and Ranging) sensors, the narrow beam of laser light is scanned over the surrounding area using a rotating mirror (see Figure 3). Laser

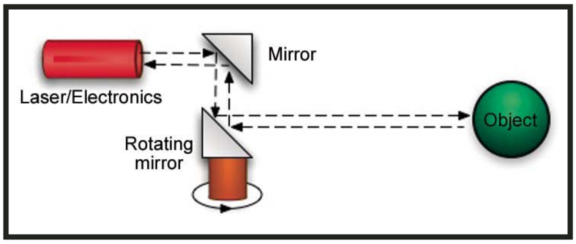


Figure 3. The laser is scanned over objects using a rotating mirror.

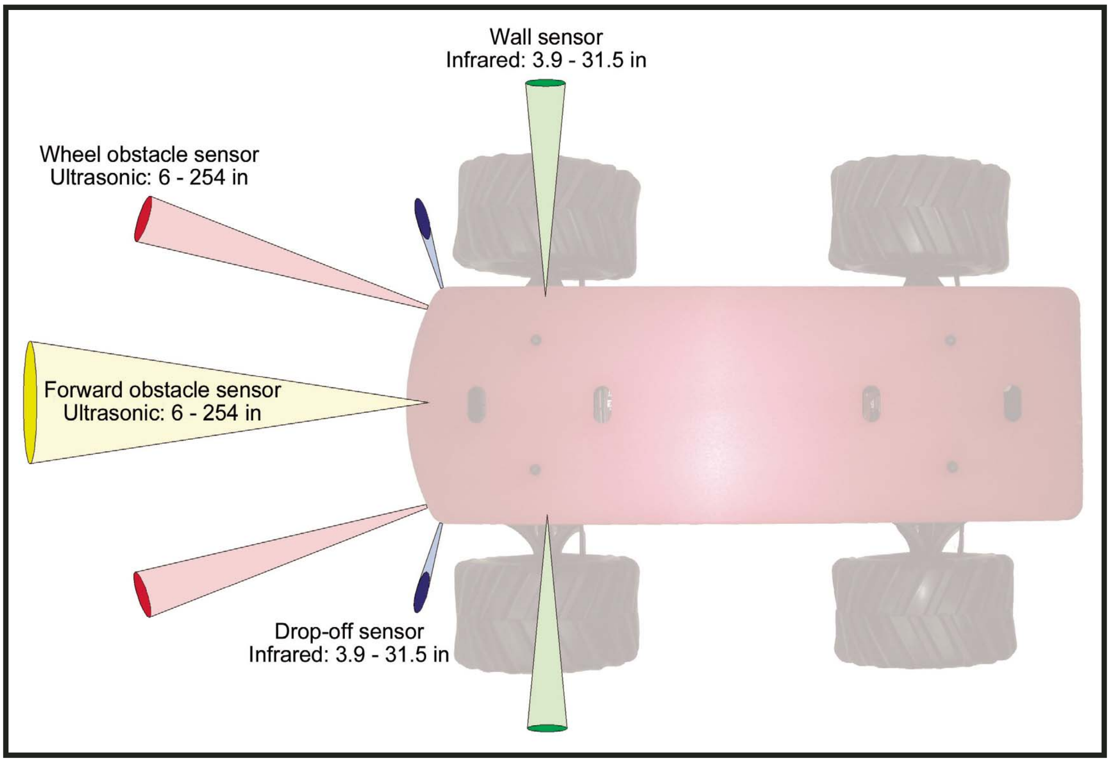


Figure 4. A configuration of ranging sensors designed to detect likely hazards.

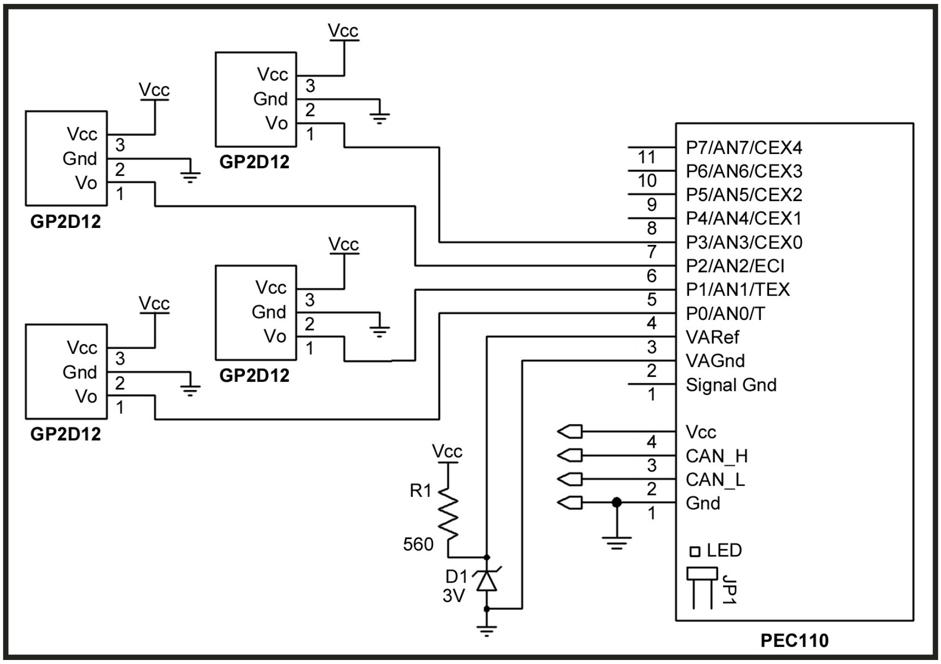


Figure 5. Schematic showing wiring of four Sharp GP2D12 sensors to a PEC-110 to create the infrared module.

range finders have a range of four to 150 meters, with a resolution finer than 50 millimeters. The large range and precision comes at a price. Laser range finders are very expensive and out of most hobbyists' price range. Amazingly, one of the entries in this year's RoboMagellan event at RoboGames used a Sick brand LIDAR unit.

Sensor Topology

While there are many ways to position and combine sensors (take a look at Bryan Bergeron's articles on sensor fusion in the July and August issues of *SERVO*), we positioned each sensor as in Figure 4 to operate alone and perform a single specific task.

Wall Sensors

We placed a GP2D12 IR sensor centered above each front wheel, parallel with the axle (perpendicular to the direction of travel). These two sensors allow the E-Maxx to sense the distance to a wall or other obstacle for avoidance or even follow alongside (hug) it.

Drop-off Sensors

A GP2D12 IR sensor placed in front of each front wheel pointing down allows us to measure drop-off for each wheel, in case we don't hit the drop-off at a perpendicular angle. The intent is to avoid drop-offs such as stairs and steep cliffs.

Wheel Obstacle Sensors

One MaxSonar EZ1 ultrasonic sensor is placed over each wheel, angled so that it hits the ground in front of the E-Maxx slightly short of its range of three feet. These give us a look ahead, so we may brake or reduce speed when necessary.

Forward Obstacle Sensor

A MaxSonar EZ1 ultrasonic sensor is placed at the front of the E-Maxx pointing directly forward to detect any obstacles directly in front of the E-Maxx. It's important to note that the effectiveness of the forward obstacle sensor is heavily dependent on speed. As the speed of the E-Maxx increases, so does its stopping distance.

Adding Sensors

The MaxSonar EZ-1 sensor returns the distance to the object in inches as one of three types of values: pulse width, serial digital, or analog. The analog value is the easiest to use. For every inch between the sensor and the detected object, the analog value increases by 10 mV. Testing the EZ-1 is as simple as supplying the sensor with 5V and measuring the difference between

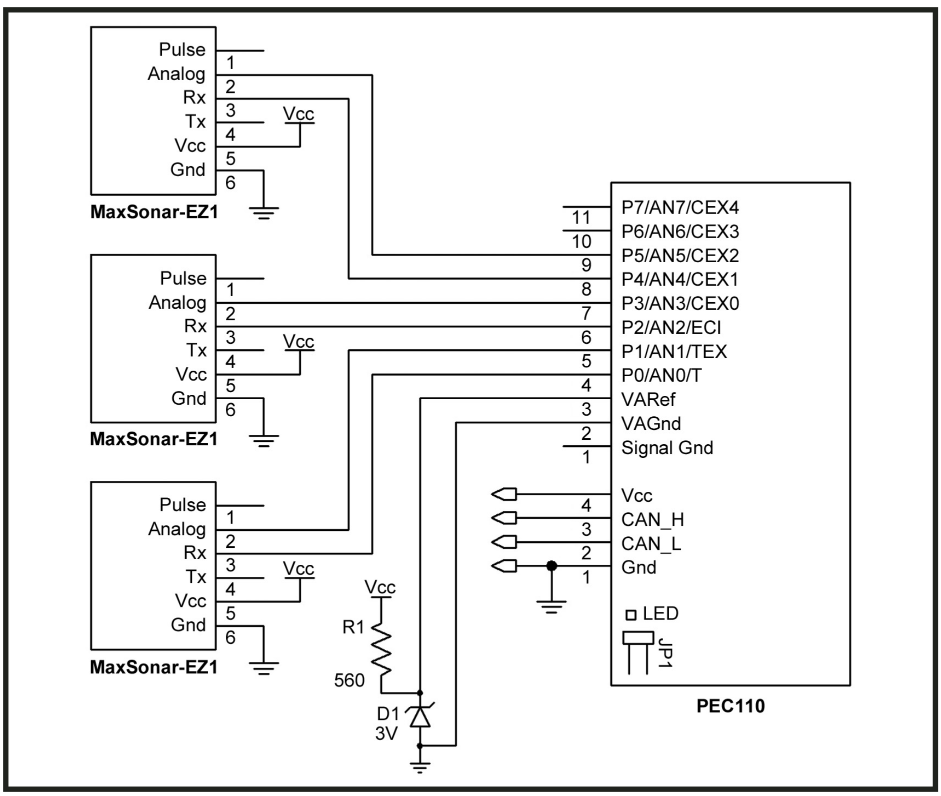


Figure 6. Diagram of three MaxSonar EZ1 sensors to a PEC-110 to create the ultrasonic module.

Analog and Gnd with a multimeter.

The GP2D12 infrared sensor is trickier because the analog voltage is not linear with respect to the object's distance. It is a logarithmic curve due to the trigonometry involved in measuring the reflectance angles. This curve is different from detector to detector so it's helpful to calibrate it and use a lookup table to map analog voltage to distance. Calibration is accomplished by placing an object in front of the sensor at known distances. At each point, use a multimeter to measure the analog value, recording the distance and analog value.

We created two new modules for the E-Maxx to handle ranging. The first module handles readings from the four GP2D12 infrared sensors (see Figure 5). The PEC-110 uses an A/D converter to read the analog values and then transmits them to the application host over its communication bus. The resistor and 3V zener diode shown in Figure 5 create the reference voltage for the A/D converter.

The second module (see Figure 6) handles readings from the three Max Sonar EZ-1 ultrasonic sensors. The same circuitry is used in the ultrasonic module as in the infrared module to supply the A/D conversion with a reference voltage. In order to avoid interference between EZ-1 sensors, each EZ-1 is turned on sequentially every 50 ms by setting its RX pin high. While this guarantees that no inter-sensor interference will occur, it does limit the frequency of readings.

Processing Sensor Output

The infrared module sends the application host the analog value and a 0, 1, 2, or 3 that identifies the sensor transmitting the reading. GP2D12 code running on the application host, in turn, uses a lookup table to find the distance corresponding to the analog value read for each GP2D12 (see Listing 1). The interface to the ultrasonic sensors is very similar, as shown in Listing 2.

Sensor information is coalesced into a range interface which incorporates each of the infrared and

LISTING 1

```
#include <stdlib.h>
#include <unistd.h>  /* UNIX standard function definitions */
#include "machineBus.h"
#ifndef GP2D12_
#define GP2D12_

    struct gp2d12_t;
    typedef struct gp2d12_t *Gp2d12;

    // Create a new  gp2d12 reference
    Gp2d12 gp2d12_createGp2d12(CommBus C, uint8_t id, uint8_t* lookup);

    // Get the distance in inches, -1 indicates no object detected
    int8_t gp2d12_getDistance(Gp2d12 G);

    // Dispose of the gp2d12
    void gp2d12_disposeRange(Gp2d12 G);

    uint8_t gp2d12_messageCallback(void* object, CAN_MESSAGE *rxMessage);

#endif /*GP2D12_*/
```

LISTING 2

```
#include <stdlib.h>
#include <unistd.h>  /* UNIX standard function definitions */
#include "machineBus.h"
#ifndef MAXSONAREZ1_
#define MAXSONAREZ1_

    struct maxsonarez1_t;
    typedef struct maxsonarez1_t *MaxSonarEz1;

    // Create a new maxsonarez1 reference
    MaxSonarEz1 maxsonarez1_createMaxSonarEz1(CommBus C, uint8_t id);

    // Get the distance in inches, -1 indicates no object detected
    int8_t maxsonarez1_getDistance(MaxSonarEz1 M);

    // Dispose of the maxsonarez1
    void maxsonarez1_disposeRange(MaxSonarEz1 M);

    uint8_t maxsonarez1_messageCallback(void* object, CAN_MESSAGE
*rxMessage);

#endif /*MAXSONAREZ1_*/
```

LISTING 3

```
done = 0;

    while (!done ) {

        /* Check Bus status */
        if (commbus_status(bus) != 0) {
            printf( "Failed to retreive status\n" );
        }
        // If the range to the last waypoint is less than dilution of precision,
        // and we are at the end waypoint
        if ((gps_getRangeToNextWaypoint(gps) < gps_getHorizontalDilutionOfPrecision(gps))
                && strcmp(gps_getNameOfNextWaypoint(gps), "END") == 0) {

            done = 1;

        } else if ( commbus_readyToTransmit(bus)) {
```

continued ...

Listing 3 continued ...

```
            // Get our current bearing
            currentBearing = compass_getBearing(compass);

            // If any obstacles are detected
            if (range_obstaclesDetected(range)) {
                // Calculate new bearing to avoid obstacles
                newBearing = range_getBearing(range, currentBearing);

            } else {
                if (gps_arrivedAtWaypoint(gps)) {
                    // New bearing, reset moving average
                    util_resetMovingAverage(movingAverage);
                }
                // Use a moving average to account for noise/errors
                newBearing = util_adjustMovingAverage(movingAverage,

gps_getMagneticBearingToNextWaypoint(gps));
            }
            // Steer the E-Maxx towards the new bearing
            steering_steer(steering, newBearing, currentBearing);
        }

    } // while
```

ultrasonic sensors, as well as their position on the E-Maxx. The range interface uses this information to perform two roles: obstacle detection and obstacle avoidance.

The method `range_obstaclesDetected()` polls each sensor to determine if its current value indicates an obstacle is present and returns true if there is an obstacle in the path of the E-Maxx. If obstacles have been detected, calling `range_getBearing(range,currentBearing)` will recommend a new bearing for the E-Maxx to follow based on which sensors detected the obstacle. If no obstacles have been detected, then navigation is deferred back to the GPS (see Listing 3).

PARTS LIST

- PEC-110: **www.machinebus.com**
- Ping))) Ultrasonic Sensor: **www.parallax.com**
- MaxSonar EZ1 Sonar Range Finder: **www.wrighthobbies.net**
- Sharp GP2D12: **www.acroname.com**

Additional Resource:

- Autonomous E-Maxx: **www.machinebus.com/emaxx**

Conclusion

Using a combination of infrared and ultrasonic ranging sensors, we added obstacle detection and programmed some basic avoidance techniques. At this point, the E-Maxx can confidently move around its environment. We are anxious to take the project further, however, our application host code has become ever more complicated, requiring a refactoring of the code.

As clean as the interface to the E-Maxx sensors and actuators is, it is surrounded by a brittle processing loop that is bowing under the weight of all the inputs. Replacing this loop with a subsumption architecture, which builds up complex abstract behaviors from layers of underlying simple ones, will give the E-Maxx the same flexibility and expandability in software that it already has in hardware.

The Autonomous E-Maxx is now able to compete in a RoboMagellan competition. However, in order to excel in the competition, it needs one last addition. In a future article, we will incorporate an AVRCam from **www.jrobot.net** to perform orange cone recognition and tracking. Members of our local robotics group (Chibots) have had excellent success with the AVRCam and our experiences with it have been positive to date. I'd like to thank everyone who helped me with this series, including the readers for their letters of support. **SV**

AUTHOR BIO

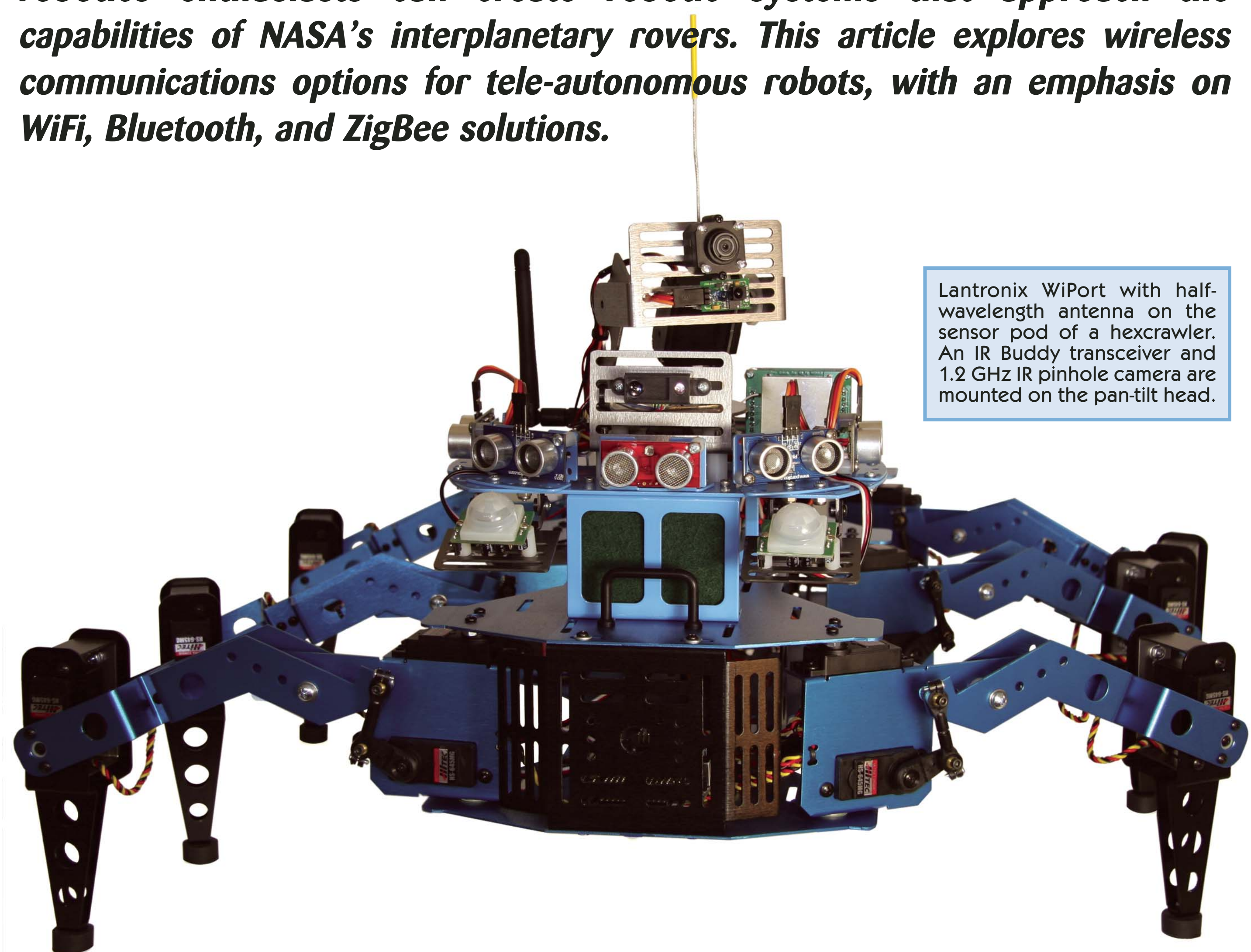
Chris Cooper is currently a software architect for Chicago-based Machine Bus Corporation. He has a B.S. in Computer Science from the University of Illinois, has presented at the OMG's Robotics SIG on Distributed Control Systems, and is a member of the Chicago Area Robotics Group (Chibots). He can be reached at cooper@coopertechnical.com

Wireless Communications for TELE-AUTONOMOUS ROBOTS

✔ by Bryan Bergeron

The Martian rovers — the latest poster children for tele-autonomous robotics — continue to demonstrate the value of extending the real-time, first-person sensory experiences of their remote operators. Not only are sensor data used as the basis for autonomous behavior, but they are processed by terrestrial computers that generate control signals that, together with operator input, direct high-level robot behaviors.

