Build and Program a "Minirobot," p. 38 • Write Your Own Puzzle-Solving Programs, p. 62 • Embedded Design Security, p. 68

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#236 March 2010

ROBOTICS

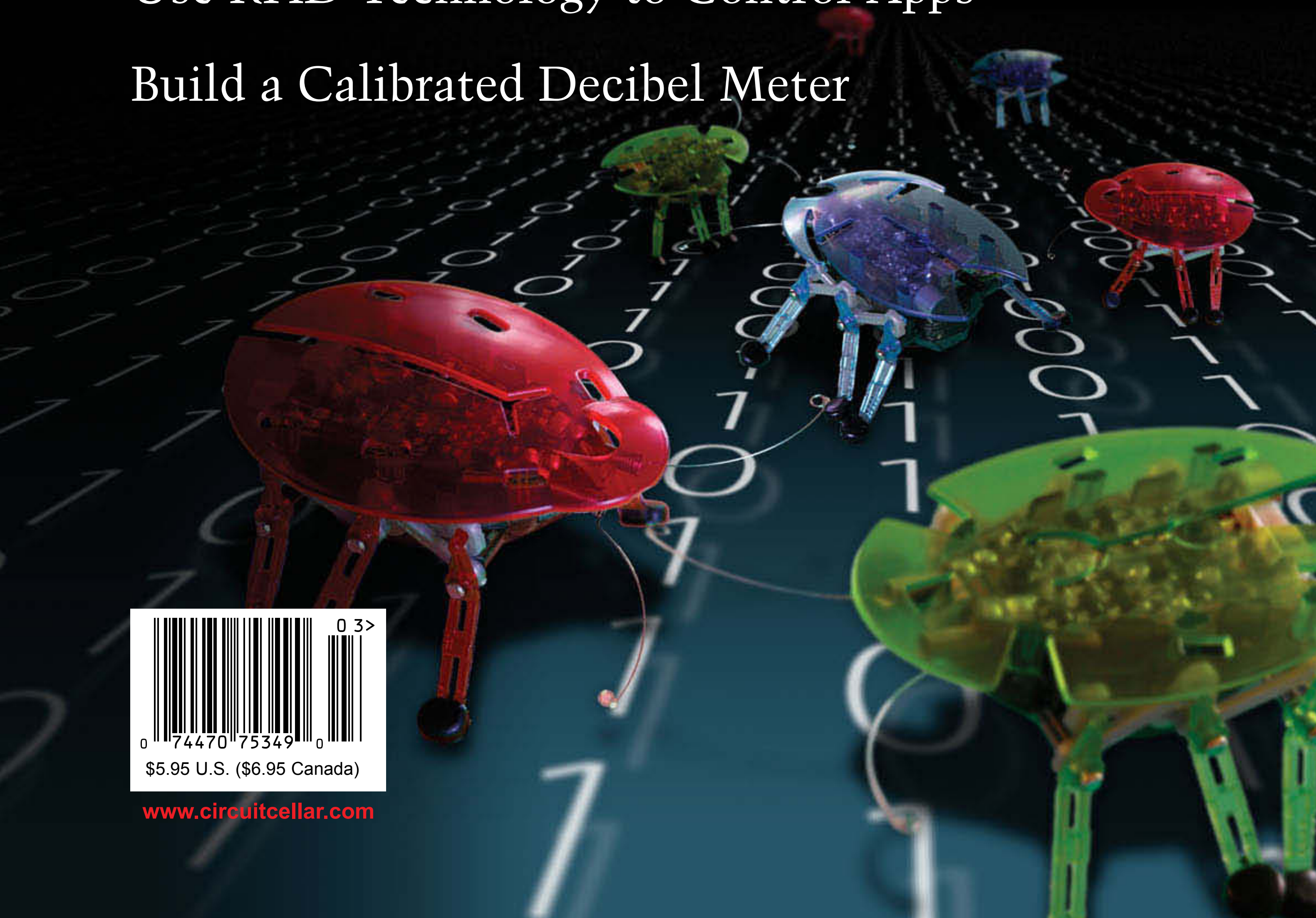
An Environment-Sensing System for Mobile Robots

A Serial Network Hub for an Animatronics System

Directional Light Sensor Design: Phase 2

Use RFID Technology to Control Apps

Build a Calibrated Decibel Meter



0 74470 75349 0 03>

$5.95 U.S. ($6.95 Canada)

www.circuitcellar.com

TASK MANAGER

Need-to-Know Info

Our editorial mission is to provide serious embedded designers, programmers, and academics with essential "need-to-know" information that will enable them to stay on the cutting edge of technological development. The information we provide enables project engineers to bring new design concepts to their company's design meetings, helps entrepreneurs deliver new MCU-based technologies to the market, and prepares programmers to write more efficient code. You might not want the same sensor system Guido Ottaviani describes on page 14, but you'll take what you learn and apply it to innovative projects of your own. You probably don't need to build a replica of Brian Millier's RFID-based liquid control system for a nitrogen tank (p. 24). But if you understand why and how he designed it, you'll be a step ahead of engineers who don't have the knowledge to build a custom RFID control system of their own.

The main point is simple: When reading a *Circuit Cellar* article, it isn't always the end product that matters. What's most important is what happens from the time an idea is hatched to the time the project first powers up. You won't find our complex embedded design content in trade journals and hobby magazines. Those publications won't enable you to design a truly novel application that will make you money. Nor will they give you the in-depth information you'll need when your project manager says: "I need a solution to this design problem. And it's due ... yesterday." Nor will any hobby projects, once finished, move your resume to the top of the pile on a potential employer's desk.

This month's articles cover many of the complicated topics that you need to master in order move ahead and excel at your workbench, at your job, or at your university: robotics, embedded security, audio output measurement, networking, RFID technology, MCU-based sensor system design, and more. To learn about networking a complex animatronics system, turn to Peter Montgomery's article on page 30, where he explains why building a six-port RS-485 hub for a custom network packet system can simplify node wiring. On page 38, Marco Aiello describes how to design and program a "minirobot" along with a handy navigation system. If you're an audio enthusiast, Larry Cicchinelli's article on page 46 presents a calibrated decibel meter for measuring audio output levels. On page 54, Jeff Bachiochi describes the last phase of his sun tracker design project, which involves time keeping and displaying data on the LCD. Turn to page 62 to learn how George Martin created a puzzle-solving program in C. Finally, check out Tom Cantrell's article on page 68 about notable advances in embedded design security.

With all of this information packed into one issue, your "engineering quotient" is sure to rise. Absorbing need-to-know information is the best way to stay one step ahead of the pack.

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
C. J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Gary Bodley
Ken Davidson
David Tweed

PUBLISHER
Hugo Vanhaecke

ASSOCIATE PUBLISHER
Shannon Barraclough

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

STAFF ENGINEER
John Gorsky

ADVERTISING
800.454.3741 • 978.281.7708 • www.circuitcellar.com/advertise

ADVERTISING REPRESENTATIVE
Peter Wostrel
Strategic Media Marketing, Inc.
2 Main St., Gloucester, MA 01930 USA
800.454.3741 • 978.281.7708
peter@smmarketing.us • www.smmarketing.us
Fax: 978.281.7706

ADVERTISING COORDINATOR
Valerie Luster
E-mail: val.luster@circuitcellar.com

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com

PRINTED IN THE UNITED STATES

CONTACTS

SUBSCRIPTIONS
Information: www.circuitcellar.com/subscribe, E-mail: subscribe@circuitcellar.com
Subscribe: 800.269.6301, www.circuitcellar.com/subscribe, Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650
Address Changes/Problems: E-mail: subscribe@circuitcellar.com

GENERAL INFORMATION
860.875.2199, Fax: 860.871.0411, E-mail: info@circuitcellar.com
Editorial Office: Editor, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: editor@circuitcellar.com
New Products: New Products, Circuit Cellar, 4 Park St., Vernon, CT 06066, E-mail: newproducts@circuitcellar.com

AUTHORIZED REPRINTS INFORMATION
860.875.2199, E-mail: reprints@circuitcellar.com

AUTHORS
Authors' e-mail addresses (when available) are included at the end of each article.

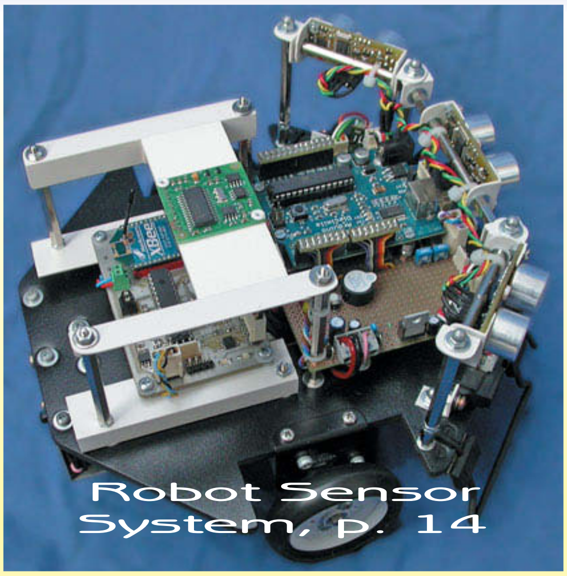
CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $29.95, Canada/Mexico $34.95, all other countries $49.95. Two-year (24 issues) subscription rate USA and possessions $49.95, Canada/Mexico $59.95, all other countries $85.** All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank. **Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call 800.269.6301.**

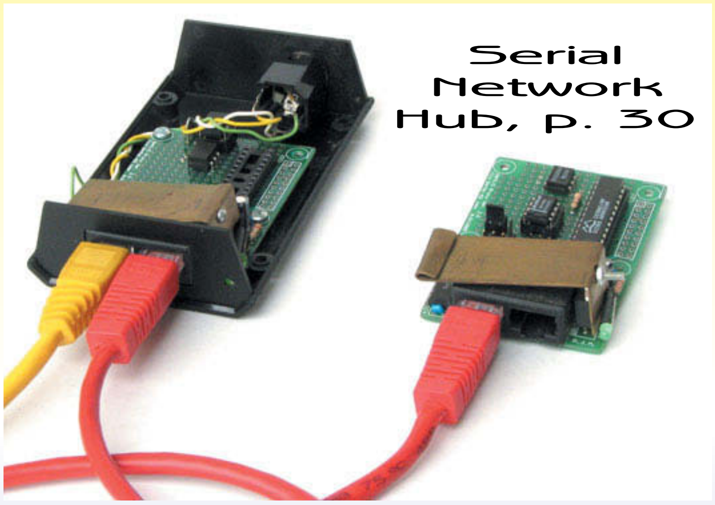
Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755-5650.

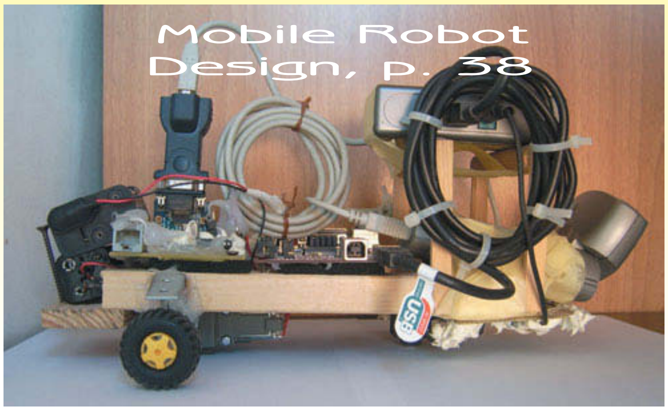
INSIDE ISSUE 236

BONUS CONTENT
The Arduino-Based "MiniEric" Robot

March 2010 • Robotics







by Guido Ottaviani

A Sensor System for Robotics Applications

If you're developing an interactive robotics platform, you'll have to incorporate a well-designed sensor system at some point. This article covers how to add "senses" to a robotics design, from sight to hearing to touch and more.

In my 2009 article series titled "Robot Navigation and Control," I explained how to build a navigation control subsystem for an autonomous differential-steering exploring robot called the "Rino" (*Circuit Cellar* 224 and 225). I covered everything from the mechanical platform to the hardware and software used for guidance and system orientation. During the last several months, I upgraded the system with a new sensor system that enables obstacle avoidance and location recognition (see Photo 1). In this article, I'll describe how I designed and implemented the sensor system. Read on to learn how to add senses like sight, smell, and sound to your next robotics application.