Thanks to standardized, affordable wireless communications components, robotics enthusiasts can create robotic systems that approach the capabilities of NASA's interplanetary rovers. This article explores wireless communications options for tele-autonomous robots, with an emphasis on WiFi, Bluetooth, and ZigBee solutions.



Lantronix WiPort with half-wavelength antenna on the sensor pod of a hexcrawler. An IR Buddy transceiver and 1.2 GHz IR pinhole camera are mounted on the pan-tilt head.

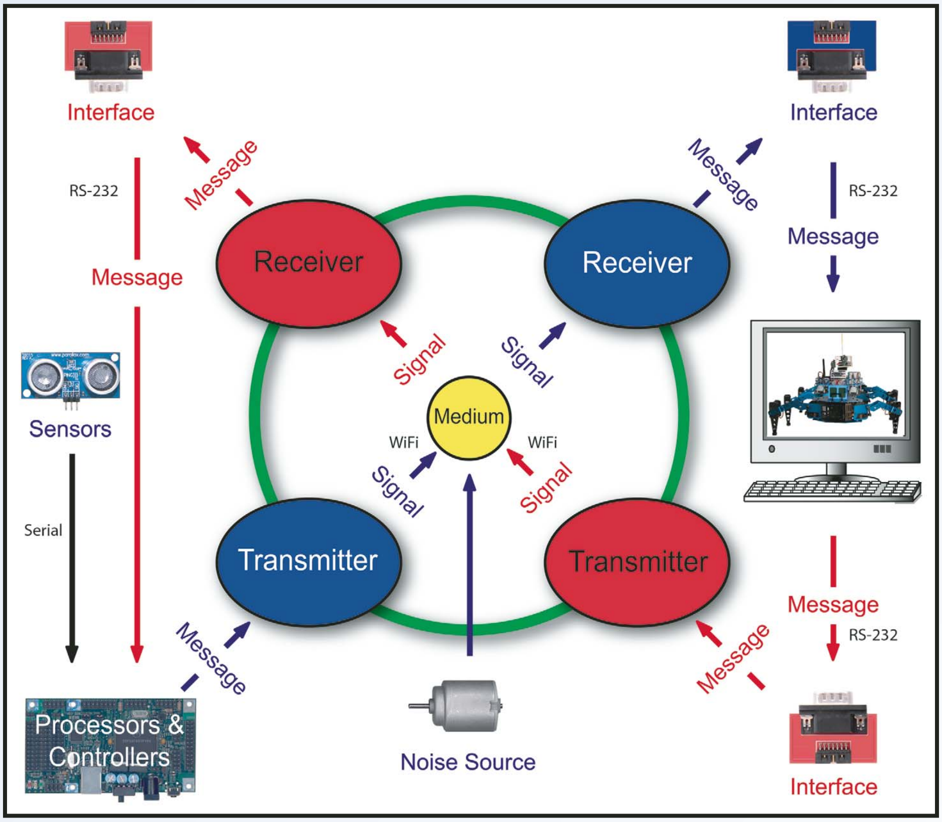


FIGURE 1. Tele-autonomous robotic architecture from a communications-channel perspective.

Sensor Data	Polling Frequency (Hz)	Required Message Rates
Accelerometer	10-100	80-800 bps per axis
Ambient Humidity	0.1	~1 bps
Ambient Light	1	8 bps
Ambient Pressure	0.1	~1 bps
Ambient Temperature	0.1	~1 bps
Battery Count	1	8 bps
Battery Voltage	1	8 bps
Direction (Compass)	10	80 bps
Inclination (Inclinometer)	10	80 bps per axis
Joint Torque	10	80 bps per joint
Range (IR Rangefinder)	10-100	80-800 bps per sensor
Monaural Audio	8 KHz	64 kbps
Motion (PIR)	100	800 bps per sensor
Distance (Shaft Encoder)	100	800 bps per sensor
Touch (Bumper Switches)	100	800 bps per sensor
Range (US Rangefinder)	100	800 bps per sensor
Servo Position	10	800 bps per servo
Video	1-30	1.2 kbps to 34 Mbps
Motor Temperature	0.1	~1 bps
Battery Temperature	0.1	~1 bps

TABLE 1. Nominal polling frequency and associated message data rates, assuming local processing of the raw sensor data and eight-bit encoding.

Robot Communications

Tele-autonomous robots are typically enabled by an asymmetrical, closed-loop communications channel. The communications are asymmetrical to the extent that robots generate more data than they receive. The channel is closed-loop in that sensors provide the operator and automated control system with feedback on the results of computer- and operator-generated commands.

Architecture

Figure 1 illustrates one of several possible architectures for the communications channel of a tele-autonomous robot. The communications channel is composed of a pair of matched transmitter and receiver units, a PC at the operator site, and a mobile robot with an on-board processor. Although the focus of this article is on the area inscribed by the green ring, walking through the entire channel can highlight several important characteristics of tele-autonomous robotic communications.

Starting at the far left of the figure, an onboard processor collects sensor data that are either processed or simply stored, and then fed to the transmitter. Within the transmitter, the message is used to vary the amplitude, frequency, and/or phase of a radio frequency (RF) carrier. The RF signal is sent to an antenna, which radiates the signal through the air and various structures en route to the receiver.

The RF signal is compromised by noise, signified by the DC motor near the bottom of the figure. In reality, there are multiple noise sources, and each negatively impacts communications. Noise may originate from poor power supply regulation, arcing within DC motors, servos, and other onboard electronics. External noise sources include atmospheric disturbances, absorption, and other devices sharing the same RF spectrum.

The effect of a specific noise

source depends on the signal frequency. For example, a 2.4 GHz signal is attenuated by water, whether in the atmosphere or a human body. As such, spectators gathered around a robot with a 2.4 GHz communications link will attenuate both transmitted and received signals. Concrete and metal have a much greater attenuation effect on signal level.

Some degradation of the RF signal is unavoidable. For example, there is no escaping the decrease in signal-to-noise ratio due to square-law signal attenuation. Doubling the transmitter-to-receiver distance decreases signal strength to $1/(2^2)$ or 1/4 of the original value. RF signals are also subject to the effects of refraction, reflection, and diffraction.

Demodulation of the RF signal in the receiver recovers the original message, which, in this example, is passed to a PC via an RS232 interface. Programs executing on the PC perform operations such as sensor-data fusion, image recognition, and path planning, both autonomously and under operator supervision.

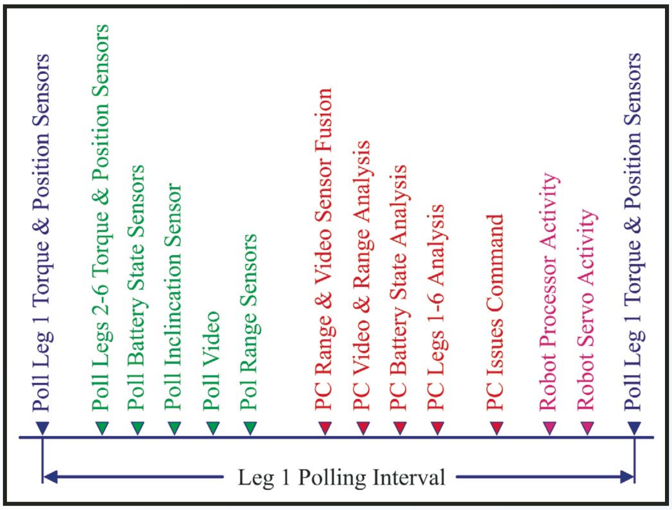
Based on summary statistics and graphics of sensor data, the operator may either allow the robot to continue autonomous operation, or issue commands to the robot. After a command or message is fed to a transmitter via a PC interface, the signal travels from transmitter to receiver, where it is demodulated. The resulting message is fed to the onboard processor, which directs motion control, energy management, and other operational behaviors.

Bandwidth

As with battery power, a robot can always use more communications bandwidth. However, it's important to know just how much bandwidth is acceptable, because the price and complexity of the communications options are usually a function of the bandwidth they support.

Table 1 lists some nominal message rates for typical robot sensors and capabilities. The polling frequency assumes a wheeled, indoor robot capable of traveling at up to a foot per second. A robot based on an e-Maxx frame and intended for outdoor use at 20-30 mph would require at least ten times the listed polling frequencies for motion, acceleration, inclination, direction, and touch. Similarly, anything less than 30 fps video would be unacceptable. The bandwidth requirement will also increase if the platform is changed from a wheeled vehicle to a crawler. Monitoring position torque for each joint of a three DOF hexapod at 10 Hz can consume at least18 x 2 x 80 bps or 2.9 kbps.

FIGURE 2. Relative event timing for polling a hexapod leg joint for position and torque.



The obvious bandwidth hog is video, with a requirement from just over 1 kbps to 34 Mbps and beyond. The bandwidth required for video depends on image resolution, whether the image is color or monochrome, frame rate, and type of compression. The range listed in Table 1 represents the range from postage stamp-sized images with a frame rate of one image every 30 or 40 seconds to 30 fps NTSC broadcast quality video.

The figures listed in Table 1 assume simple monitoring of individual sensors and don't reflect the overhead imposed by interdependencies, timing issues, an emphasis on advanced PC-based behavior control over autonomous behavior, or an imperfect communications channel. Consider the case where the remote PC can assume full motion control of a 3-DOF hexcrawler equipped with 18 torque and position sensors, battery voltage and current sensors, inclinometer, and six range sensors.

Focusing on one leg of the hexcrawler, assume the torque and position are polled, the data are encoded by the onboard processor, and transmitted to the PC. The message — consisting of six eight-bit values — is received and fed to the PC. The motion-control algorithm executing on the PC considers the torque and position values, as well as the latest data on battery temperature, battery current, inclination, range, and the last video frame to determine that the leg is snagged on an obstacle. The algorithm determines that the most appropriate behavior is to stop all forward movement, move the leg back and up to free the leg, and then reassess the situation.

Assuming a multi-core, multi-GHz PC, and an efficient motion-control algorithm, the first command — to stop all forward movement — is issued 1/1000th of a second after receipt of the leg data. Updated torque and position data are received by the PC, and the process repeats.

The relative timing of events is illustrated in Figure 2. Note the need for the PC to poll the other legs, which could also be snagged on some obstacle, as well as the provision for sensor fusion and other PC-based processing. Given the PC control requirements, the nominal polling frequencies and message data rates listed in Table 1 become inadequate if the hexapod is expected to move faster than a few feet per hour. Furthermore, the bandwidth required for communicating control codes from the PC, communications errors, and PC computational abilities — which have been ignored to this point — suddenly become relevant.

Assuming the sensors and processor aboard the robot can keep up with demand, a reasonable minimal polling frequency for Leg 1's torque and position sensors is 100 Hz. Given the PC's need for up-to-date battery state and inclination data, inclinometer,

	WiPort	Eb500	XBee-Pro
Protocol	WiFi 802.11b	Bluetooth	ZigBee 802.15.4
Weight	29 g	13 g	4 g
Size	34 x 33 x 11 mm	70 x 41 x 9 mm	33 x 24 x 3 mm
Voltage	3.3 VDC	5-12 VDC	3.3 VDC
Current Xmt/Rcv	480/170 mA	35/8 mA	270/55 mA
Range Outdoors	100+ m	10-100 m	1.6 km
Message Throughput (Max)	921.6 kbps x 2	230 kbps	115.2 kbps
RF Signal Data Rate	11/5.5/2/1 Mbps	1 Mbps	250 kbps
Frequency	2.4 GHz	2.4 GHz	2.4 GHz
Modulation	DSSS	FHSS	DSSS
Antenna Options	Remote	Surface Mount	Surface Mount, onboard, or remote
Connectivity	TC/PIP, Port Redirector	Parallax AppMod	Virtual Com Port
Latency	Moderate	Low	Low
Configuration Required	Yes	No	No
Cost — Robot Side	$125	$100	$35
Cost — PC Side	$100	$60	$110
Security	Mod	High	High
Ease of Use	Mod	High	High

TABLE 2. Comparison of three wireless communications options for tele-autonomous robots.

battery current and voltage, and rangefinder sensors should be polled at 100 Hz, as well. Unless the sensor data are encoded and compressed, the raw message bandwidth requirements approach 24 kbps, excluding video. This figure assumes timing is perfect, and that data are available as soon as they are requested. A reasonable heuristic is to provide communications bandwidth capacity at least double the calculated minimum.

Message Rate vs. RF Rate

In evaluating the specifications on wireless communications options, it's important to differentiate message or data rate from RF signal rate. WiFi provides a maximum RF rate of 11 Mbps, but the degree to which this bandwidth is exploited depends on the rate data are produced by sensors, the processing power of the onboard processors, the bandwidth of the level change interfaces, and the design of the WiFi device.

For example, a robot based on the popular BS2p has a maximum message throughput of 920 kbps, assuming the microprocessor is used solely for simplex communications. The BS2p, like many microcontrollers, can't receive while transmitting, and vice versa. Furthermore, it can't process sensor data if it's sending or receiving serial data. As a result, a BSP2 used for collecting sensor data, handling motion control, and adjusting timing so that the robot can act on commands from the PC might have a message throughput of less than 5 kbps.

The Parallax Propeller Chip, the New Micros ServoPod, and any of several single-board computers running real-time Linux are obvious candidates for high-throughput, tele-autonomous robot controllers. However, adding more local intelligence is necessary but insufficient to guarantee optimal message throughput. Communications protocols such as WiFi insure that whatever digital data are sent to the transmitter are faithfully recovered by the receiver, despite noise, signal attenuation, and other sources of error. However, robust sensor encoding schemes are required for maximum message throughput.

Wireless Options

Assume that our goal is to develop a tele-autonomous robot capable of monitoring onboard temperature, energy status, and expenditure, as well as environmental conditions, and can autonomously regulate energy expenditure. The exact specifications — and cost — depend on the components selected for the robot, including the communications hardware and software. Table 2 illustrates the price, throughput, range, size, weight, and power requirements for three standards-based communications options — the WiPort, eb500, and XBee-Pro — which are based on the WiFi (IEEE 802.11b), Bluetooth, and ZigBee (IEEE 802.15.4) standards, respectively.

WiPort

The Lantronix WiPort is clearly the most sophisticated of the three devices considered here. The compact module offers dual serial ports that can each handle messages at up to 921 kbps, an embedded web server, and a built-in web interface for configuring the device. The WiPort module provides 2 MB of Flash memory — enough for about 50 web pages.

The WiPort is available separately, or, as pictured in Figure 3, as part of an evaluation board. The board provides full access to the WiPort's ports, a wired Ethernet connection, a mount for the antenna, a regulated power supply, two nine-pin serial connectors — one for RS232 and one for RS232/422/485 — and diagnostic LEDs.

Readers familiar with setting up a wireless web access point will feel at home with the WiPort. For everyone else, the setup requires some study. The first hurdle is determining whether the WiPort's IP address will be automatically assigned or fixed. If the former, you'll have to determine the subnet mask and gateway used by your PC by typing "ipconfig/all" at the command prompt.

Assigning a fixed IP address to the

WiPort involves running the DeviceInstaller application with the WiPort connected to the PC via a null modem cable or wirelessly through a web browser. I found the browser approach erratic — sometimes the WiPort was available, and other times it would drop the connection completely. I resorted to using a null modem cable and HyperTerm to set the IP address.

Even with a secure connection, the numerous web pages of variables and options are daunting to the uninitiated. There are configuration screens for options ranging from security, network name, type, and channel, to encryption keys and the WiPort pin settings. The Lantronix software simplifies the setup process — but you still have to read the documentation to determine what's relevant, and the consequences of accepting the default settings. I had good luck setting up the WiPort as part of an ad hoc network for point-to-point communications with my WiFi-enabled laptop.

The WiPort RF signal is modulated using a Direct Sequence Spread Spectrum technique in which multiple channels are used simultaneously. This is relevant because when one channel is blocked, the others continue, with a stair-step decrease in RF signal throughput. As noted in Table 2, the WiPort automatically adjusts RF signal rate from 11 to 1 Mbps as a function of channel availability.