ARDU-RINO

The first step in the design process for the updated Rino robot was selecting the right controller. With Microchip Technology PIC experience under my belt, I figured it would be more exciting to try something else—so long as it wouldn't require too much programming effort.

I knew an 8-bit MCU was more than enough to drive the sensors normally used in amateur robotics, because sensors I'd usually used had a response time measured in terms of several milliseconds or more. While developing the remote console that I described in my 2009 articles, I learned a lot about the Arduino hardware and software platform. It's inexpensive and powerful. (And it's developed in Italy. I'm a bit patriotic.) The best thing is that Arduino is growing in popularity all over the world. It's completely open-source, so it's easy to find any kind of hardware interface or software library you need. You have the option to use an off-the-shelf piece of software or build your own library using C programming and even controlling every single bit of the included Atmel microcontroller. For all of these reasons, I chose to use an Arduino Diecimila board—and so the "ArduRino" was born (see Photo 2).

The Arduino's specifications include the way the expansion boards must be implemented. They are called "shields."[1] But I didn't follow the standard. I used a perfboard to build my own version of a shield (see Photo 3). The sockets on the perfboard are for interfacing the Arduino with the

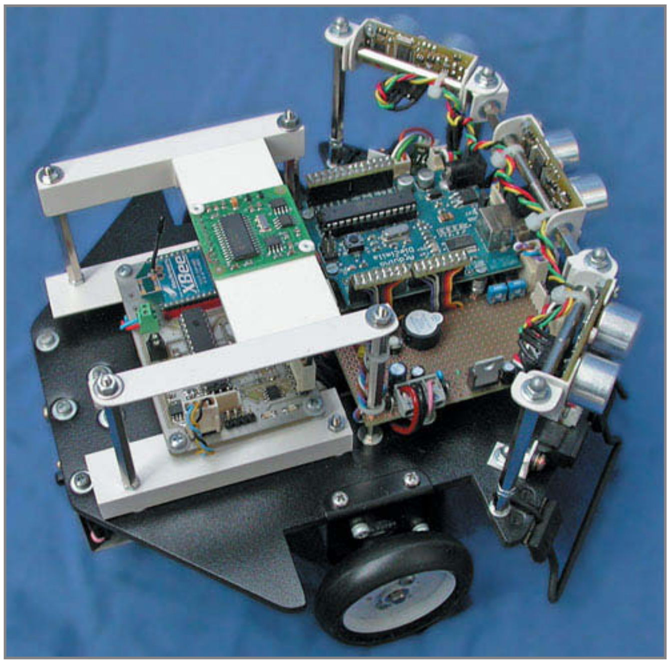


Photo 1—My upgraded robotics platform features a new sensor system that enables obstacle avoidance and location recognition.

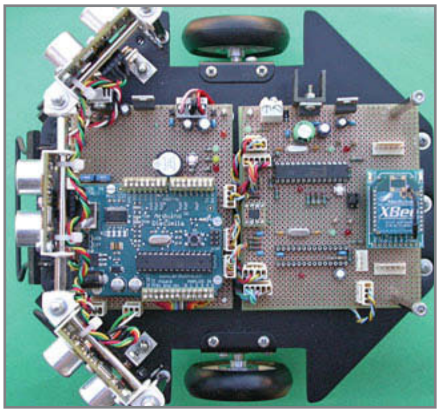


Photo 2—This is my robot's sensor board with most of the necessary devices installed.

rest of the components and boards. Among them are some signaling LEDs, switches, and a Maxim MAX127 data acquisition system (i.e., eight analog-to-digital channels with an I^{2}C interface). This is also useful for expanding Arduino analog ports for future applications. With the "Wire" library for Arduino, it's easy to use the I^{2}C bus with just few lines of code.[2] Sample code to control the MAX127 is also available online. A couple of low dropout voltage regulators on the board supply the Arduino and Figaro TGS822 gas sensors (see Figure 1). A component layout of the shield is posted on the *Circuit Cellar* FTP site.

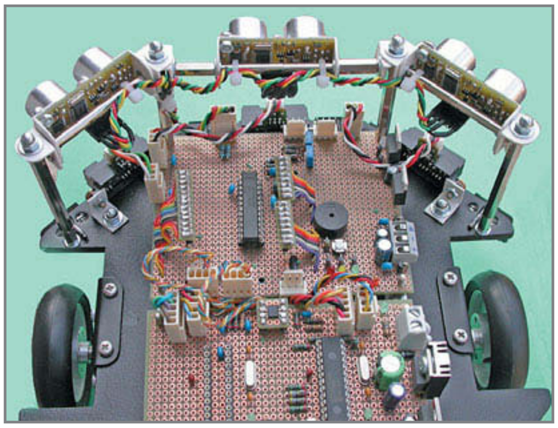


Photo 3—I have my own version of a shield for the Arduino.

SMELL

A gas source (e.g., slowly evaporating alcohol) is a frequently used "target" in many robotics competitions. The TGS822 gas sensor has high sensitivity to organic solvent vapors such as ethanol (see Photo 4). It includes a semiconductor element that, in presence of detectable gas, increases its conductivity in proportion to the gas concentration in the air. The sensor works linearly only at a specific temperature. To obtain that, the sensor contains a heater that warms up the semiconductor. This heater is the most power-consuming part of the entire sensor. For this reason, a dedicated voltage regulator is used on the sensor board. It requires some time after power-up to begin working linearly.

On the board in the lower part of the platform are two gas sensors to increase the range of detection (see Figure 2). The signals from the sensors are mixed in one output only with two diodes and connected to an ADC input on the MAX127 IC.

The highest voltage level between the two sensors breaks the diodes' threshold and goes to the A/D port. In fact, we don't need to know which sensor is involved in measuring. This configuration saves one port of the converter. When the level crosses a certain threshold, the robot stops and illuminates a red LED to show that the goal is reached.

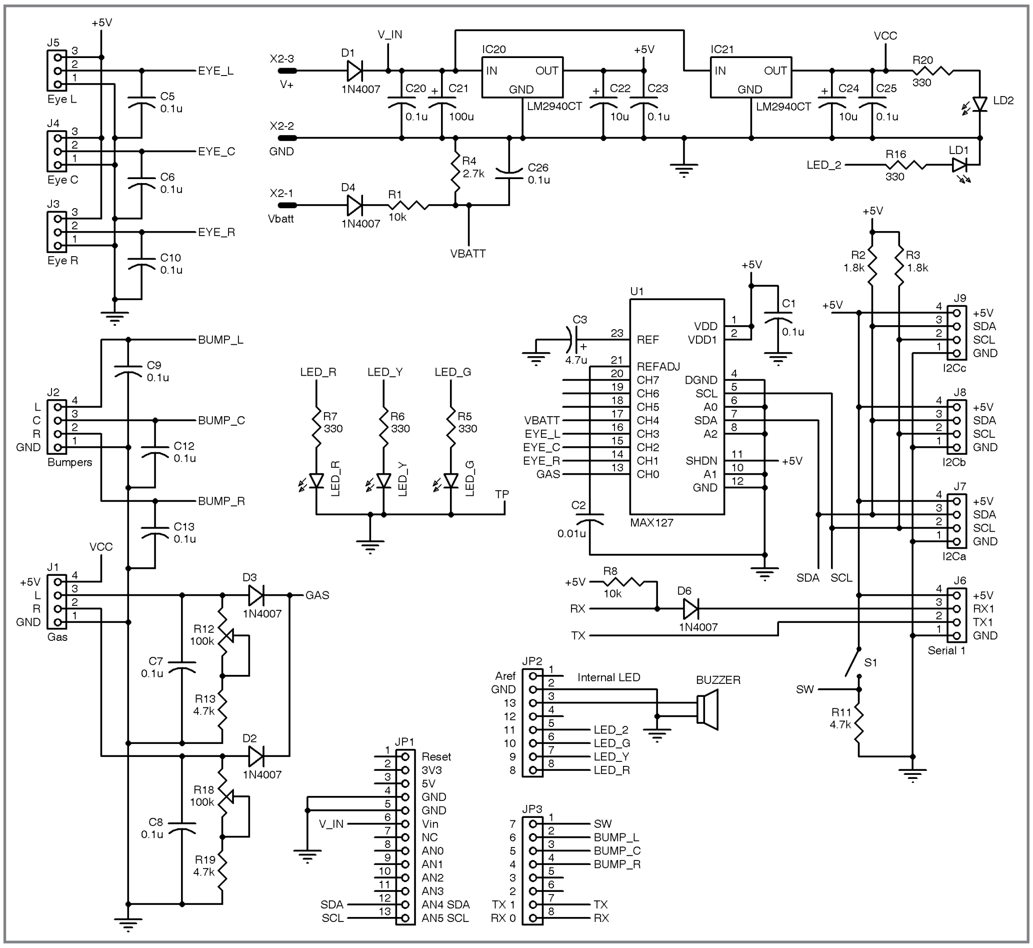


Figure 1—The board is based on an Arduino. Thus, most of the circuits serve to connect and adapt various sensors to the Arduino's ports. The only active component is the MAX127 A/D-to-I^{2}C expander.

SIGHT

Providing a robot with the sense of sight enables it to avoid obstacles and detect targets. Different technologies (e.g., as ultrasonic sight, infrared, and visible light) provide you with different object-detection options (see Photo 5). Each technology has its pros and cons.

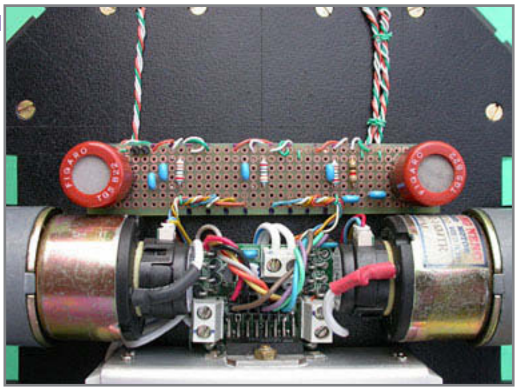


Photo 4—Figaro TGS822 gas sensors are installed on the bottom of the robot.

Ultrasonic sight is what bats use to see. It involves waiting for the echo of ultrasonic waves reflected by objects. My design features three Devantech SRF08 ultrasonic range finders that are driven through the I^{2}C bus (see Photo 6a). With a protocol similar to the one used to read and write a 24CL*xxx* EEPROM, they can be programmed and read.

Using ultrasonic waves at 40 kHz with a good range of measurement is achieved, but with a large beam width. This is useful for detecting obstacles with a limited number of sensors, but not to map the object in its exact coordinates. Something better can be obtained with 200-kHz ultrasonic sensors, but at a higher price. There are several different mathematical and statistical methods for decreasing uncertainty, such as the virtual force field (VFF) method and the vector field histogram (VFH) method. (For more details, refer to the documents and website listed in the Resources section at the end of this article.)

For infrared sight, I used three Sharp GP2D120 analog distance-measuring sensors (see Photo 6b). This kind of sensor uses an array of receivers to compute the angle of the infrared beam after the reflection on the object. The internal circuit returns a voltage proportional to the distance with a transfer function specified in the datasheet. They are cheaper and smaller than ultrasonic sensors, but the measuring range is limited. For this reason, there are different models for different ranges. The GP2D120 is the best option for this application because it starts from 4 cm, which enables the robot to detect nearby objects. This is fine for mapping the objects with precision, but there is a risk of missing thin obstacles. It works well with mat surfaces. But shiny or glossy objects can be problematic because the IR beam can be reflected away from the sensor. For instance, I have a cylindrical umbrella stand in my house that's similar to a large metallic can. The robot can't avoid it with only the IR sensors. Furthermore, IR sensors can go blind or return false measurements if an intense light (even in visible range) hits them directly. The analog output of the sensors is read through three ports on the MAX127 ADC to I^{2}C chip. I can use the same bus used for SRF08 sensors.

Another target in many explorer robot competitions is a light source.