The WiPort module requires a hardware interface to convert the 3.3 V CMOS logic level to RS232. The SN75C3223 multichannel RS232 line driver/receiver is an obvious candidate, with 1 Mbps throughput per channel. Connecting the line driver/receiver to the WiPort is the most challenging part of the exercise, given that the unit uses a recessed, 2 x 20 pin, 1 mm micro header.

The logical connection between the WiPort and PC is straightforward, thanks to the Windows-based Com Port Redirector software that accompanies the WiPort. With the Com Port Redirector, accessing a serial device connected to the WiPort is as easy as writing to the assigned virtual port.

Compared with the other two devices considered here, the WiPort is the most difficult to configure. This is due to the complexity — and power — of WiFi, not the Lantronix implementation. The WiPort provides enough bandwidth for multiple users to access robot data, such as in a classroom environment. Using the ubiquitous WiFi protocol also opens options, such as the ability to collect sensor data and control a robot from any computer on the Internet.

Eb500

The eb500 — which is developed by A7 Engineering and marketed by Parallax — provides a gentle introduction to Bluetooth-based tele-autonomous robotics (see Figure 3). Bluetooth operates on the same 2.4 GHz band as WiFi, but is significantly different in terms of power, bandwidth, modulation technique, and throughput.

Bluetooth 1.2 has a theoretical maximum data rate of about 1 Mbps. The eb500 implementation supports a maximum sustained bidirectional rate of about 230 kbps. I found the range of the eb500 adequate for indoor communications between rooms separated by two walls.

Bluetooth supports a much lower maximum sustained transfer rate than WiFi, but that rate is relatively constant, owing to the frequency hopping spread spectrum (FHSS) modulation. The eb500 maintains message throughput by changing frequency about 1,600 times per second.

Of the two dozen profiles compatible with the Bluetooth standard, the eb500 supports the widely used Serial Port Profile. After installing the software included with the eb500, the device looks like a serial port to a PC. I don't consider security an issue in robotics communications, but if you suspect your neighbor might take control of your robot during the night, Bluetooth offers several provisions for security, including encryption and access control.

As with WiFi, one of the advantages of working with the eb500 is that many PCs, laptops, and PDAs have Bluetooth connectivity. I've used my HP2755 pocket PC with good results. For a laptop without integrated Bluetooth, the Linksys Bluetooth USB adapter is a good option. There are many similar adapters available on the market.

The maximum message throughput of 230 kbps limits tele-autonomous communications to data from a few sensors, perhaps an audio feed, and video at a few frames per second. A bandwidth-saving alternative is to use a wireless spy camera for video and the eb500 for control codes and sensor data, as in Figure 4. Using this configuration, you'll probably run out of program space on your BASIC Stamp before you saturate the eb500.

The eb500 is limited to a robot based on the Parallax Board of Education or other board that provides an AppMod header, but it's a quick route to a tele-autonomous robot. A7 Engineering includes source code in C++ suitable for developing a robot control handset with a pocket PC.

I have two issues with the eb500. One is a matter of convenience. When mounted in the header, the eb500 is frequently in the way. Not only does it block the camera view of the rear of the vehicle shown in Figure 4, but it can't be tucked away in a larger rover

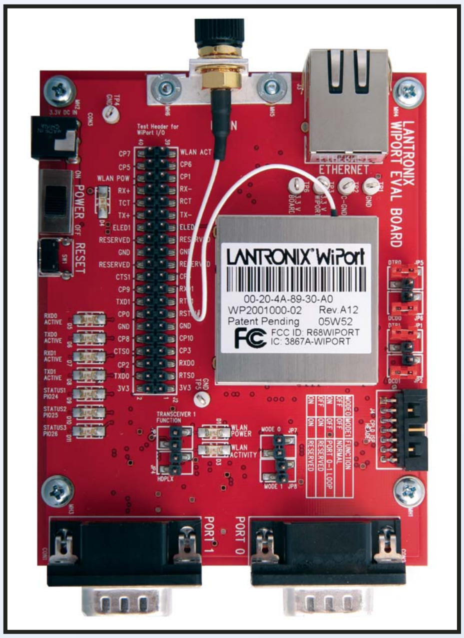


FIGURE 3. Lantronix WiPort evaluation board. The silver WiPort module and pigtail antenna lead are detachable. The 100-gram board (including module) measures 80 x 102 x 17 mm.

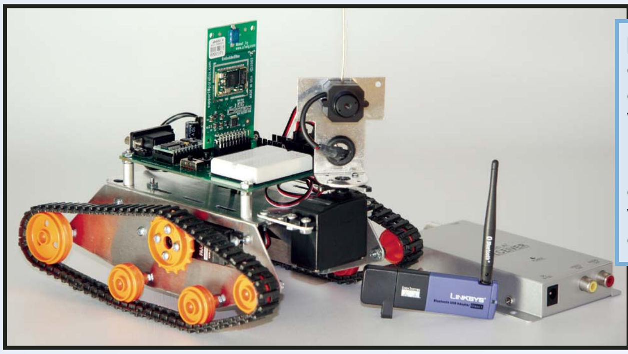


FIGURE 4. A7 Engineering's eb500 Bluetooth transceiver and 1.2 GHz pinhole camera with audio transform a Boe-Bot into a tele-autonomous robot. Linksys Bluetooth USB adapter and the audio/video receiver for the camera are shown on the right.

or crawler robot. An angle adapter for the AppMod header provides some leeway in mounting the board, at the expense of orientating the surface mount antenna as off vertical.

My major issue with the eb500 is something I regard as a design flaw. As I learned the hard way, if I/O pin 5 is configured as an output, the card will literally fry. Even though documentation makes this danger clear, when I loaded a routine for the servos that happened to reset the I/O pins, the card was destroyed. This potential trap seems inappropriate for a product aimed at the hobby market.

XBee-Pro

The XBee-Pro is MaxStream's entry in the rapidly expanding ZigBee (IEEE 802.15.4) market. The XBee-Pro, like the ZigBee modules available from other manufacturers, is intriguing because it supports mesh networks, as well as point-to-point and point-to-multipoint communications options.

Antenna options for the XBee-Pro include a surface mount antenna, a connector for an external antenna (see Figure 5), and a whip antenna attached to the body of the chip. The XBee-Pro, like the lower power XBee, uses an odd 20-pin, 2 mm spacing configuration. MaxStream suggests using a pair of 10-pin, Molex (PN 87340-2024) single-row headers to mount the chip. Like the odd connector used on the WiPort, this imposes some inconvenience in interfacing the XBee-Pro to custom robot circuitry.

The XBee-Pro requires a hardware interface to convert between TTL and RS232 levels for communications with boards and devices. One option is the $60 RS232 interface board, which is essentially an RF modem without the black case. A less expensive option is to use the Max232 level shifter, which can handle up to 129 kbps message throughput.

The mandatory PC-side RF modem is available in either series or USB models. The USB model, which is powered through the USB connector, is simply an XBee-Pro chip and interface board mounted in a 12 x 7 x 3 cm box with signal level, power, and transmit LED indicators and a standard antenna port. As with the WiPort, the XBee-Pro RF Modem ships with virtual com port drivers so that the PC sees the modem as a standard COM port.

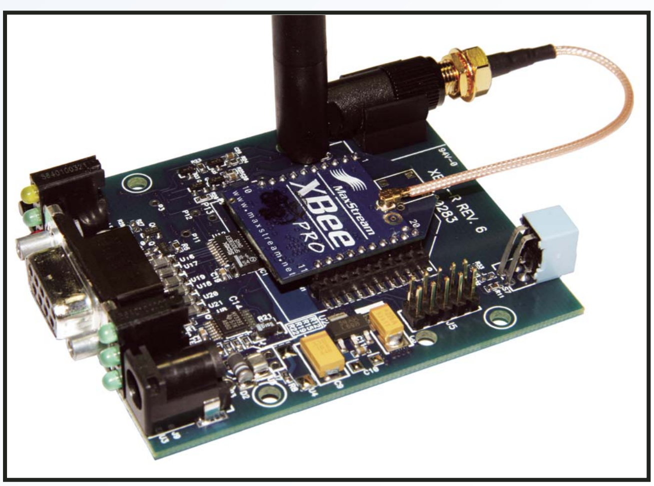


FIGURE 5. XBee-Pro and RS232 carrier board.

When installed under Windows XP, the RF modem acted like a standard Plug and Play device and asked for the driver, which is included on CD-ROM. For a single robot controlled by a PC, that's all there is to the setup. The modem acts like a serial line replacement.

Tele-autonomous operation of multiple robots, or operating a robot in the presence of other RF modems, involves identifying the stationary RF modem as a coordinator, associating the remote XBee-Pro chips with the coordinator, and establishing a personal area network. There are a number of options that may be applicable to some installations, such as various sleep modes, including cyclic sleep that wakes in a pre-determined time. Sleep might be applicable in a remote, solar powered robot, where every microwatt of energy must be conserved.

The software tools that ship with the XBee-Pro are superb. My favorite utility is the range test, shown in Figure 6. As the name suggests, the utility is designed to determine the maximum range of an XBee-Pro. A bar graph is used to display the number and percent of packets that make it from the RF modem to an XBee-Pro and back to the RF modem. With the help of this utility, you can determine best antenna placement and orientation, maximum range in and out of doors, and whether you need to activate the automatic packet resend option.

The simplicity of installation, excellent software suite, low power requirements, extended range, and mesh networking supported by the XBee-Pro place the features set somewhere between those of the WiPort and eb500. However, XBee-Pro message throughput is limited to only 115.2 kbps. Video at more than a few frames per second is problematic, and real-time audio would consume half of the available bandwidth under the best conditions. That said, the XBee-Pro is more than adequate for sending real-time control codes, monitoring battery voltage, power drawn by robot subsystems, and reading a half-dozen range sensors.

Antennas

The maximum range and throughput figures listed by manufacturers of RF communications systems assume best-case conditions, including an optimally configured and placed antenna system. Key antenna variables range from type, orientation, distance from the ground, and location on the robot chassis, to size relative to the chassis. Taken together, these variables profoundly affect communications range, receiver noise level, radiation and reception directional properties, security, as well as susceptibility to interference from — and potential to cause interference to — other components on the robot.

Types

The three basic antenna configurations used in tele-autonomous robotics are surface mount, affixed full-size, and remote full-size. Surface mount antennas contain small surface mount devices, such as the Centurion BlueChip shown in Figure 7. The 12 x 6 x 2.5 mm device, which weighs only 0.21 grams, replaces a 54 mm, 8 gram antenna. The primary penalty for this size and weight savings is a loss in efficiency, relative to a full-sized antenna.

Affixed full-size antennas — an option on the XBee-Pro — provide the efficiency of full-size antennas with a marginal weight penalty. However, an affixed full-size antenna requires the XBee-Pro chip to be mounted horizontally. As with surface mount antennas, affixed antennas can't be buried in the bowels of a robot chassis or touch any of the wires or cables connecting other components.

Remote full-size antennas are connected to the communications device by means of a cable. The advantage of a remote full-size antenna is flexibility in placement of the transceiver and in selecting the optimal location of the antenna. For a mobile robot, this usually means as high as possible on the largest contiguous piece of metal on the chassis.

Groundplane

Electrically, the shortest antenna that can resonate at a specific frequency is a half wavelength. For this reason, an antenna composed of two quarter-wave elements or poles placed end-to-end and fed in the center — a dipole — is typically considered the standard for gain and efficiency comparisons. A quarter-wave vertical antenna can resonate if the "missing" quarter-wave element is supplied by a conducting surface or groundplane, typically below the vertical antenna element. At frequencies up to tens of megahertz, the physical ground can provide this groundplane for a fixed antenna. However, for microwave robotics communications, a metal or other conducting surface must be available on the robot proper.

The printed circuit board groundplane for the quarter-wave, surface mount antenna used in the eb500 is visible in Figure 7. With the integrated groundplane, the eb500 can be used with a plastic or fiberglass robot. In contrast, a remote quarter-wave antenna must be attached to a metal surface on the robot — the larger, the better. Another option is to simply use a vertical dipole antenna and incur a modest weight and size penalty.

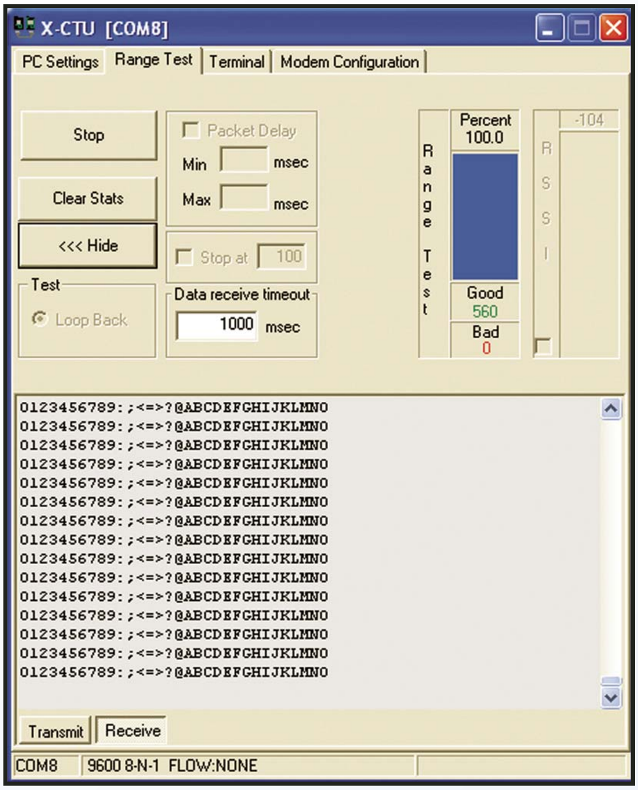


FIGURE 6. Range test screen for the XBee-Pro.

Polarization

The physical orientation of the antennas used on your PC and robot should match. That is, a vertical antenna generates vertically polarized RF signals that are best detected by a vertically oriented receiver antenna. Although you're free to use horizontal, vertical, or circular polarization, vertical antennas are the lightest, most practical alternative for terrestrial tele-autonomous robots. For best results, use PCMCIA and USB wireless adapters with external antennas that can be manipulated to provide true vertical polarization.

Directivity and Gain

Antenna directivity and gain go hand-in-hand. A vertical dipole or vertical quarter-wave antenna radiates equally well at all directions in the horizontal plane, with virtually no energy radiated perpendicular to the horizon. Assuming robot and PC control station antennas are at ground level, the more energy directed toward the horizon, the greater the communi-

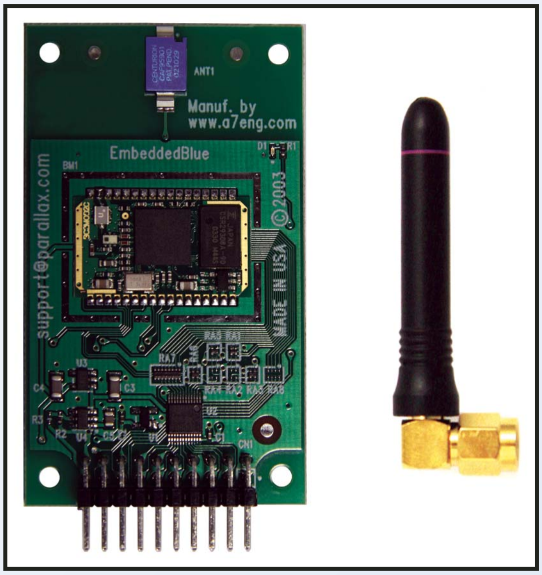


FIGURE 7. The Centurion BlueChip™ quarter-wave, surface mount antenna near the top of the eb500 (left) is electrically equivalent to the full-sized, quarter-wave antenna (right).

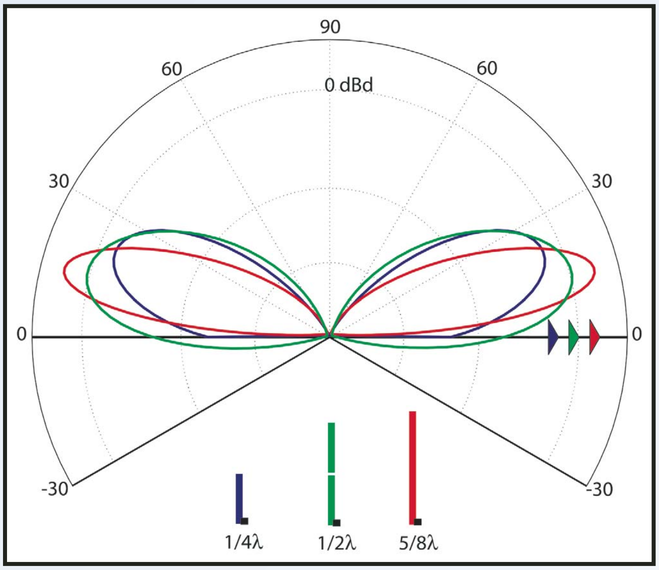


FIGURE 8. Idealized radiation patterns for quarter and five-eights wave vertical whip and half-wave vertical dipole antennas. The patterns represent a slice through a "dough-nut" that surrounds the vertically oriented antennas, with each antenna at the origin of the plot.

cations range.

Figure 8 illustrates the directivity or radiation patterns of three common robot antenna configurations — quarter wave and five-eights wave vertical "whip" antennas and a free-standing vertical half-wave antenna. The quarter-wave antenna creates a radiation pattern with an average vector about 25 degrees above the horizon. In contrast, the five-eights wave antenna generates an average vector about 15 degrees above the horizon, meaning that more energy is directed at an antenna on the horizon. This relative gain amounts to a little over a half dB — which is significant.

The half-wave antenna — which consists of two quarter-wave antennas placed end-to-end and fed in the center — provides about 1/3 dB gain over a quarter wave vertical and about 1/3 dB less gain than a five-eights wave vertical whip. As noted earlier, the dipole is a common reference for other antenna designs, and references to antenna gain are commonly described in terms of dBd — dB relative to a dipole. The other designation often found in advertisements is dBi — dB relative to a theoretical isotropic radiator. When comparing antenna gain figures, determine which unit is referenced. The gain of a thumbtack can look impressive when described in terms of dBi.

High gain omnidirectional antennas, such as multi-element collinear antennas, provide a very low angle of radiation. However, because they weigh several pounds and may extend a meter or more, their use is limited to PC-side communications.