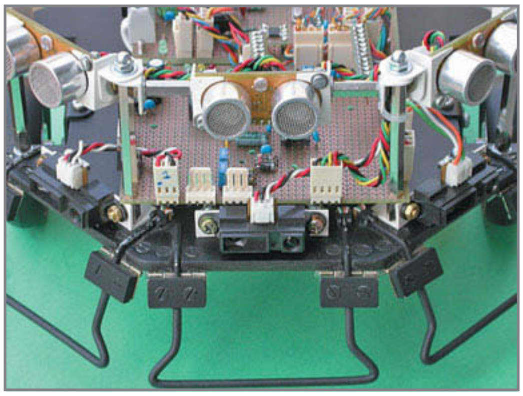


Photo 5—Most of the sensors—range finders, IR distance measuring sensors, mechanical bumpers, and CDS photoresistors—are located in the front. All of them are replicated on three sides with an angle of 45° to expand the system's range of sight.

The goal is for the robot to detect the light source, stop a predetermined distance from the light, and then illuminate a green LED. To do this, the robot must measure the distance from the object and detect the light. When the light intensity breaks a predetermined threshold and the distance is less than 20 cm, the robot initiates its "job done" procedure.

Note that a CDS photoresistor is already installed on the SRF08 module. It can measure the ambient light intensity and put the value in a register that's readable through the I^{2}C bus in the same fashion as distance values.

TOUCH

The bumpers I added to the design are intended to detect objects that come into contact with the robot (see Photo 7a). They are useful particularly if the other sensors fail or if an object can't be detected for some reason.

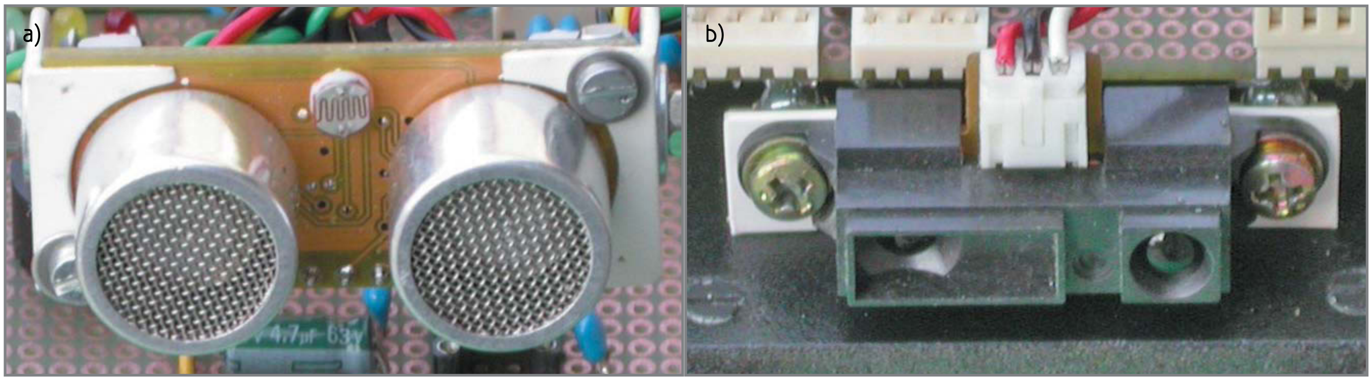


Photo 6a—The SRF08 module contains a complete circuit that handles the procedures for measuring distances between 3-cm and 6-m with ultrasonic waves. b—The GP2D120 infrared module can accurately measure objects from 4 to 30 cm.

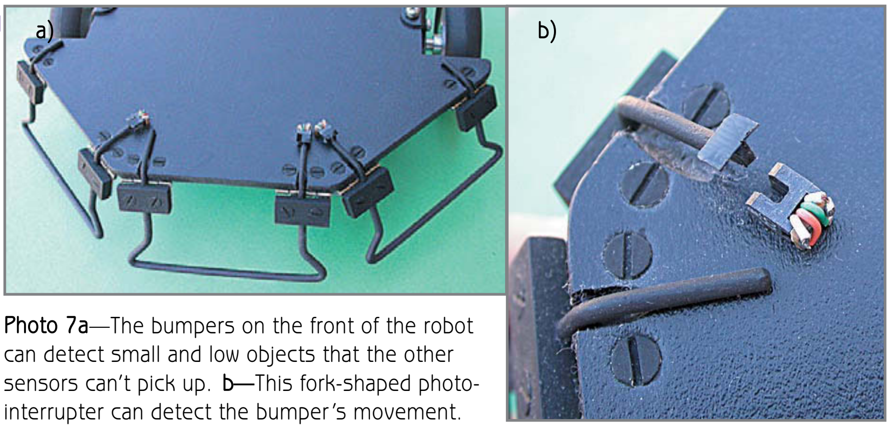


Photo 7a—The bumpers on the front of the robot can detect small and low objects that the other sensors can't pick up. b—This fork-shaped photo-interrupter can detect the bumper's movement.

The bumpers are based on fork-shaped photo-interrupters (see Photo 7b). When the lower part of the bumper touches the object, the top part rises and allows the IR light beam to reach the photo-transistor and close the circuit. It needs just a little touch to move the barrier enough to close the circuit. After that, the bumper can still move back a few centimeters so the robot can smoothly decrease its speed to zero without blocking the motor.

SOUND

Sounds are also common targets in robotics competitions. For instance, let's say a speaker in one of the course's boxes emits a 4-kHz tone. Once sounded, the robot must sense the sound, stop close to the box emitting the sound, and illuminate a yellow LED.

A simple method for recognizing a tone is to use microphones to receive the sound, amplify it, and then filter it. You can use a tone decoder to trigger the controller when the signal exceeds the threshold. This requires the robot to travel randomly until it comes close to a speaker.

A more sophisticated sound-detection method uses an "electronic ear" to detect the direction from which the sound originates. The setup requires a couple of microphones at a distance of one wavelength (about 8 cm) to each other. You then must perform some math with Microchip dsPIC DSP libraries to measure the phase difference between the two signals. But this method is beyond the scope of this article.

I'll focus on the "simple" solution. I used it for one of my previous robots, which had some success in robotics competitions. It isn't easy to detect a single tone in a noisy environment (with most of the noise being generated by the robot itself). You can use two Texas Instruments OPA2244 double op-amps, a single quad op-amp chip, or something else with similar features.

Refer to the complete schematic in Figure 3. Let's start with the virtual ground section. The first difficulty is the power supply. Usually, you have a single 5-V power supply for your robot. To have an output signal level swing wide enough to avoid spurious waves generated by signal clipping, you need at least a quasi rail-to-rail op-amp. Fortunately, many modern op-amps can work with a single power supply and an output swing from ground to VCC – 1 V, allowing an output signal of 1.4 V_{RMS} with a 5-V power supply. Using a single power supply, you need to create a "virtual ground," offsetting the input and output signals to half of the power supply. There are some different techniques to do this. Many electronics books cover the topic. A good reference is also Bruce Carter's application note titled "A Single Supply Op Amp Circuit Collection" (Texas Instruments, 2000).

With a multiple-stage amplifier, the best solution is to dedicate one section of the chip to obtain a single low-impedance virtual ground circuit for all of the stages. The resistors and capacitors needed to obtain a filtered reference are only connected to this op-amp. The output can be used as the signal ground for all of the other stages.

Refer to the mixer section in Figure 3. To catch the sound from every direction, three Electret microphones are mounted in three different directions. In this stage, the signals are mixed and slightly amplified. Each stage has a little gain, getting an overall gain factor of 250. In this way, you get all the gain needed to drive the tone decoder at the right level, with the best characteristics in terms of stability and input impedance for each stage. The decoupling capacitors, together with input resistors, have a first 3-kHz high-pass filter effect that cuts off most of the noise.

Experience has taught me the importance of the decoupling filter formed by R36, C33, C34, C35, and C36. It filters the power supply for the active microphones. The NE567 has an internal oscillator to act as a PLL (see description below). This oscillator works at the same

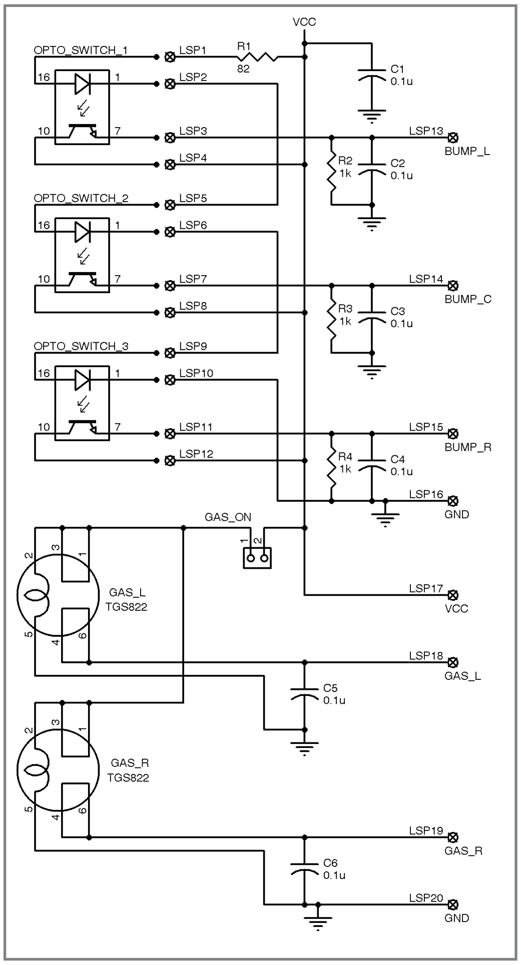


Figure 2—The lower board includes the optical switches for the bumpers and some components for signal conditioning.