In addition to decreasing the angle of radiation, relative gain can be increased by directing radiation in the horizontal plane. A multi-element Yagi antenna, with a short director in front and large reflector in the rear, achieves gain by eliminating most of the radiation from the sides and rear of the antenna. A 16-element Yagi is capable of providing about 12 dBd gain — equivalent to a dipole antenna fed with eight times the power. Moreover, a Yagi reduces noise pickup and enhances signal reception. The downside is that the antenna must track the robot. A Yagi is inappropriate for robots smaller than a truck.

Cable Loss

The penalty for the flexibility of a remote antenna is cable loss. The thin, flexible RG-174U commonly used on antenna pigtails has a loss of about 3 dB — 50% — per two meters at 2.4 GHz. If your robot design calls for an antenna mounted more than a few inches above the chassis, consider moving the communications device to the antenna mount.

For the PC side, if you use a 16-element Yagi and six meters of RG-174U, the net gain is zero. Instead of RG-174U, use foam dielectric RG-58 for antenna runs up to about two meters. For longer distances, low-loss Heliax® hardline is the best technical option. Unfortunately, the adapters for Heliax and brands of hardline are expensive.

Interference

Before you invest in a high-gain, multi-element Yagi or collinear antenna, remember that the FCC (Federal Communications Commission) limits the effective isotropic radiated power (EIRP) of wireless network devices. One reason for this limitation is to limit interference to other devices operating in the same band. However, operating within legal output power limits doesn't obviate local interference.

I eventually traced the erratic operation of the WiPort noted above to interaction with a digital compass that was mounted a few centimeters from the WiPort's antenna. After I moved the compass to the main robot chassis, replaced the compass wiring with shielded cable, and slipped ferrite beads over the wire ends, the WiPort functioned flawlessly. The bottom line is that although the wireless hardware may be plug-and-play, be prepared for unexpected interactions with other systems.

From Here

The WiPort is the clear winner for throughput, but expect to spend several hours with the documentation. The XBee-Pro provides the greatest range

RESOURCES

WiPort
www.Lantronix.com

CMUCam and CMUCam2
www.cs.cmu.edu/~cmucam/cmucam2/index.html

eb500 Bluetooth Transceiver
www.Parallax.com

X-Bee Pro
www.MaxStream.com

Linksys Bluetooth USB Adapter
www.Linksys.com

Wireless USB
www.usb.org/developers/wusb

Antennas, transmission lines, and interference: The American Radio Relay League
www.arrl.org

and setup is Plug-and-Play, but message throughput is the lowest of the three options discussed here. The eb500 is the quickest and easiest route to tele-autonomous robotics — as long as your robot is built around the BASIC Stamp.

Don't expect one of these or any other single solution to solve all of your communications needs. As noted earlier, a wireless pinhole camera with audio is an expensive, lightweight means of adding high-bandwidth audio and video capabilities to your tele-autonomous robot. Furthermore, don't limit your designs to RF communications.

I've had good luck with the IR Buddy from Parallax ($50/pair) in a hybrid, multi-robot communications system. Sensor data are exchanged between robots using IR Buddy, and an RF system is used to communicate between my PC and the master robot. Communications with the IR Buddy is essentially line-of-sight and limited to about 75 kbps message throughput, which is acceptable for exchanging data that aren't time-critical. Each IR Buddy draws 20 mA on transmit, 2 mA on receive, and weighs only two grams. Hardware and software overhead are minimal.

This snapshot of the rapidly evolving world of wireless communications is hardly the final word. Communications options for tele-autonomous robotics are expanding at an increasing rate. Bluetooth, WiFi, and ZigBee are evolving standards, and the capabilities of future versions will greatly overshadow the performance available today, and even the enhanced standards will be challenged by new technologies.

One of most promising wireless technologies for robot communications is Wireless USB (WUSB), which supports 480 Mbps RF throughput at three meters and 110 Mbps at 10 meters [1]. A PC or other WUSB hub can support a swarm of up to 127 robots, thanks to the use of 7 GHz of bandwidth allocated to ultra-wideband (UWB) radio.

Because WUSB is designed to replace the USB cables used for everything from keyboards and mice to cameras and computer monitors, the chipsets will eventually be more affordable than current wireless options. Moreover, because WUSB is backwards compatible with the wired USB 2.0 standard, software is readily available. WUSB is designed to provide RF throughput in excess of 1 Gbps — more than enough bandwidth to support live video, audio, and sensor feeds from dozens of tele-autonomous robots.

As a final note, of the affordable video cameras with built-in RS232 interfaces, I've found the CMUCam for the BASIC Stamp and general-purpose CMUCam2 are the most well documented and easiest to use. The on-board processing saves communications bandwidth, and ample third-party support simplifies integration. **SV**

REFERENCE

[1] W. Jones, "*No Strings Attached,*" IEEE Spectrum, 43(4): 16-8.

GEERHEAD

by David Geer

Contact the author at geercom@alltel.net

Using Hot and Cold Running Water to Flex Nickel-Titanium Robot Muscles

It Sounds Simpler Than It Is

Make a Muscle!

Dr. Stephen Mascaro and his research team in the department of mechanical engineering at the University of Utah are making robot muscles, working in a vein of research called Wet Robotics. Rather than the fascinatingly technical explanation you might envision, they are simply actuating a metal that contracts in response to hot water and expands in response to cold water. But there is where the simplicity ends.

They are making artificial muscles for robots that work much the same as our muscles do. The technology uses shape memory alloys (SMAs), which are nickel-titanium alloy strands or poles with one of the rarest traits among all metals — the capacity to contract in response to heat.

SMAs

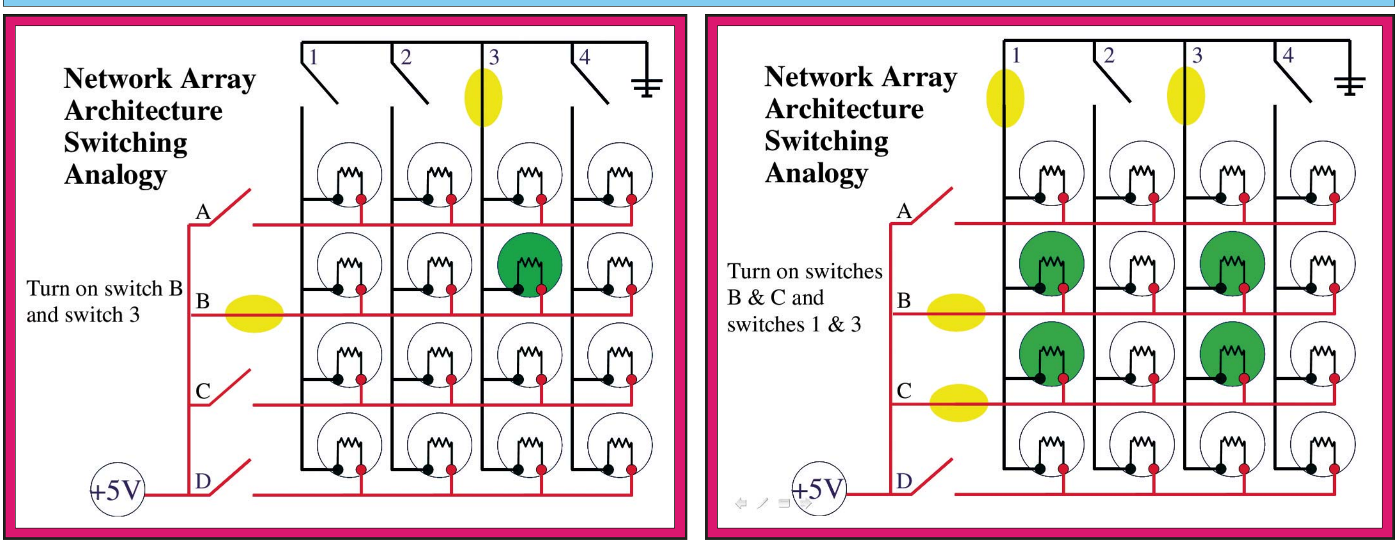
Most metals tend toward expanding in response to the application of heat. Nickel-titanium does just the opposite. "What's going on is that they [the nickel-titanium strands] are actually realigning their crystal structure when you heat them up. So, the crystals of nickel and titanium realign into a more compact orientation," says Dr. Mascaro.

Experimentation is ongoing into practical, scalable, and efficient ways to heat up and contract SMAs to make them act like muscles in order to apply them as a robot muscle technology in humanoid robots.

Some researchers (for other applications) heat SMAs by filling them with electric current. This is called Joule or Resistive heating, according to Dr. Mascaro. "You heat them up (to temperatures) above their transition (contraction) temperature. This varies depending on the concentration of nickel vs. titanium," says Dr. Mascaro.

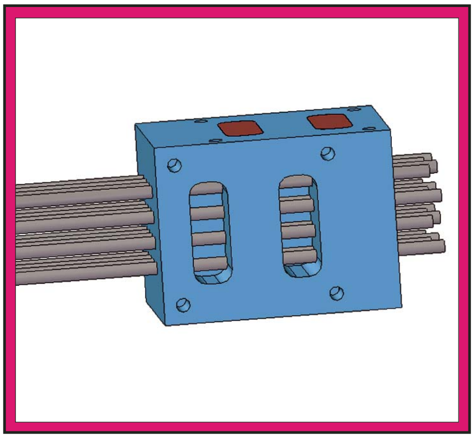
Above their transition temperature, the SMAs are in a "super elastic state" in which they are quite pliable. You can actually heat these wires up and stretch them like a rubber band.

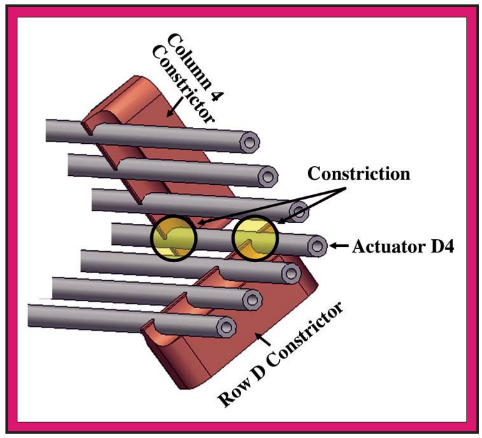
Figures 1 and 2. Demonstrates the Matrix idea of the Matrix Manifold and Valve system (MMV) whereby a particular muscle can be activated based on the column and row that it's in inside the matrix.



You can see the advantage of being able to manufacture SMAs with the balance of nickel and titanium that gives you contraction at the most desirable temperature for a given application.

Some SMAs have transition temperatures at "room temperature" so that you can "grab them" barehanded and "stretch them out" — a very unusual experience the first time around! They use these particular SMAs for dental applications, "to stretch onto the teeth to hold them in place where your dentist wants them," explains Mascaro.





Figures 3 and 4. Details the parts of the Matrix Vasoconstriction Device (MVD), including the housing and water vessel tubing and the constrictors used on the Matrix to constrict and open the flow of water.

Other SMA Properties, Limitations, and Problem Solving

As actuators [potential robot muscles], SMAs are about 1,000x as strong as human muscles for the same size muscle. They don't, however, contract as much as the human muscle. Human muscle contracts 20 percent whereas the SMAs only contract about four percent.

While you can heat up and contract SMAs as fast as you would like using electricity — fast enough to approximate human muscle contraction speeds — they don't cool down fast enough to expand with the speed of human muscle.

How do you cool these SMAs down quickly enough to solve this problem? Some researchers accomplish this by putting the SMAs in a cooling fluid. This introduces another problem: "Now you have the weight of this fluid added to your system and you've lost the original advantage of these actuators — that they are tiny, lightweight, and give you a lot of strength without having the bulk of an electric motor," says Dr. Mascaro.

This is the problem that lead Dr. Mascaro to his idea of how to use SMAs as muscles. With Dr. Mascaro's technology, you embed the SMA wires within tubes of flowing fluid for cooling purposes. This gives your robots their own sort of blood vessels. In the human body, energy is carried to and from the muscle groups via blood vessels. With the robot muscles, you have cold fluid removing heat, and as we will discuss, hot fluid producing the artificial muscle contractions.

When you use electricity, the energy it produces is lost in the cooling fluid. Dr. Mascaro uses hot fluids, in this way the hot and cold fluids are both recycled back into hot and cold reservoirs. This produces better energy efficiency than using electricity.

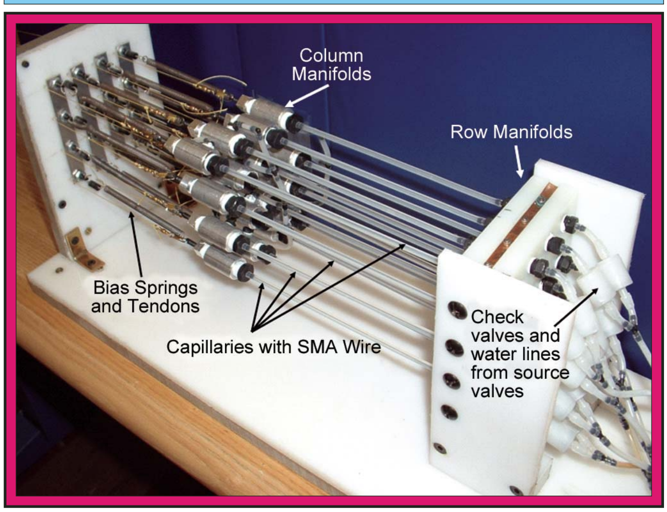
Actuating in Dr. Mascaro's Array Design

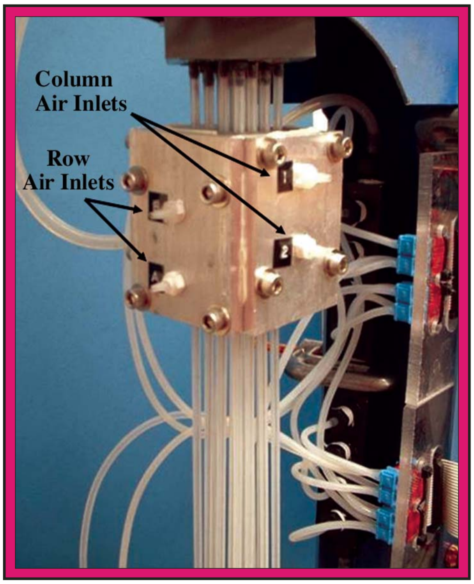
How do you put large arrays of these muscles into a compact area like a robot arm? The advantage

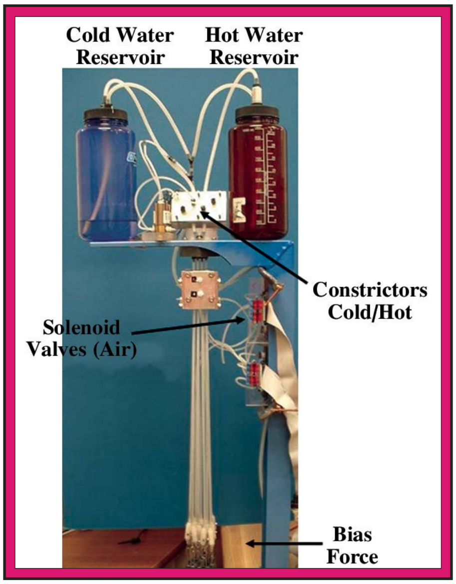
LEAKY, RESISTIVE MUSCLES

This valve-based attempt at actuating heat-responsive contracting nickel-titanium alloy metal strands for robot muscles had some issues: The hot water leaked into tubing that contracted strands the researchers didn't want to contract at the time, taking away some control of the muscles; and the valves were resistive to the water flow, costing some energy (see Figure 5).

Figure 5. According to Dr. Mascaro, this figure shows the MMV prototype, which exposed new challenges in the form of fluid resistance from the solenoid valves and parasitic behaviors such that if the device tried to deliver hot water to one of the actuators, later on in the sequence the hot water might leak into one of the other actuators at a time when they didn't want to activate that muscle.







Figures 6 and 7. Photos of the prototype MVD and the completed MVD, respectively.

maintains so long as you have a high-strength to low-weight ratio. How does this scale so that you can have, say, 100 muscles in a robot arm, and control them all without having to have 100 separate controls for each one?

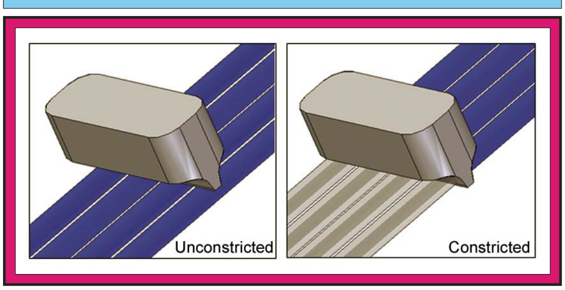
This is where Dr. Mascaro's Matrix Manifold and Valve (MMV) system comes into play. The system arranges the artificial muscles into rows and columns so that only one switch is needed for each matrix manifold and so for several muscles, as well.

In production, a robot limb that would need to extend or leverage something would have matrix manifolds with a hundred muscles in 10 x 10 arrays. The current proof of concept model has 4 x 4 arrays.

A NEW LEASE ON SMA ROBOT MUSCLE LIFE

Constricting and opening the vessels/tubing directly removed the need for valves. This removed the resistance and provided greater control over where the water flowed (see Figure 8).

Figure 8. Demonstrates the new muscle system, which constricts the vessels transporting water to the SMA muscles or "unobstructs" them, rather than using solenoid valves, which presented their own problems in the form of fluid resistance.



Manufacturing

Dr. Mascaro expects that manufacturing techniques and processes will be key in going beyond even the 10 x 10 robot muscle scenario. Connecting the actuators as an integrated array with more than 100 SMA muscles is a scalable manufacturing issue; something that can't feasibly be practical as the work of a few human scientists working by hand in the lab. Manufacturing should take the potential muscle count to hundreds at first, eventually even thousands.

A scalable manufacturing method hasn't been addressed yet, but this will be addressed in the lab, so that it can be taken to production.

Power, Force, and Speed

In the big picture, Dr. Mascaro has figured out how to make the muscles forceful while maintaining their light weight. The trick here is to make them fast while maintaining light weight, to get high-power-to-weight ratio, rather than just high-force-to-weight ratio.

Wet Robotics/Human Body Muscle Comparisons

The cardiovascular system in the human body has many functions:

1. Delivering chemical energy to the muscles.

2. Thermal regulation of the body.

"When your body gets hot during a workout, your blood starts flowing faster, so the heat is transported by your blood stream out from the core of your body to the extremities where sweating removes the heat," says Dr. Mascaro.