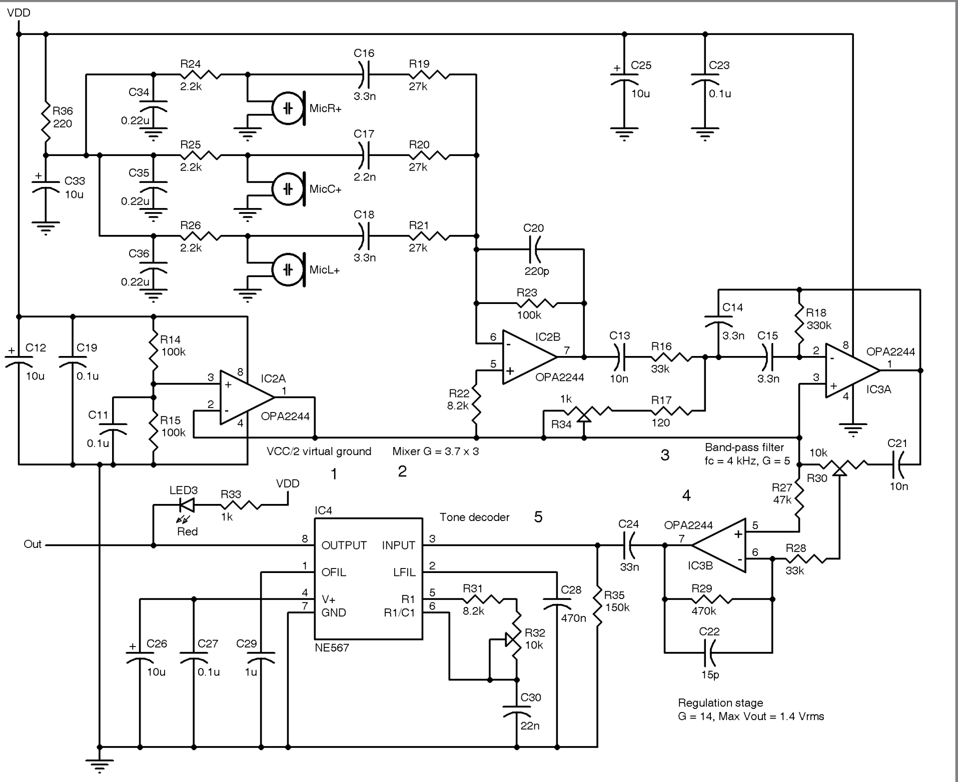


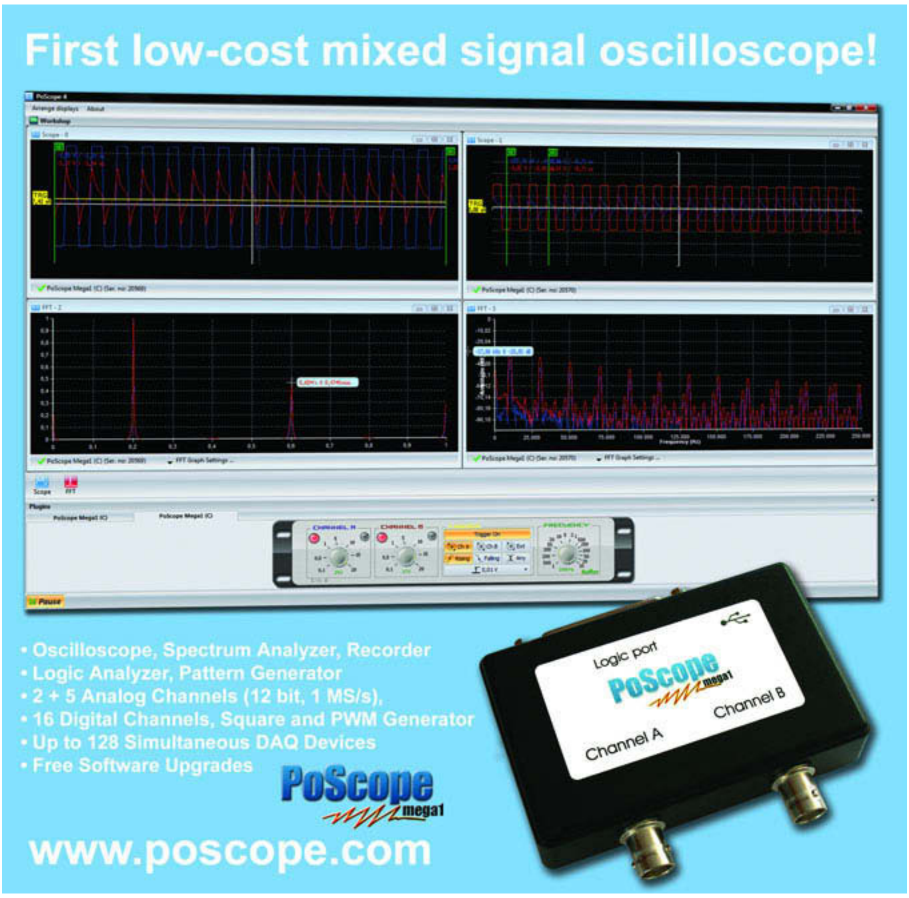
Figure 3—This three-chip board, used to detect the 4-kHz tone generated by the sound target, could be logically divided into five sections, each one with its precise purpose in the decoding process (as noted by the numbers in this figure and detailed in the text). To increase this circuit's performance, the board's overall gain is obtained with a small amount of gain for each stage, as noted with the "G" parameter in the diagram.

frequency we want to reveal (4 kHz). In the power supply line, there is therefore some 4-kHz noise coming from the NE567. Due to the high gain of the amplifier chain, the power supply of the microphones (the input of the chain) has to be well filtered to avoid undesirable behaviors.

Now refer to the band-pass filter section in Figure 3. After mixing and a first amplification, the signal goes to the band-pass section that filters out all the unwanted signals. This stage has a gain factor of five in 4-kHz central frequency with an 8% bandwidth (320 Hz). Once again, there are a lot of programs online that can do the entire job of finding the right values for resistors and capacitors (e.g., Captain's Universe, www.captain.at/electronics/active-filter/). Using a variable resistor for R17, you can accurately trim the central frequency.

Note that the 4-kHz signal alone goes to the final stage that amplifies it for another gain factor up to 14. This stage adjusts the sensitivity of the complete system. The regulation trimmer is at the input of the stage to avoid saturation.

Refer to the tone decoder section of Figure 3. The correct signal level is sent to the input of the tone decoder. This is a classic application of the Philips Semiconductors tone decoder/phase-locked loop (PLL) NE567. When the frequency of the input signal is the same as the internal NE567 oscillator, the PLL locks and sets the digital output pin low. The internal Voltage


First low-cost mixed signal oscilloscope!
• Oscilloscope, Spectrum Analyzer, Recorder
• Logic Analyzer, Pattern Generator
• 2 + 5 Analog Channels (12 bit, 1 MS/s),
• 16 Digital Channels, Square and PWM Generator
• Up to 128 Simultaneous DAQ Devices
• Free Software Upgrades
PoScope mega1
www.poscope.com
Logic port
PoScope mega1
Channel A
Channel B

Controlled Oscillator (VCO) frequency depends on some external components. The primary function of this component is to drive a load whenever a sustained signal within its detection band is present at the input. The bandwidth, center frequency, and output delay are independently determined by means of four external components. The values of these components can be calculated with the formulas in the datasheet or with a program (e.g., NE567.bas at http://web.tiscali.it/i2viu/electronic/electron.htm). Eventually, the output of NE567 becomes a digital signal that can be connected to an input port on the microcontroller so you can know when the sound target is revealed (see Photo 8).

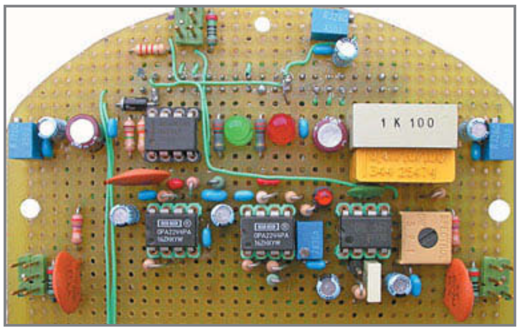


Photo 8—This is a prototype of the sound board on perfboard.

DIRECTION

A roboticist wants the ability to track a robot's location and direction of movement. But designing such a system is not so easy. In my 2009 articles, I described a method to estimate position without any external reference by odometry—meaning, by reading the distance traveled by each wheel. As I detailed in my 2009 article, this procedure could also be very precise when using high-resolution encoders and wheels with a small contact area with the floor. But, even with a small amount of error for each turn, the precision decreases proportionally with the space traveled since the error is cumulative. This increases the uncertainty circle after several dozens of meters, especially with an irregular floor, where wheel slipping and jumping can occur.