Controls in the body constrict and open the blood vessels to regulate your core temperature. With fluid flow in the robot muscles, temperature regulates the muscle contraction and expansion.

MMV

The MMV uses a valve on each row and column of the array. By opening the correct row and column valves in combination, you send hot or cold fluid to the correct SMA.

While the row and column arrays passed muster, the valve system had to go. The valves are standard solenoid valves and while they are common and predictable, the water has to flow through a small hole in the valve. "This introduces fluidic resistance. We can turn the flow off and on to any of the muscles, but when it's on, we don't want any resistance at all, which you can't do with solenoid valves," explains Dr. Mascaro.

Solution: Rather than putting valves in inline with the flow of fluid, why not try to constrict the vessels used to transport the fluid? Dr. Mascaro has developed a Matrix Vasoconstrictor Device (MVD) to replace the MMV. Rather than using the solenoid valves, this device constricts all the rows and columns of the vessels using air pressure, so that they can completely collapse the vessel, stopping all of the fluid flow, or open it and fluid flows without any resistance, according to Dr. Mascaro.

As you might assume, the SMA muscle apparatus and MVD constitute a fully closed system in which the same fluid is used all the time, a fluid that doesn't dissipate. "The only energy input to the total system is keeping the hot water hot, the cold water cold, and some means to pressurize the system to keep the flow moving," says Dr. Mascaro.

You Gotta Have Heart

Dr. Mascaro and crew are now working on a robotic heart to pump the fluid in and out of the SMA muscles. The same SMA muscles — which are nourished by the hot and cold fluid pumped by the heart itself — will power the robotic heart, explains Dr. Mascaro. However, this work is just in the beginning stages and does not yet appear in any of Dr. Mascaro's papers as of this writing. **SV**

RESOURCES

Stephen Mascaro's Page
www.mech.utah.edu/~smascaro/research.htm

Wet Robotics, Image 1
www.mech.utah.edu/~smascaro/video/WetSMA2Hz.mpg

Wet Robotics, Image 2
www.mech.utah.edu/~smascaro/video/vastactuators.mpg

Mascaro paper on Wet Robotics
www.mech.utah.edu/~smascaro/pdf/Mascaro-2003-IROS-WetSMA.pdf

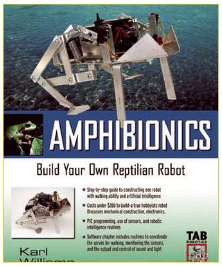
Mascaro paper on Wet Robotics
www.mech.utah.edu/~smascaro/pdf/Mascaro-2003-ICRA-WetSMA.pdf

Other University of Utah Robotics
www.cs.utah.edu/research/areas/robotics/robotics.html

Mind Candy For Today's Roboticist

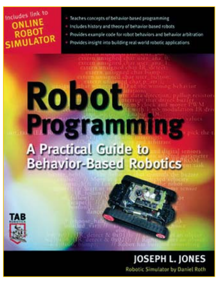
The *SERVO* Store

Amphibionics
by Karl Williams



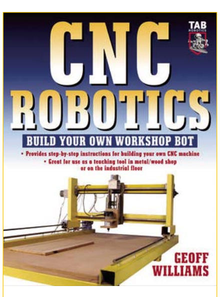
This work provides the hobbyist with detailed mechanical, electronic, and PIC microcontroller knowledge needed to build and program a snake, frog, turtle, and alligator robots. It focuses on the construction of each robot in detail, and then explores the world of slithering, jumping, swimming, and walking robots, and the artificial intelligence needed to make these movements happen with these platforms. Packed with insight and a wealth of informative illustrations. **$19.95**

Robot Programming
by Joe Jones / Daniel Roth



This ingenious book/website partnership teaches the skills you need to program a robot — and gives you a virtual robot waiting online to perform your commands and test your programming expertise. You don't need to know robotics or programming to get started! Using an intuitive method, *Robot Programming* deconstructs robot control into simple and distinct behaviors that are easy to program and debug with inexpensive microcontrollers with little memory. Once you've mastered programming your online bot, you can easily adapt your programs for use in physical robots. **$29.95**

CNC Robotics
by Geoff Williams

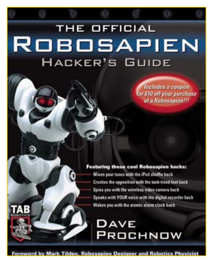


CNC Robotics gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot — and that can be customized to suit your purposes exactly, because you designed it. Written by an accomplished workshop bot designer/builder, this book gives you all the information you'll need on CNC robotics! **$34.95**

SERVO CD-Rom

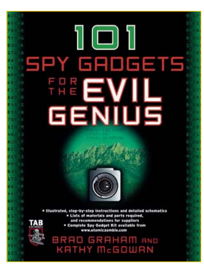
Are you ready for some good news? Starting with the first *SERVO Magazine* issue — November 2003 — all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 1, issues 11-12 and Volume 2, issues 1-12, for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

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**

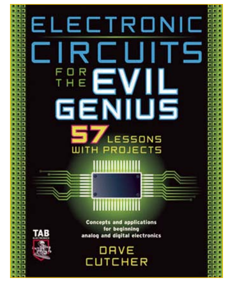
101 Spy Gadgets for the Evil Genius
by Brad Graham/Kathy McGowan



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**

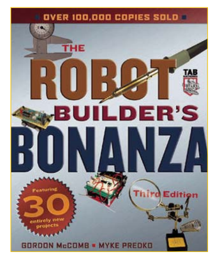
We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Electronic Circuits for the Evil Genius
by Dave Cutcher



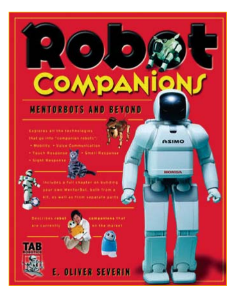
Cutcher's 57 lessons build on each other and add up to projects that are fun and practical. The reader gains valuable experience in circuit construction and design and in learning to test, modify, and observe results. Bonus website **www.books.mcgraw-hill.com/authors/cutcher** provides animations, answers to worksheet problems, links to other resources, WAV files to be used as frequency generators, and freeware to apply your PC as an oscilloscope. **$24.95**

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**

Robot Companions
by E. Oliver Severin

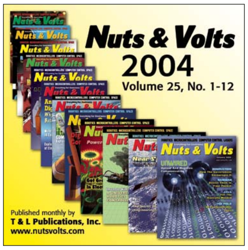


Inside *Robot Companions*, you'll find all the details, plans, and information you need to make a robot partner part of your daily life, at a price you can afford. Author E. Oliver Severin, originator of some of the technologies that make robots friendly, useful, and educational, shows you how to find or build your own robot helpmate — either from commercial kits or an assembly of separate, off-the-shelf parts. **$24.95**

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

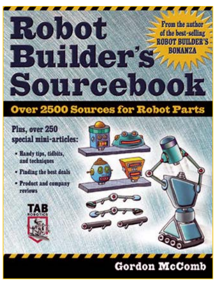
Mind Candy For Today's Roboticist

Nuts & Volts CD-Rom



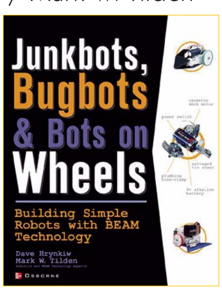
Here's some good news for *Nuts & Volts* readers! Starting with the January 2004 issue of *Nuts & Volts*, all of the issues through the 2004 calendar year are now available on a CD that can be searched, printed, and easily stored. This CD includes all of Volume 25, issues 1-12, for a total of 12 issues. The CD-Rom is PC and Mac compatible. It requires Adobe Acrobat Reader version 6 or above. Adobe Acrobat Reader version 7 is included on the disc. **$29.95**

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**

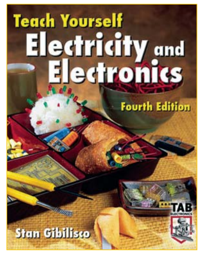
JunkBots, Bugbots, and Bots on Wheels
by Dave Hrynkiw / Mark W. Tilden



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**

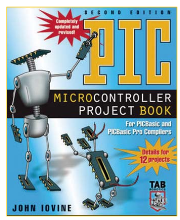
Check out our online bookstore at www.servomagazine.com for a complete listing of all the books that are available.

Teach Yourself Electricity and Electronics — Fourth Edition
by Stan Gibilisco



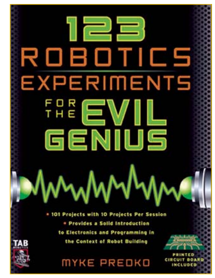
Learn the hows and whys behind basic electricity, electronics, and communications without formal training. The best combination self-teaching guide, home reference, and classroom text on electricity and electronics has been updated to deliver the latest advances. Great for preparing for amateur and commercial licensing exams, this guide has been prized by thousands of students and professionals for its uniquely thorough coverage ranging from DC and AC concepts to semiconductors and integrated circuits. **$34.95**

PIC Microcontroller Project Book
by John Iovine



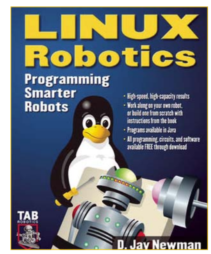
The PIC microcontroller is enormously popular both in the US and abroad. The first edition of this book was a tremendous success because of that. However, in the four years that have passed since the book was first published, the electronics hobbyist market has become more sophisticated. Many users of 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.

EVENTS CALENDAR

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

The organization that came up with the Space Elevator: 2010 contest — the Spaceward Foundation — is working on a new robotics competition. The new event is called the Mars Robotic Construction challenge and there's $250,000 in prize money up for grabs.

The Mars Robotic Construction challenge is designed to be a realistic scenario in which one or more robots must complete a construction task that would be a typical part of assembling an ore processing plant or oxygen generation plant on Mars. The initial task will be to construct a water-tight pipeline between a "resource generator" and a storage tank. Each team is given a maximum of 24 hours in the Marssimulation arena to achieve the objective. The task is considered complete when the pipeline successfully transfers water to the storage tank.

Just like the real thing, the teams will never see the Martian environment except through the sensors of their robots. The human members of the team will remain at home while the robots are shipped to the competition site. Technicians will take care of placing the robots in the Martian arena.

Even the 20 minute communication delay with Mars will be simulated. Each team is allowed to communicate with and direct their robots from home. The team will use the Internet with an added 20 minute latency to direct the robots. Obviously, a great deal of autonomy and cooperation among the robots will be required in order to complete the task in the amount of time allotted.

Part of the goal is to better understand where the optimum dividing line between human control and autonomy lies for this type of robotic construction task. The final details — such as total weight limits for the robots and specs for the pipeline components — are still being worked out. If you'd like to find out more or get involved, please visit the Spaceward Foundation's Mars Construction challenge website at **www.spaceward.org/marsChallenge.html**

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**

September

1-4 **DragonCon Robot Battles**
Atlanta, GA
Radio-controlled vehicles destroy each other at a famous science fiction convention.
www.dragoncon.org

9 **SWARC Texas Cup**
Mike's Hobby Shop, Carrollton, TX
Radio-controlled vehicles destroy each other Texas-style.
www.robotrebellion.net

16-17 **RoboCup Junior Australia**
University of NSW, NSW, Australia
There are over 600 RoboCup Junior teams in Australia. Regionals narrow this down to about 200 teams who will compete at the University of NSW to see who's the best at building LEGO-based autonomous soccer robots.
www.robocupjunior.org.au

30 **Robothon**
Center House, Seattle Center, Seattle, WA
Events continue on October 1 for two full days of robot contests that include line-following, line-maze, Robo-Magellan, walker races, mini sumo, and 3 kg sumo.
www.robothon.org

October

14 **Robot-Liga**
Kaiserlauter, Germany
Includes mini sumo, line search, labyrinth, master labyrinth, robot volley, and robot ball.
www.robotliga.de

20 **Elevator:2010 Climber Competition**
Las Cruces, NM
Autonomous climber robot must ascend a 60 meter scale model of a space elevator using power from a 10 kW Xenon search light at the base.
www.elevator2010.org/site/competition.html

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

Wood — It's What Your Robot Is Made Of

Modern robot building is a decidedly 21st Century technology. So, it may appear rather daft to use one of civilization's oldest manufactured materials to build your next 'bot: Yes, I'm talking about wood. In this age of high-tech plastics and engineered metals, wood seems an archaic choice, but, in fact, it's a supremely logical one. Wood is relatively cheap and easy to get, plentiful, and most varieties are easy to work with.

The structure of your robot is one key to its success. For the average desktop-sized automaton, your robot's frame needn't be cross-welded and powder coated to resist rust. It can be as simple as two hunks of wood you found on aisle nine at the hardware store, lashed together with some glue and screws.

In the column, we discuss the use of wood to make workable and near-universal bases for the average wheeled and tracked robots.

A Brief Overview of Material Choices

Before we get going, let's review the three most common construction materials used for amateur robot-making. While there are many types of materials that may be used, three stand out as particularly advantageous to the robot builder: wood, plastic, and metal.

Let's start with wood. Wood is universally available, reasonably low-cost, and easy to work with using ordinary shop tools. Hardwood plywoods (the recommended wood for most robot bases) are very sturdy and strong. On the downside, wood is not as strong as plastic or metal. It can warp with moisture (it should be painted or sealed), and it can crack and splinter under stress.

Next is metal — in all its forms. Aluminum and steel are the most common for robot construction. Metal is very strong. Aluminum is available in a variety of convenient shapes (sheet, extruded shapes). Metals are dimensionally stable even at higher loads and heats, and they won't splinter or crack, even when used with a fairly heavy robot. However, metal tends to be the heaviest of all the common construction materials. Working with metal requires power tools and sharp saws/bits for proper construction. All in all, metal is harder to work with than all the other materials, and can be the most expensive of the lot.

Finally, there's plastic. It's strong and durable and comes in many forms — including sheets and extruded shape. Several common types of sheet plastic (acrylic, polycarbonate) are readily available at hardware and home improvement stores. Other types can be purchased via mail order at reasonable cost. All plastics melt or sag at higher temperatures, even as low as 165 degrees Fahrenheit. Some types of plastic — such as acrylic — can crack or splinter with impact or when drilling. PVC and styrene plastics are not dimensionally stable under stress.

There is no single "ideal" material for constructing robots. Each project requires a review of:

- *The robot itself*, especially its physical attributes (large, small, heavy, light).

- *The tasks the robot is expected to do*. Robots that do not perform heavy work (lifting objects, smashing into other robots) do not need heavy duty materials.

- *Your budget*. Everyone has a limit on what he or she can spend on robot materials. Tight budgets call for the least expensive materials.

- *Your construction skills*. Wood and plastic robots are easier to build than metal ones.

- *Your tools*. Building metal robots requires heavier-duty tools than when building wood or plastic 'bots.

Making Wood Bases and Frames for Robots

Wood is among the oldest construction materials on the Earth, and is arguably the most useful. Contrary to the naysayers, it even has applications in robot building. Wood is fairly cheap and easy to find, and it's easy to work with using common tools.

Solid bases made of wood are best constructed using hardwood plywood.

TABLE 1. Recommended Grits: Aluminum Oxide or Garnet

	Grit			
Use	F	M	C	EC
Heavy sanding			X	X
Moderate sanding		X		
Finish sanding	X			

Grit Key	Name	Grit
EC	Extra course	30-40
C	Coarse	50-60
M	Medium	80-100
F	Fine	120-150

These are available in various thicknesses, with 1/4" being a good all-around choice for a robot under 10 inches in diameter. Even smaller robots can use the 1/8" or 5/32" thickness. Hardwood plywood is routinely available at craft and hobby stores, as well as many home improvement stores. You can buy it in convenient cut-down sheets of 24" x 24" (and even smaller). One common name for hardwood plywood is "aircraft plywood," as used in the construction of model airplanes. However, aircraft plywood doesn't mind if you use it for a ground-based application like robotics!

For a square base, you need only cut the wood to size using a hand or power saw. A power saw with a guide fence (e.g., a table saw) is preferable, as this will yield the straightest cut. You can mark the desired dimensions right on the surface of the wood using a #2 pencil. It's important to not assume the plywood pieces you buy are cut square. Verify and trim the sides to ensure squareness.

For a shaped base — round, oval, or something else — a scroll saw or band saw make short work cutting the wood. Draw the desired shape on the wood using a pencil, then follow the lines during cutting. With some practice, you can even cut nearly round bases, though a better approach is to use a small router or hobby tool and a circle-cutting attachment. Drill a hole for the center of the base, and screw the attachment in place. Position the router at the desired distance from the center, and make the cut.

Because wood is a rather heavy material, it's sometimes advantageous to not use a solid piece for the robot base. Instead you can create a frame out of wood, and then cover the frame with a lighter material, such as 1/8" plastic or wood, or even a piece of Formica countertop. Wood frames can be constructed using strips of hardwood. You can buy the strips pre-made, or cut your own if you have a table saw. For frames under 12" square, strips 3/8" to 5/8" wide are adequate — use 3/4" or even 1" wide wood for larger robots.

Wood selection is critical. Stay away from softwoods, such as pine, fir, and redwood. Though inexpensive and easy to find, these woods are not strong enough, except for the smallest of bases. Several good all-around hardwoods that are available throughout most of North America are alder, ash, American beech, and poplar. These woods are also among the least expensive of the hardwoods.

Square frames can be constructed using miter cuts; nail, staple, or glue the corners together, or better yet, use flat corner brackets for a rock-solid construction. The brackets can be secured using wood screws or (preferred) machine screws and nuts. Box frames can be constructed using two (or more) square frames, anchored together with plywood or plastic panels. As needed, cut out segments of the panel to save weight. Avoid removing too much material, or the panel will be weakened. The panel can be affixed to the frame using wood screws or machine screws and nuts.

Wood Isn't Finished Until It's Finished

You can extend the life of your wood robot bases and frames, not to mention enhance their looks, with simple finishing. Wood finishing involves sanding — which smoothes down the exposed grain — then painting or sealing. Small pieces can be sanded by hand, but larger bases benefit from a power sander.

Sandpapers are available in a variety of grits — the lower the grit number, the more coarse the paper.

WORKING WITH WOOD

Of all the materials used in robot making, wood is the easiest to work with. Yet wood is not without its limitations and troubles. First and foremost is that wood can warp if it is exposed to moisture. The warpage can occur when the wood swells with moisture, or when it dries back out. Warpage is difficult to remove, so it's best to just avoid it in the first place. Keep your wood stock in a cool, dry place, and once you build something with it, paint or seal the wood to prevent warpage later on.

Use only sharp saws and drill bits. Dull tools make you work harder, and the extra friction can burn the wood. When drilling, back the wood with a piece of scrap. This helps prevent splintering as the bit punches through. You may also place a piece of masking tape on the entry and exit points of the wood. This also helps reduce splintering.

Avoid bearing down hard when drilling. This only serves to dull the tip of the bit. Work slowly. Let the tool do most of the work. To drill large holes, start with a smaller hole. For deeper wood, periodically lift the bit out of the hole to remove the built-up saw dust and wood chips.

When cutting wood, going with the grain will go faster than going across the grain. You can tell the direction of the grain by looking at the surface of the wood. The grain is the telltale growth marks in the wood. Alter the rate of feed into the saw blade accordingly. Slow down when cutting across the grain.

Lastly, while hardwoods are recommended for robot bases and frames over softwoods, stay away from the very heavy hardwoods such as oak. Though oak is plentiful and fairly inexpensive, it adds too much weight to your robot.

The recommended approach is to start first with a coarse grit to remove splinters and other rough spots, then finish off with a moderate or fine grit paper. For wood, you can select between aluminum oxide or garnet grits. Aluminum oxide lasts a bit longer. Sandpapers for wood are used dry. For hand sanding, wrap the paper around a block to provide even pressure over the wood. The job will go a lot faster if you have a power sander. Wear a respirator mask to keep the sawdust out of your lungs. Take a look at Table 1.

Wood can be sprayed or painted with a brush. Brush painting with acrylic (available at craft stores) produces excellent results with little or no waste. One coat may be sufficient, but two or three coats may be necessary. Woods with a so-called open grain may need to be sealed first using a varnish or sealer, or else the paint will "soak" into the wood, no matter how many coats you apply. You may also opt to skip the painting step altogether, and apply only the sealant.

Assembly With Connectors, Hinges, and Gussets

Basic frames are constructed using connectors, fasteners, or hinges. A square frame can be made using four lengths of wood, and four corner brackets, which you can get at any hardware store. A box frame can be constructed using 12 lengths of wood, four corner brackets, and eight T-shaped brackets. Again, these brackets are common finds at the hardware store.

All frames will have screw holes for assembly. For thicker material — say over 5/8" or so — you can use \$6 or \$8 flathead wood screws for the assembly. For thinner woods, or to allow easier disassembly and reassembly of the base, use 6-32 or 8-32 machine screws, nuts, and washers. The washers are important so that the screw heads and nuts don't dig into the wood, possibly cracking the wood, and making the joint weak.

Remember: Safety First!

Unless you happen to buy your robot with all its wood pieces already cut out for you (and yes, there are such robotics kits available; see the sidebar elsewhere in this column), working with wood means working with tools. And working with tools involves some risk. Knife blades can break off and fly through the air. Chips of wood, plastic, or metal can be propelled into the eyes of unsuspecting onlookers. Fingers can stray too close to spinning saw blades. Safety is often taken for granted in the shop, and that's bad. Always work with safety in the front of your mind, not the back. You'll enjoy robotics so much more with all your body parts intact and functional.

- Safety glasses or goggles should be worn at all times, even during simple hardware assembly. Get a pair that is comfortable, and provides an unobscured view. Safety glasses should also be worn by any spectators in the shop.

- Ear protection is highly recommended when using power tools, such as saws or high-speed drills, and especially pneumatic tools.

- Periodically inspect your tools to ensure they are in proper working order, and that all safety devices are functional. This is particularly important for saws. The guards should open easily, and not jam as the saw cuts into the wood.

- Never defeat the safety device of a tool. Eventually, the tool will defeat you.

- Keep your cutting and sawing tools sharp. Resharpen or replace dull tools. Among the most common and serious accidents occur because a saw or drill bit is dull, and the operator applies too much force on the tool.

- When using power tools, don't hold the work in your hands. Use clamps and vises.

- Never work barefoot. It poses an electrical shock hazard, and you may step on sharp shards and scraps of material.

- Don't wear loose-fitting clothing or jewelry. Roll up the sleeves of your shirt, and remove your tie, or tuck it into your shirt.

- A shop apron will keep your clothes clean. Do not tie the apron in the front.

- Work only in well-ventilated areas, especially when applying paints or adhesives, or when soldering, brazing, or welding.

Sources

Hardware stores and home

ROBOT KITS MADE OF WOOD

A few online companies sell robot kits using precut wood. The wood is cut using either a computerized router or a laser. For example, Mekatronix (**www.mekatronix.com**) has offered the Talrik and other wood-based robot kits for many years. (The company also offers the kits in ABS plastic.) The wood kits use five-ply aircraft hardwood plywood, and are available from the manufacturer or through a number of distributors worldwide.

In the UK, Milford Instruments (**www.milinst.com**) sells a number of unique wood-based kits, including a talking head and a bipedal walker. Both are motorized using RC servos.

C&S Sales — a company best known for its electronic tools and test equipment — sells a line of low-cost wood robot kits. These kits are for lightweight duty only, constructed of a thin punched wood laminate. On the plus side, the price for most kits is under \$20, and construction is simple, requiring only a screwdriver.

improvement outlets are excellent sources for wood, fasteners, basic hardware, tools, tool accessories (saw blades, bits, etc.), paint, glues and adhesives, casters, wheels, plastic sheeting, metal stock, and hundreds of other items.

The resources listed here either have an online presence, or are national chains. In the United States, many hardware stores belong to one of several co-operative chains, such as True Value and Ace. Other independent hardware stores may be located near you, and you should not hesitate to visit them.

Keep a notebook of what each area hardware store offers, so the next time you need a particular item, you'll know just where to go to get it.

In addition to these sources, be sure to check out the local hobby stores for sheets of hardwood plywood used in the construction of model airplanes and ships, as well as craft stores for smaller precut wood pieces and woodworking tools.

Ace Hardware
www.acehardware.com

Independently-owned chain of hardware stores. In my experience, a number of the Ace Hardware stores I frequent have a number of products not carried by the "big guys" (Lowe's and Home Depot), such as unusual fasteners and hardware. Don't overlook the small stores in your area for unique components for your robots.

Aubuchon Hardware
www.aubuchon.com

Online and local hardware stores across the northeast. Online catalog boasts over 70,000 items. Products include hand and power tools, fasteners, hardware, plumbing, and electrical.

B&Q
www.diy.com

B&Q is a large do-it-yourself chain in the United Kingdom, with lots of goodies. Check the web page for near-by store locations and phone numbers.

Constantines Wood Center
www.constantines.com

Not a hardware store per se, but a huge depot of all kinds of woods. They also sell tools, adhesives, and woodworking hardware.

CornerHardware.com
www.cornerhardware.com

Now you can go to the hardware store using only your computer. **CornerHardware.com** is like the neighborhood hardware store, except it's open on Sunday.

Do it Best
www.doitbest.com

Independent chain of hardware

stores, in the United States and worldwide. Some online sales of smaller, shippable items.

Home Depot and Home Depot Maintenance Warehouse
www.homedepot.com
www.hdsupply.com

The world's largest home improvement chain. The company also has a printed catalog for maintenance and repair supplies, and an online e-commerce store. Depending on where you live, same-day or next-day shipping may be available to you; otherwise you'll wait two or three days to get your stuff. Use the locator at the site to find a warehouse near you (they're in major cities across the US).

Lowe's Companies, Inc.
www.lowes.com

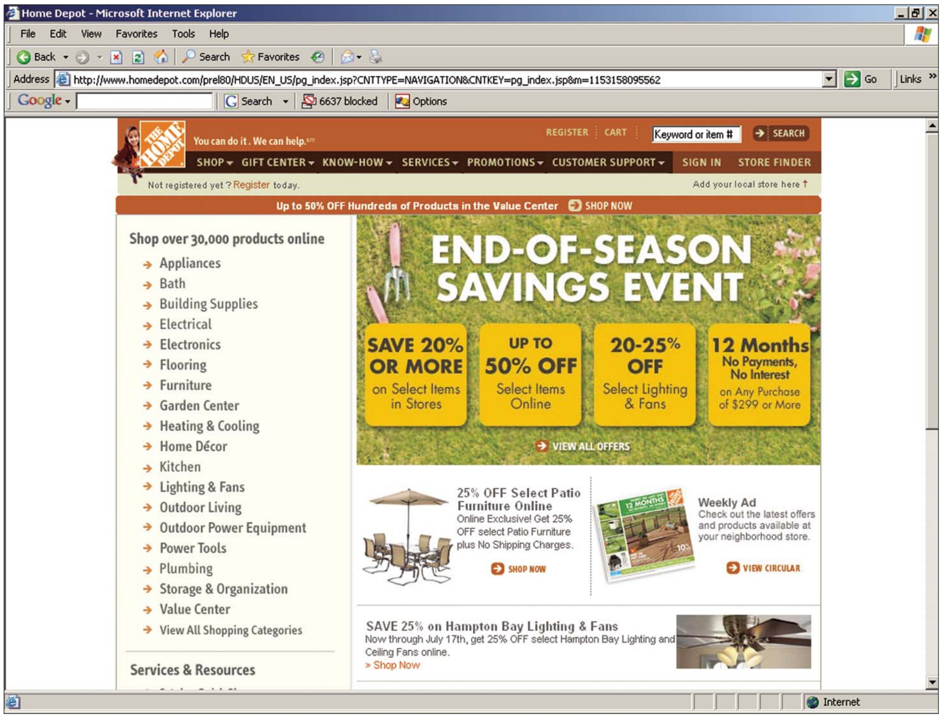
An alternative to Home Depot, with a selection of fasteners, hardware, and tools. Retail stores and online sales. Check the web page for a store locator. Lowe's has 600+ superstores in some 40 states. The site includes a "how-to library" on home repair and remodeling. I looked ... nothing about building robots. But, some of the articles might be useful to learn about materials and tools, and the best way to use them.

Rockler Woodworking and Hardware
www.rockler.com

Rockler carries hand and power woodworking tools, hardware, and wood stock (including precut hardwood plywoods). Among important hardware items are medium-sized casters, drop-front supports (possible use in bumpers or joints in robots), and drawer slides.

TruServ Corporation
www.truserv.com

TruServ is the corporate parent of a number of hardware stores, home improvement centers, and industrial supply outlets, including the True Value hardware store chain in the US. See also **www.truevalue.com**.



One of the world's largest retailers, let alone home improvement stores, Home Depot boasts local outlets in most areas of the country, as well as online sales.

Woodcraft
www.woodcraft.com

Woodcraft is a woodworking superstore. They carry everything you'd need — wood, tools, and materials. Check out their map of local stores, and their online instruction.

Woodworking.com
www.woodworking.com

Woodworking.com is an online resource on all things wood. Tools, materials, website reviews, how-to articles, user-to-user forums, you name it. **SV**

Programmable Logic

by Gerard Fonte

The goal of this bimonthly column is to provide a basic understanding of the various programmable logic techniques.

There are a lot of powerful low-cost components available today that are rarely considered by hobbyists — and even some engineers — because of unfamiliarity.

You have to be comfortable with the idea and concepts of programmable logic before you will be likely to employ them.

Designing a Seven-Segment Counter

Or How to Implement an Arbitrary Count Sequence

Yes, that's right, not a seven-segment counter/decoder, but a counter that generates the seven-segment pattern directly. This is an example of how to approach the problem of a non-standard counting sequence.

Old habits die hard. Digital designers used to be forced into certain designs because of the limitations of the available ICs. It wasn't practical to consider counting sequences other than BCD (Binary Coded Decimal), Johnson, LFSR (Linear Feedback Shift Register), and binary. These counting sequences were the only ones available or easily implemented. If another sequence was required, then one of these counters was decoded into the desired pattern.

This decoding obviously required additional logic and reduced overall performance. This basic and common approach is shown in Figure 1. Figure 1 also defines the LED segments with the letters "a" through "g."

However, with the advent of FPGAs (Field Programmable Gate Arrays), the counter implementation logic is not limited to conventional patterns. Arbitrary counter patterns can be chosen, as needed, to streamline logic throughout the design. What's more, this approach will usually reduce resources, as well as improve performance.

Let's take the example of a decimal counter that is to drive a seven-segment LED display. This is probably one of the most difficult patterns to generate. It has no logical sequence. And, with seven output lines, there are 128 possible combinations of which only 10 are used. The conventional approach would be to take a BCD counter and use it to drive a seven-segment decoder. This is shown at the gate level in Figure 2 (synchronous counter adapted from 1986 Xilinx data book).

The Xilinx 3000 series part is a convenient but arbitrary choice for implementation. For a 3000 series chip, this requires 3.5 CLBs (Configurable Logic Blocks) for the decoder (seven outputs plus logic) and two more CLBs for the BCD counter (four flip-flops plus logic).

Two layers of logic and one interconnect slow the input to output speed. (The standard Invert, AND, OR logic is implemented. However, the inversion is included as part of the RAM look-up logic-table, indicated with the "bubbles" on the AND inputs, rather than with discrete inverters. Additionally, sometimes it's easier to decode when a signal is off rather than when it is on. In such cases, the final OR is replaced with a NOR.)

Conversely, designing a fully synchronous counter to count directly in a seven-segment pattern will be shown to take a total of four CLBs and require only a single layer of logic. Thus, this approach saves 1.5 CLBs (out of 5.5 or 27%) and has front-to-back delay reduced by over 50%.

Defining the Sequence

Of course, the big problem is how to go about designing an arbitrary sequence count. The solution is not

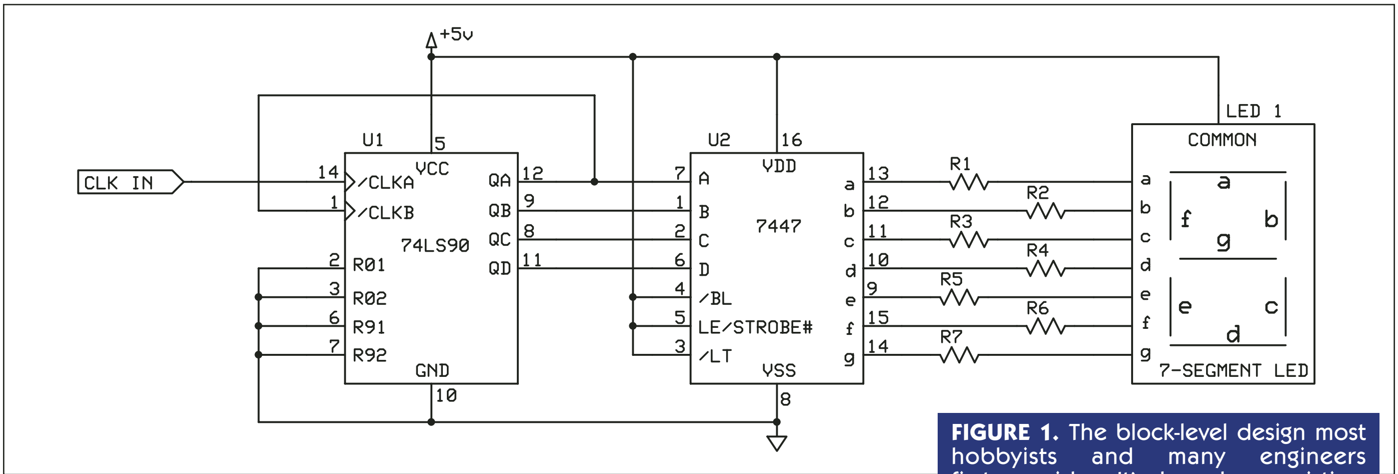


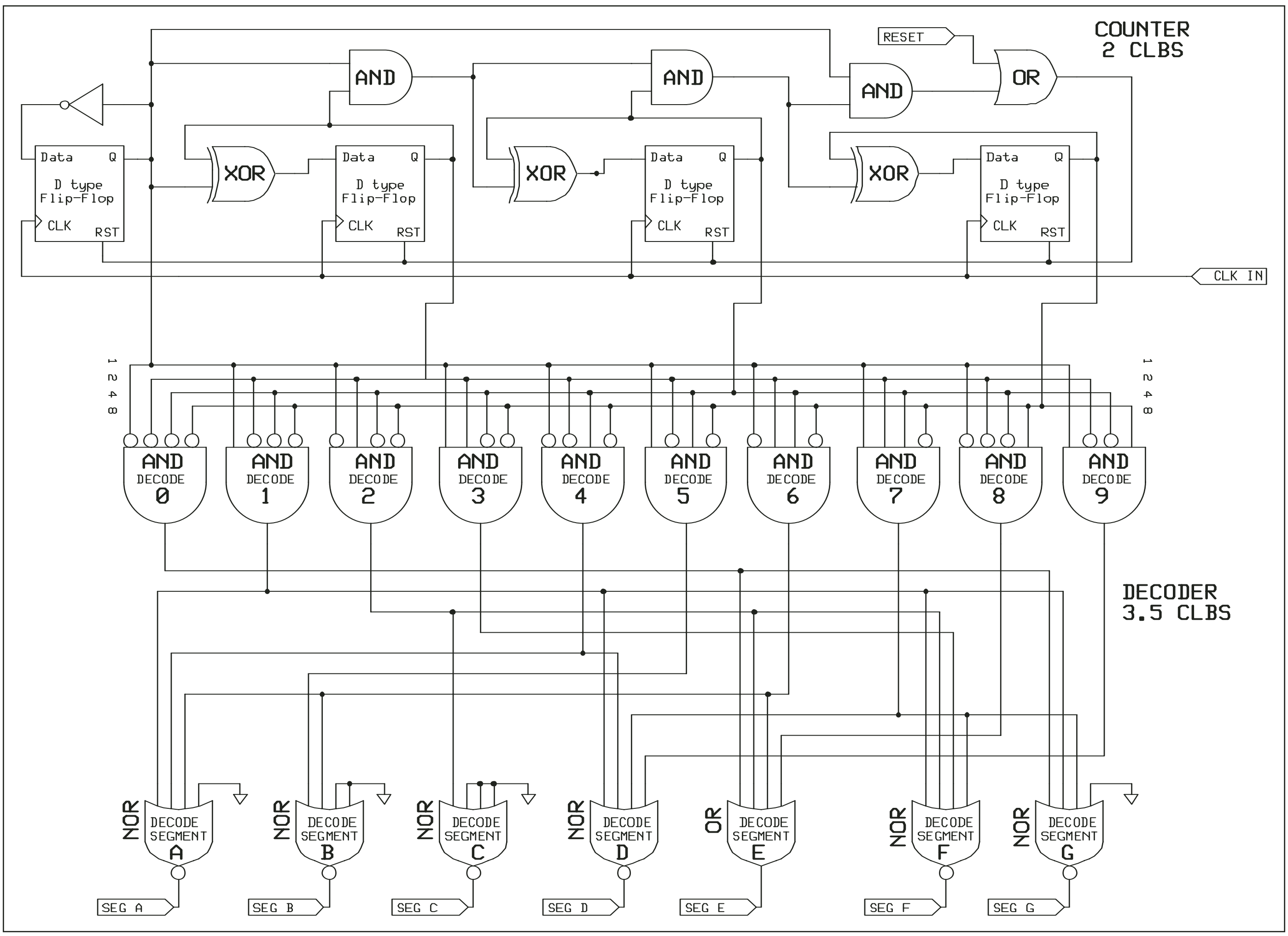
FIGURE 1. The block-level design most hobbyists and many engineers first consider. It's based on existing, low-level ICs. This design is very slow. Note that the physical positions of the seven segments are identified with the letters A through G.

really that difficult. It requires a few steps and a little thinking. But remember, reduced resources equates to smaller chips. And improved performance allows the use of slower chips. So, a little extra effort at the front end can yield a substantial savings in parts costs at the back end.

To start, the first thing to do is to define the full counting sequence that is needed. For the seven-segment counter. Table 1 illustrates the counting sequence. Figure 1 identifies the physical segment layout.

The next step is the hardest. Examine the sequence for a reduced set that defines all (or most) of the required states. Since this example requires 10 states (out of a possible 128), we know that a minimum of four segments will be needed to define these 10 states

FIGURE 2. The typical design approach uses a BCD counter followed by a seven-segment decoder. This conventional approach is relatively slow and uses a lot of CLBs.



Display Number	Segment a	Segment b	Segment c	Segment d	Segment e	Segment f	Segment g
0	ON	ON	ON	ON	ON	ON	
1		ON	ON				
2	ON	ON		ON	ON		ON
3	ON	ON	ON	ON			ON
4		ON	ON			ON	ON
5	ON		ON	ON		ON	ON
6			ON	ON	ON	ON	ON
7	ON	ON	ON				
8	ON	ON	ON	ON	ON	ON	ON
9	ON	ON	ON			ON	ON
SEGMENT ACTIVITY	7	8	9	6	4	6	7

TABLE 1. This table shows what segments are on for each digit. The physical placement of the segments is shown in Figure 1. The bottom row — Segment Activity — indicates how many times the segment is used when all the digits are displayed. This is useful in determining which segments should be incorporated in the counter.

(out of the 16 possible using four bits).

Usually, the best place to start is to search for binary patterns. Look for segments that are active for half the full sequence. Unfortunately, in this instance, there are no segments that are on for exactly five out of the 10 counts. The closest are D, E, and F which are on six, four, and six counts, respectively.

Examining these three segments shows that D-E defines only three states while E-F and D-F both define all four possible binary states (00, 01, 10, 11). Obviously, F is needed. But, should E or D be used with it? In this example, only E is on for less than five states. This aspect means that E has a better chance of resolving states (but it's no guarantee). Therefore, the first two segments are chosen to be E and F. They define the maximum number of states (4), with the minimum number of segments (2).

At this point, we can start to eliminate segments. For example, segment C is on nine out of 10 times. This segment clearly will be of little use in decoding the remaining six states. Likewise, segment B — on for eight out of 10 states — seems unlikely to provide much help. Therefore, we only have to see how three segments match up with our chosen two.

The easiest approach to this is to simply list them. This is done in Table 2. It is also helpful to add a column that indicates a "binary value." This extra column makes it easier to identify the states, however it has no relationship to the decimal value to be displayed.

Here, it is important to remember that we are shooting for four bits for the counter. These three tables show three segments/bits. It is impossible for a single bit to uniquely identify three items (the most is two). Therefore, D and G cannot be used because they create triple states. In the case of D, the triple state is the binary value of 7. For G, it's the value of 3. By elimination, our third bit has to be A (the upper left-most group). It's the only one that has no triple states. Now we have three of the four bits needed: E, F, and A. These bits provide seven out of the eight maximum states possible with three bits (binary state 4 is missing).

This table has seven states with the binary value of 1, 3, and 7 occurring twice. What is needed is a segment that separates these pairs. We see that the binary value 7 occurs in states 0 and 8. Is there any segment/bit (not already used, of course) that is different in states 0 and 8? Only segment G has this attribute. Therefore, only segment G can be considered as the fourth bit. The binary value 1 occurs in states 3 and 7. Segment G is different in these states, too. So far, so good.

The last binary value pair of 3 occurs in states 5 and 9. Unfortunately, G is ON in both of these states. This means that there is no set of four segments (starting with E and F) that will provide a fully decoded 10-state pattern. As you can see, when decoding a subset, there is no guarantee that the subset provides the needed binary patterns.

At this point, there are a number of options (giving up is NOT an option). The first is to restart the procedure with two different segments. In this case, it doesn't appear that any other choice works any better.

The second choice is to modify the closest segment and then decode the "error" and correct it with additional logic. While this may seem to be a reasonable choice, in practice, it limits

Display Number	e	f	a	binary value	e	f	d	binary value	e	f	g	binary value
0	On	On	On	7	On	On	On	7	On	On		6
1				0				0				0
2	On		On	5	On		On	5	On		On	5
3			On	1			On	1			On	1
4		On		2		On		2		On	On	3
5		On	On	3		On	On	3		On	On	3
6	On	On		6	On	On	On	7	On	On	On	7
7			On	1				0				0
8	On	On	On	7	On	On	On	7	On	On	On	7
9		On	On	3		On		2		On	On	3
				no triples 7 states				triple 7 6 states				triple 3 6 states

TABLE 2. This is a side-by-side comparison of three possible groups of segments for efficient counting. The "e-f-a" entry is the best choice because it identifies seven out of eight possible different states and has no triple state. A triple state cannot be resolved with only one additional bit.

TABLE 3. The final counting sequence to be implemented. The four columns "e-f-a-X" will form the actual counter. The four columns "b-c-d-g" will be decoded but are not part of the actual counter. The "binary value" is the counter value associated with the counter segments. Note that this sequence bears no relationship to the actual decimal value shown on the seven-segment display (Display Number).

Display Number	e (MSB)	f	a	X (LSB)	binary value sequence	b	c	d	g
0	ON	ON	ON		14 (1110)	ON	ON	ON	
1					0 (0000)	ON	ON		
2	ON		ON		10 (1010)	ON		ON	ON
3			ON		2 (0010)	ON	ON	ON	ON
4		ON			4 (0100)	ON	ON		ON
5		ON	ON		6 (0110)		ON	ON	ON
6	ON	ON		ON	13 (1101)		ON	ON	ON
7			ON	ON	3 (0011)	ON	ON		
8	ON	ON	ON	ON	15 (1111)	ON	ON	ON	ON
9		ON	ON	ON	7 (0111)	ON	ON		ON

the design. Which brings us to the third choice.

Since there has to be a bit that doesn't exactly follow a segment, let's make that bit pattern as useful as possible. For example, we see that the last three states (7, 8, and 9) are the problem patterns that occur twice (binary values 1, 3, 7). So, make these three states ON for our additional bit/segment (labeled X). This choice allows the X bit/segment to be used as a negative-going ripple-clock for a subsequent counter.

Unfortunately, it can't be 50% duty cycle (five of 10 states) because state 5 must be different from state 9. For illustration, let's make X ON for the last four states in the sequence. Our final four-bit pattern is shown in Table 3. Segments B, C, D, and G are added for convenience.

Implementing the Counter

The reason for the previous exercise was to reduce the number of variables needed to define the counting sequence. We could have kept the full seven bits and built a seven-bit counter. However, decoding seven bits takes much more resources than decoding four bits. Additionally, four bits matches the Xilinx architecture well. However, the following procedure can be used for any bit length counter of any sequence.

The procedure that we will implement is to create a counter to directly provide segments E, F, and A. The other segments (B, C, D, and G) will be decoded synchronously. The result will be a fully synchronous seven-segment counter.

We start by designing the synchronous four-bit counter portion. We want the binary sequence to be: 14, 0, 10, 2, 4, 6, 13, 3, 15, and 7. (Note, the bit weights do not match the decimal values displayed on the LED. It's easy to get confused about which bit means what. That's why I use a lot of tables and notes.) We will start with four D-type flip-flops with all four outputs returned to the CLB inputs for decoding. We use standard Invert, AND, OR logic for decoding the active states of the bits (but with bubbles on the inputs of the AND gates instead of discrete inverters).

For example, bit X (the LSB) is on for the last four states. (Remember, for a synchronous counter, decode the state previous to the desired state. See sidebar.) In this case, decode states 6, 11, 3, and 15 instead of 11, 3, 15, 7. This is easy if you set up the flip-flops from LSB to MSB. This requires four four-input AND gates (with invertable inputs) and one four-input OR gate. See Figure 3, top. Do the same for bit E (the MSB). For bits F and A, it is easier to use a NOR gate and decode those states where the bit is OFF (four states instead of six states.) Of course, with Xilinx look-up table logic, there is no speed trade-off with either approach.

The only difference for the non-counting bits (B, C, D, and G) is that the flip-flop output is not fed back into the logic. Their states are decoded in the same manner as A, E, F, and X. See Figure 3, bottom. (Note, asynchronous decoding for bits B, C, D, and G is possible. This saves four D-flip-flops, but adds an extra logic delay before the outputs become stable.) The synchronous approach allows counting at nearly the toggle rate of the flip-flops. And, as seen in Figure 3, only four CLBs (3000 series) are needed.

The design may seem at first to be very complex and use a lot more gates than the traditional approach. This isn't really true. The gates are look-up tables so the number of gates used has no actual meaning and is irrelevant. The complexity comes from combining a decoder and a counter. But if you compare Figures 2 and 3, the overall complexity is reasonably similar. The big difference comes in the effort required for the design.

Using a standard counter and standard decoder is clearly easier and faster to create. But if you choose to think and expend some effort, your device will operate faster, be smaller, and cheaper than the common approach. This is a common circumstance. Putting in mental effort early in the design stage saves a lot later.

Considerations

It's important to remember that the counting sequence is arbitrary. This means that a carry/terminal-count will not necessarily occur after state 15. In

Decoding the Previous State

The reason that the previous state must be decoded is because it takes time for the data going into the flip-flop to appear at the output. With a synchronous design, all the flip-flops are clocked at the same time. This means that feedback from the flip-flop into the decoders will not be available until after the clock occurs. For that reason, the previous state is decoded. That gives the feedback logic a whole clock period to settle and be stable for the next clock. This approach allows for very fast operation.

this case, the carry/terminal-count occurs after state 7 (but is not implemented). Additionally, resetting the counter to all zeros may not set the counter to the initial state. In this case, the initial state is 14. All zeros is the second state (in fact, it's possible to have a counting sequence without the all zeros state. In that case, be sure a reset doesn't cause an invalid state problem.) Each of these problems can be overcome with additional logic, of course. (No reset is implemented in this example but a working design would require it. It can be added without additional CLBs.)

Also remember that this example was overly difficult and contrived. In the real world, arbitrary sequence counters/pattern generators are needed because their sequence/pattern has attributes that solve other problems. As in any engineering decision, the benefits and limitations of implementing arbitrary sequence counters/pattern generators are a trade-off between various attributes.

This approach is clearly not going to be the first choice for a general-purpose counter. However, it is always useful to have an alternative solution in your bag of tricks when difficult situations arise. **SV**

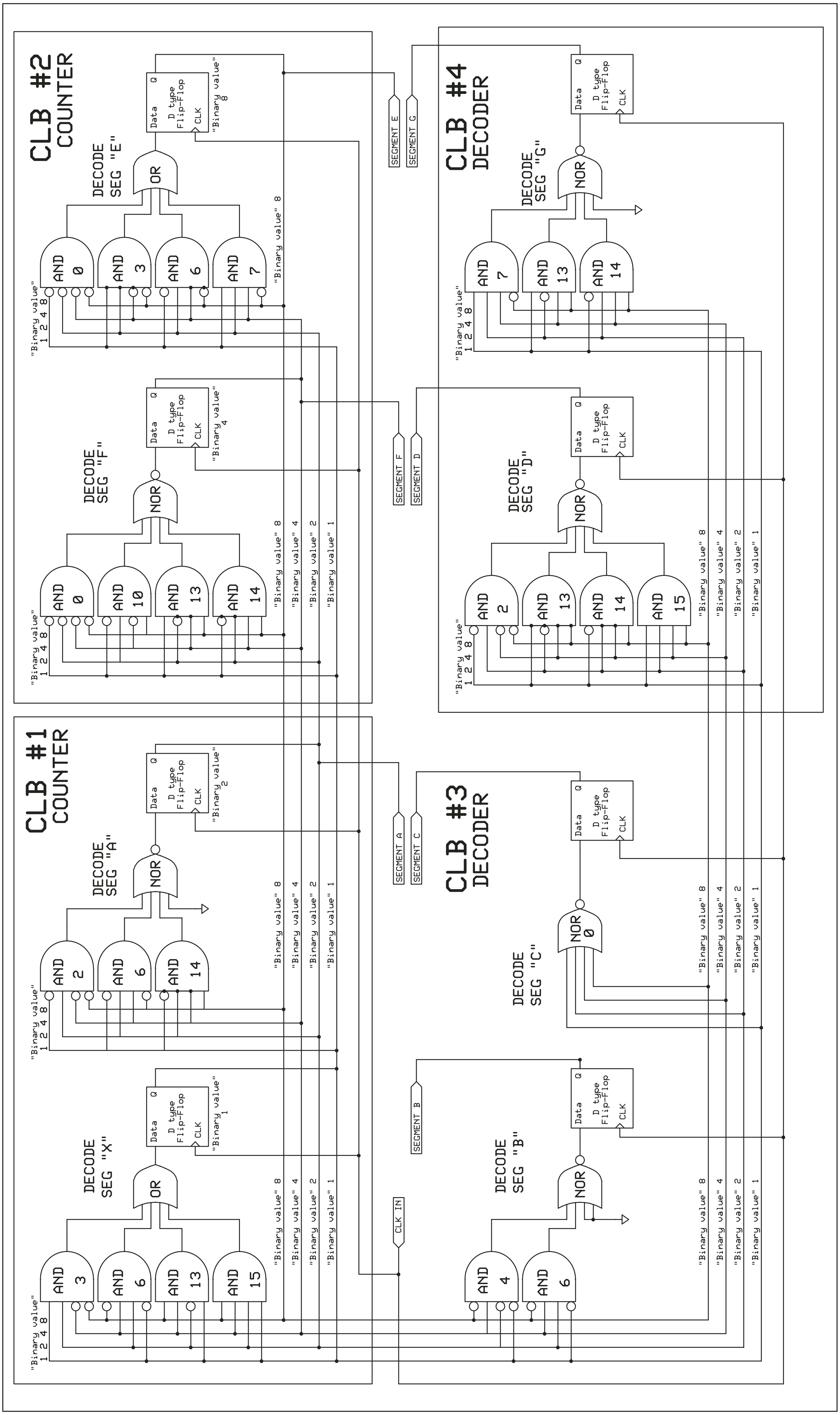
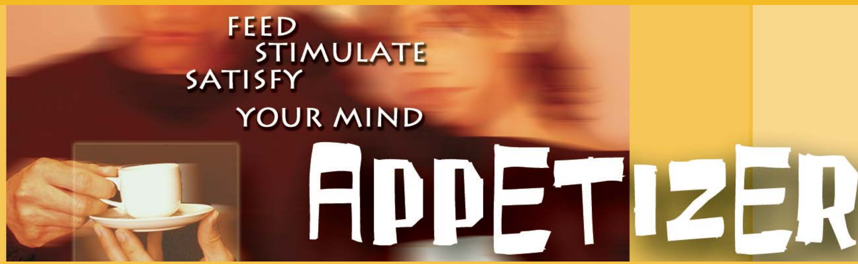


FIGURE 3. The final design for the seven-segment counter. It combines the decoding and counting for a fast, synchronous design that uses few resources. The counting sequence is completely arbitrary.



Instant Gratification is Part of the Problem, Robotics is Part of the Solution

by L. Paul Verhage

Allow me to stand on my soap box and tell you what I think the major problem with the world is. Too many people are sharing too few resources. This results in too many people who are profoundly unhappy and unhealthy. Part of the solution is to be found in lifting more of the world's population to a middle class standard of life while not endangering the long term survival of humanity and our planet. That means the solution will be found in the intelligent application of science and technology. That's engineering. Engineers are going to save the world.

In the 1950s and 1960s, American research and engineering initiated the Green Revolution. Where once farms only fed dozens of people, farmers have increased their productivity to the point where a billion fewer people are in fear of starvation. The United States graduated a lot of the scientists and engineers who brought about the Green Revolution (and other technological wonders we enjoy today).

Today, far fewer American students are graduating with these technical degrees than in the 1950s. Take a moment to look at how many of your math, science, and engineering teaching assistants are (or were) foreign students. Having this many foreign students in our universities speaks volumes about the greatness of our engineering programs. So why is a smaller percentage of Americans getting technical degrees when the rest of the world recognizes the value of an American university degree in the technical fields?

I believe part of the problem stems from our culture's desire for instant gratification (and exploited by marketing). Just like Pavlov's dogs, we're trained to respond to any marketing that promises instant gratification. Don't believe me? Then watch television. Count the number of commercials telling you happiness is found by working for long periods of time. Now count the number of commercials telling you happiness is found by wearing the right clothes, drinking the right soft drink, or driving the right automobile. Whatever the criticisms you have for our modern culture, you have to admit that its marketing sure is effective.

How are we going to turn this around? We need a motivating activity that incorporates the math, science, and engineering that we should be learning in order for Americans to save the world. It needs to incorporate software (logic), electrical, mechanical, and civil engineering. Take a guess at what hobby incorporates every one of these engineering aspects. As anyone who has built a robot knows, robotics combines the mechanical (building the robot body light and strong), the electrical (designing printed circuit boards and wiring up sensors), and the logical (writing code to integrate sensors and body). When you make your robot function in the real world, you're adding civil engineering awareness to your repertoire of skills.

Robotics not only teaches an integration of engineering, but it also teaches patience. Raise your hand if every one of your robots has worked the first time, or if your robots work so well that you'll never update them. Robotics will encourage more American students to take engineering and get them past the expectation of instant gratification.

So, I feel the way to keep America in the technological lead is to introduce as many people — especially young people — to robotics. Here's how I think robotists like *SERVO* readers can be a part of the solution: join a robotics club. And if there isn't one locally, start one. A major function of our robot clubs must be selling robotics to everyone. We need to convince potential club members that they'll get a lot out of coming to meetings. And that the time and effort they'll spend is worth their time. Give your newest members a chance to talk about what they're doing. Excitement is contagious.

You can base your club on a nicely designed robot kit, but that usually focuses people on some electrical and logical engineering (it's missing all of the mechanical and some of the electrical). Instead, you should encourage people to begin a robot at a more fundamental level — they should be encouraged to build robots from scratch. BEAM robotics and the BoRG (Boise Robotics Group) are examples of this approach.

To help your new members become robotists, develop lesson plans for the club. Document what members need to learn and how they can learn it efficiently. These lesson plans don't need to be detailed. Start by writing down what topics you want people to learn. Later fill in the learning goals with some teaching ideas. Here's a rough list of what I think we should be teaching our new members:

- Identifying components by sight.
- Identifying schematic representations of popular components.
- Knowing the typical functions of components.
- Knowing the units associated with electrical components.
- Understanding the typical range of values of components used in robotics.
- Reading the value of components.
- Using a DMM to measure voltage and continuity.
- How to solder.
- How to manipulate wire.
- How to program a microcontroller.

The BoRG is developing lesson plans and we'll be happy to share our notes. After all, the BoRG is always looking to assimilate others from around the world.

After you decide what to teach, develop incentives to keep people in your robotics club. The most popular incentives include competitions and public demonstrations. But remember, it's not going to be easy to keep people engaged since many want instant gratification. That's probably the major problem that the BoRG is facing. Our numbers fluctuate constantly. Our membership numbers go down during summer vacation and back up when school starts. We end up with a lot of new members and not as many returning members.

The BoRG is trying to reach out and retain members in several ways. First, since so many people are active outdoors during the summer (Idaho is a great outdoor state), we're moving our robotics competition to February. That's a time when there are fewer outside activities to distract people. Second, since our attendance follows the school year, we're planning to get the word out through the schools.

One underserved educational community we're trying to reach is the home schooled population. They have a network for sharing ideas with each other that we'd like to tap into. Finally, we're hoping to integrate our competitions with other school activities. Many NASA Space Grants have a Mars Rover (or Red Rover) competition. Red Rover teams design LEGO robots that perform simple planetary exploration tasks. So here's a group of potential members fresh with some mechanical engineering experience that should be ready to take the next step of combining that experience with electrical and logical engineering.

So those of you reading my editorial can help save the world by learning and teaching robotics. Good luck and please feel free to contact me with your ideas and responses (Paul.Verhage@boiseschools.org). **SV**

Then and NOW

ROBOTS WHO LISTEN

by Tom Carroll

Speech recognition systems applicable to robot use have dramatically dropped in price over the last few years. Rather than going through the myriad of computer-based solutions that we can use to make our robots listen to us, I'd like to talk a bit about why it has been so difficult to implement this listening ability.

The understanding of human speech seems to top the list of desires for robot experimenter's projects, sometimes even more than basic mobility. "Now, a robot who can listen to my voice and obey my commands; that is the starting point for an intelligent machine," you say to yourself. Speech is a human's way to communicate with others so it stands to reason that speech recognition is the most natural way for us to communicate with our robots.

This is the subject that I always enjoy talking about with other experimenters. I still like to go back to Isaac Asimov's *I Robot* series of short stories and the story — Robbie. Young Gloria Weston was given a robot — Robbie — as a babysitter and companion. This mute robot would quietly sit next to Gloria as she told him stories. She frequently would give him numerous verbal commands in her childish ways, which he would quickly obey. Her mother quickly tired of the robot and had her father send it back to the factory without Gloria's knowledge of what they had done. One day, while visiting a museum, Gloria was transfixed by the world's first talking robot and began to ask it "Mr. Robot, Sir. Have you seen Robbie?" Of course this robot had no clue about what she was asking and was about to 'blow a circuit' when the operator ran up and told the gathering crowd that they could not talk to the robot without an attendant.

For those of you who know Asimov's stories, this is where young Susan Calvin is introduced as a student, quietly taking notes on the robot and the spectators. She later became Dr. Calvin, a robopsychologist for US Robots and Mechanical Men, Inc. Later in the story, Robbie would save Gloria's life and gain acceptance by all the Weston family members.

What I've always found to be interesting is how robot speech came after speech understanding for robots in Asimov's stories. When he wrote the first story about Robbie in 1940, it was entitled *Strange Playfellow* and published in *Super Science Stories*. Crude "robot" speech was already being investigated, yet true machine speech understanding was still a long ways in the future.

I assume this speech understanding was a must for Asimov's robots to be able to react to his "second law of robotics" — *"A robot must obey the orders given it by human beings except where such orders would conflict with the first law."* (The first law prevented a robot from injuring a human or allowing a human to be injured.)

What is Speech Recognition?

The terms, speech recognition or voice recognition have always bothered me as they really do not imply the features we actually desire for our robots to possess, but that's the screwy English language that we use. As I've mentioned in other articles, I can "recognize" Russian speech and, in fact, actually recognize that it is the language that Russians use, but I don't have a clue what Russians are saying. A dog can recognize its master's voice and come running with tail wagging when it hears him say "hey, boy, ya wanna go to the vet and get neutered?"

We really cannot use the term that I use sometimes — speech understanding or cognition. Well, enough of that. We cannot even come to a unified definition of a robot, so how are experimenters supposed to decide on how to describe how robots listen?

A speech recognition system installed on a computer can identify each word through a complex set of algorithms and print them out in a sentence, but few computers available to experimenters actually "understand" what the line of words mean or the context in which they are used. We just program into the microprocessor or stand-alone speech recognition board that the words "go right" triggers another line of code to make the left motor (in a differentially-driven robot) turn more revolutions than the right motor. Well, it's really not quite that easy, but that's the basic principle.

The bottom line is: Speech recognition sounds a bit more applicable to a computer understanding commands given to it, as "speech" refers to a series of words that imply an idea, command, or meaning. "Voice" recognition can refer to just the sound of a person's voice or a single word triggering the computer. Though many magazines and companies

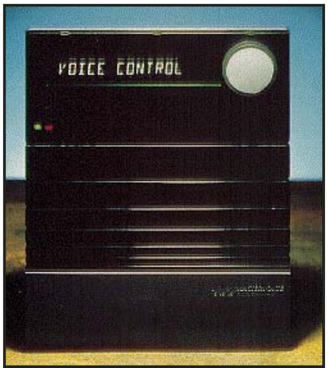


Figure 1. Butler in a Box.

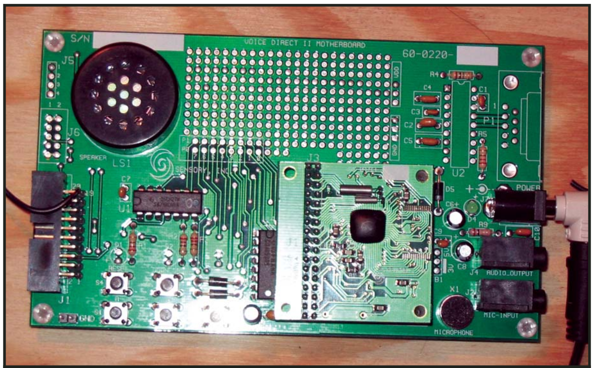


Figure 2. Voice Direct II.

use the two terms interchangeably, I'll go along with speech recognition for this article. Besides, speech recognition had 19 million hits on Google, vs. 11 million for voice recognition.

The Nuts and Bolts of Speech Recognition

Speech recognition is the ability of a machine or program to receive and interpret words spoken to the system, or to understand and carry out spoken commands. Applications can vary anywhere from dictation (office steno, voice input word processing, court reporting, etc.), control of machines (industrial, medical, battlefield, etc.), telecommunications (cell phones, telephone verification messages, etc.), and, of course, robots (home, industrial, etc.). There are numerous other applications and many are applicable to physically disabled persons to assist them in daily living.

The bane of all speech recognition systems has always been background noise (machines, others talking, barking dogs, etc.) and variations in the human voice. Directional or closely held microphones and filters help clarify the input voice signals to the computer. For use with computers, the audio signal from a microphone or audio source must be converted into digital signals by an analog-to-digital converter.

If a computer is to understand the speech input, it must have a database or vocabulary of words or phonemes and a rapid way of comparing this data with the input. It is at this point where we can divide the speech recognition systems into speaker independent and speaker dependent systems.

Higher-end speaker independent systems can be so designed that anyone can enter a voice command, word, or phrase and the computer can understand them. A speaker dependent system requires a voice template (or templates) of a single person's voice speaking the required vocabulary words. It is sometimes advantageous to have a computer or robot that can verify the speaker and be controlled only by that person. This variation is called speaker verification.

The most complex system is continuous listening of speech without having to press a button to start a period of listening. The reference speech patterns can be stored on a hard drive for a computer-based program, or static RAM or Flash memory for a stand-alone board. A comparator checks these stored word or phoneme patterns against the output of the A/D converter and makes a determination of what word was spoken into the microphone.

Early Speech Recognition Systems for the Experimenter

The earliest "speech recognition" that I remember for home experimenters was not really speech recognition at all, but was just "syllable counting." One of my early robots used this method for fairly crude control. Quite a few toy vehicles have also used this system that converts spoken word syllables into pulses to drive relays for control.

I happened to have a bunch of 12V relays and some telephone office stepping relays that my brother had given me. The huge telephone relay, the amplifier and microphone, and other relays almost doubled the size of the robot and I had to almost shout the commands: "Stop! Now go right! Go left!" etc. If I managed to screw up the three syllables "Now go right" and said instead, "Now go left," the robot would count the three syllables and still go right. My friends quickly found that watching paint dry was more exciting than watching my stupid robot go in the wrong direction.

Microprocessor Based Speech Recognition

Since there have been so many manufacturers of speech recognition systems, including main frame, personal computer based, and stand-alone board level units, I'll concentrate on the systems available to the experimenter. The government, military, and many universities have long been experimenting with systems for their particular purposes, yet the greatest breakthroughs came with inexpensive experimenter's units.

Back in the early '80s, many of the new 6502, Z-80, and 8080-based personal computers had manufacturers designing speech synthesis and recognition stand-alone boards for experimenters. Steve Ciarcia featured the Lis'ner 1000 speech recognition system in his popular *Ciarcia's Circuit Cellar* column in the November 1984 *Byte Magazine*. The Lis'ner 1000 was a low-cost ($150), high-performance speech-recognition system for the Apple II or any 6502-based system.

After experimenting with the Votrax SC-01 speech synthesizer for a while, I was given a Lis'ner 1000 by my friend, Dave Freemen, co-founder of Advanced Computer Products in Santa Ana, CA. It was a bit cantankerous to program and use but it finally made my robots seem a bit more human, at least more human than the telephone relay cheating thing. I first used a KIM-1 single board computer and later, an AIM-65 board as the controller. Now, instead of my robots leaping to their death from my workbench when I turned on the power, at least they learned to wait for my verbal command before taking the unexpected death dive. (I later discovered that the set of "H" bridges that I designed myself occasionally went from zero output to full output when a signal was applied. Using a friend's design corrected the problem).

ViaVoice from IBM and Dragon System's Naturally Speaking are two software-based systems that became available to experimenters a bit later that improve over time as the computer learns and adapts to individual speaker's speech patterns. These two companies have further refined their products into Dragon Dictate and "VoiceType" from IBM, and Apple Computer also has a speech recognition program called Voiceprint, all for speech-to-typed-word applications. Say goodbye to the stenographer.

Butler in a Box

One of the first out-of-the-box fully integrated systems that I ever got a chance to experiment with was the Butler In A Box from Mastervoice, now called AVSI, Inc. Butler In A Box was created by founder and company president Gus Searcy, a professional magician. I invited Gus to one of our Robotics Society of Southern California meetings in the mid '80s to demonstrate his new speech-controlled home automation device. All of the RSSC membership voted this device the coolest thing that they'd ever seen. "A magician shouldn't have to get up to turn the light on," he said. "As a result, I decided to create the illusion of an invisible magic genie."

Figure 1 shows the large dictionary-sized standalone automation computer that uses a smart controller and X-10 compatible devices. You can program the system by using a keypad on the box. There are several Butler In A Box models ranging in price from $1,795 to $3,995. These aren't cheap puppies but the 'cool factor' is way up there.

Sensory Speech Recognition Products

To really get a handle on the latest speech recognition systems available to robot experimenters, I spent quite a while on Google checking out manufacturers, comments, about systems, and which ones were more applicable to robot experimenters. I was somewhat familiar with Sensory's Voice Direct II $49.95 speech recognition board that features continuous listening and recognition technology and allows almost any device to be controlled with just the sound of one or two key words or a short phrase (see Figure 2).

It then listens for up to three seconds to recognize one of up to 15 additional "command" words or phrases lasting up to 2.5 seconds each. When a command word is recognized, the Voice Direct II will raise one or two output pins high for one second, which can be used to control external devices. All the trigger and command words are speaker dependent, so the recognition technology will work for any language.

It can also be configured to have one to three different continuous listening trigger words or phrases, each with up to five speaker-dependent command words. The University of Vermont's College of Engineering and Mathematical Sciences used this system in an experimental van for the physically disabled where the driver could verbally command various functions such as windshield wipers, lights, etc., without removing a hand from the steering wheel.

The VR Stamp

Recently, the Dallas Personal Robotics Group had a series of responses on its website to a discussion on Sensory's VR Stamp, available for $39.99 at Digi-Key and other places. Figures 3 and 4 show the 40 pin DIP configuration and the block diagram of the VR stamp and the on-board microcontroller. After reading the different replies, I decided that I had to have their VR Stamp Toolkit to develop a reliable speech recognition system for a robot I'm working on. The kit comes with two VR Stamp modules and a programmer board with a built-in microphone. Also included is a VR Stamp Toolkit CD-ROM with a lot of useful documentation and speech tools, a serial-USB cable, speaker, and a wall-wart power supply.

I was a bit disappointed that there was no printed documentation with the kit as I had to search back and forth on the CD-ROM and print out what I needed, but there is an amazing amount of information on the CD needed to set up the system and for other applications. Besides basic information guides in Adobe Acrobat format, there are four software installations that you can use for different applications and the needed drivers.

A quick connection to the 9 VDC wall-wart power supply, the included speaker and the USB cable to the computer and the system was up and running. The 40 DIP zero insertion force socket on the VR Stamp Programmer board allows a quick interchange of two programmed VR Stamps. The 3-1/4" x 4" programmer board size is convenient for mounting on a robot, though the 40 pin DIP Stamp and an associated microcontroller is all you'll really need after programming the Stamp. I found the VR Stamp to be quite accurate, though pronunciation and word speed and separation helped the accuracy a bit.

Figure 3. 40 pin DIP configuration of the VR Stamp.

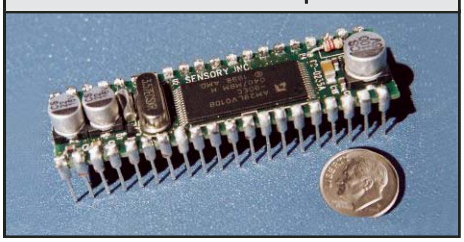
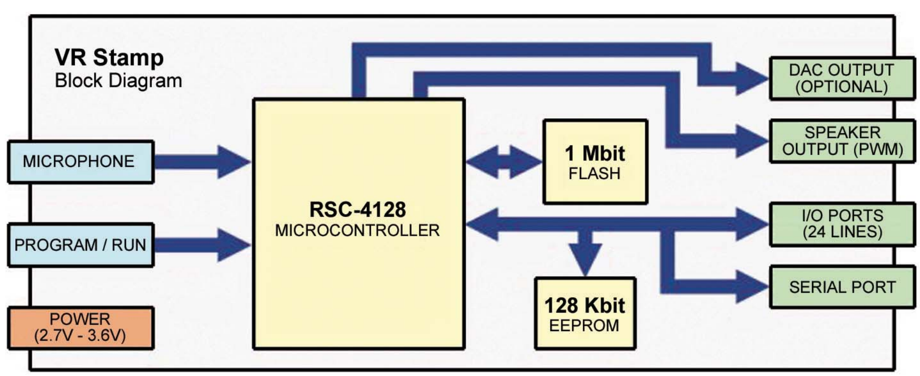


Figure 4. VR Stamp block diagram.



RESOURCES

Sensory Systems
www.sensoryinc.com

AVSI Automated Voice Systems
www.mastervoice.com

Dragon Systems
www.speechtechnology.com/dragon/

IBM (and Dragon)
www.voicerecognition.com

Another company associated with Sensory and their VR Stamp is a company based in Belgrade (formerly Yugoslavia) — mikroElektronika. It was established in 1997 as a publishing firm specializing in electronics, and has become well known for PIC, AVR, 8501, and other microcontroller development tools. They also make the EasyVRStamp development system board for the VR Stamp voice recognition modules. It was designed for students and engineers to explore the capabilities of VR Stamp voice recognition modules. The development board's $129.95 price includes a $40 VR Stamp — a real bargain for those who want to start experimenting with speech recognition.

I have just touched upon this complex and exciting part of the new robotics age. Speech recognition, like most areas of electronics, is making rapid advances. One day soon we'll be able to tell our robot "Robbie, go deep and catch this forward pass." Of course, we'll need to work a bit on the mechanics, but true speech cognition is definitely on the near horizon for robot experimenters. Enjoy talking with your new friend who will gladly listen to you. **SV**

ADVERTISER INDEX

THESE ARE THE REAL DEAL
SerialLCD™
2X16 and 4X20
Serial Displays
RS232 and TTL Inputs
9600/2400 Baud
254 Backlight levels
RoHS Compliant
Input Serial Data
Normal/Inverted
Outputs for
Buzzer or Relay
Easy to use!
Priced from $34.95
Complete info
at BasicX.com
COLOR MAKES THE ORDINARY... EXTRAORDINARY!
NetMedia
NetMedia Inc., 10940 N Stallard Pl, Tucson Arizona 85737 tel.520.544.4567 fax.520.544.0800 www.BasicX.com