To increase dead-reckoning precision, many kinds of "sensor fusion" methods are experimented with. Gyrodometry is a sophisticated method.[3] This uses a combination of sensors that requires some computation capabilities for both the microcontroller and the programmer who writes the code.

Knowing the absolute orientation may be useful to partially compensate for the errors caused by odometry. A simple and affordable way to measure bearing is with a digital compass. This kind of sensor is popular because it is easy to find and inexpensive. Plus, you can find a lot of code examples on the Internet in every programming language.

The popular Devantech CMPS03 electronic compass can be interfaced in different ways. I chose I^{2}C interfacing. This method makes it easier to drive the compass for various purposes. The bearing value can be read in a 2-byte variable with a theoretical resolution of 0.1° (from 0 to 3,599, meaning 359.9°), which is much higher than the real precision. The calibration procedure can be started with a specific command, even with an automatic sequence that turns the robot 90° and performs a measure, then turns another 90° step and performs another measure, and so on, for each cardinal point until reaching 360°. As described in the manual, calibration is important and must be performed at least once after installation for the purpose of precision. It compensates for some static magnetic fields, and it corrects the inclination reading. Magnetic field inclination varies throughout the world.

To correctly use the values returned by the compass, keep in mind that this electronic device uses exactly the same physics as a regular magnetic needle compass, with all the pros and cons. The compass must be kept parallel to the floor to avoid reading the vertical axis of the Earth's magnetic field. Some digital compasses are tilt-compensated with a three-axis magnetic sensor and a three-axis accelerometer to avoid this effect in a given range (sort of an electronic version of the gimbals used on shipboard compasses), but they are more expensive. The compass is affected

Leading Embedded Development Tools...

Easy to use, from concept to final product

- MDK-ARM is an Integrated Development Environment for ARM and Cortex™-M microcontrollers
- RL-ARM is a Library Collection designed to solve real-time and communication challenges
- ULINK*pro* is a high-speed Debug and Trace unit for detailed analysis of software quality

www.keil.com 1-800-348-8051

KEIL™ Tools by ARM

by surrounding magnetic fields that are often much stronger than the Earth's magnetic field (e.g., iron objects, magnets, and electricity).

SOFTWARE & SETTINGS

The software runs on an Arduino Diecimila board. All the sensors except the bumpers are interfaced on the I^{2}C bus directly or through a MAX127 converter. The battery voltage is monitored with an A/D port to determine when a power shortage is coming. Thanks to the Wire libraries, the software is very simple, and it's just a matter of cyclically polling each sensor with the right timing.

Sequence	Time (ms)	First action	Second action
0	0	Left US set	Gas set
1	14	Gas read	Left IR set
2	28	Left US read	Center US set
3	42	Left IR read	Center IR set
4	56	Center IR read	Center US read
5	70	Right US set	Right IR set
6	84	Right IR read	Vbatt set
7	98	Vbatt read	Right US read

Table 1—Here you see a description of the actions performed to read sensor values in every state machine cycle. Alternating set and read actions for a different kind of sensor in any cycle, it follows the timing required by the specifications. A full cycle requires 98 ms. After that, the values are sent to the navigation board 10 times per second.

Because the SRF08 sensors come with the same address, the first setting must assign a different I^{2}C address to each one. It is needed just once to change the default address E0. Thus, you must have only one sonar on the bus at a time. When powering up the SRF08 without sending any commands, it will flash its address on the LED—one long flash followed by a number of shorter flashes, indicating its address from 0 to 15. The flashing is terminated immediately by sending a command.

Other sensor settings are not permanent, so they must be applied at every startup. Thus, before the main cycle starts, the analog gain and measuring range must be configured on the SRF08. The analog gain register sets the maximum gain of the analog stages. To set it, just write the value to the gain register at location 1.

During the ranging process, the analog gain starts off at its minimum value of 94. This is increased in approximately 70-µs intervals up to the maximum gain setting, set by register 1. The maximum possible gain is reached after about 390 mm. Providing a maximum gain limit enables you to fire the sonar more rapidly than 65 ms. Since the ranging process can be very short, a new ranging process can be initiated as soon as the previous range data has been read. A potential hazard with this is that the second ranging process may pick up a distant echo from the previous "ping," which could create a false result. To reduce this possibility, the maximum gain can be reduced to limit the module's sensitivity to the weaker distant echo, while still be able to detect nearby objects. The maximum gain setting is stored only in the module's RAM and is initialized to maximum at power-up, so if you only want to do a ranging every 65 ms, or longer, you can ignore the range and gain registers. Note that the relationship between the gain register setting and the actual gain is not a linear one. Also, there is no magic formula to apply. It depends on the object's size, shape and material. Try playing with different settings until you get the result you want. If you appear to get false readings, it may be echoes from previous "pings." Try going back to firing the SRF08 every 65 ms or longer.

An internal timer sets the SRF08's maximum range. By default, this is 65 ms or the equivalent of 11 m of range. This is much farther than the 6 m the SRF08 is actually capable of. It is possible to reduce the time the SRF08 listens for an echo, and hence the range, by writing to the range register at location 2. The range can be set in steps of about 43 mm (0.043 m or 1.68") up to 11 m. The range is ((Range Register × 43 mm) + 43 mm), so setting the Range Register to 0 (0x00) gives a maximum range of 43 mm. Setting 255 (0xFF) gives the original 11 m (i.e., 255 × 43 + 43 = 11,008 mm). For this purpose, I chose a value of 57 to obtain a 2.5-m range (57 × 43 + 43 = 2,494-mm range). Considering a sound speed of 340 m/s, I can apply a ping time of about 14 ms: (54,988 mm)/340 m/s = 14.67 ms. Other sensors require no initialization.

As I already mentioned, the digital compass must be calibrated for the local inclination. But this procedure is needed just once, and it is not part of the running software.

CYCLING

A sensor cycle is performed in sequence every 14 ms, which gives enough time to stabilize measurement for each kind of sensor. Sonars wait for echoes every 14 ms, and the different modules must not be fired at the same time. IR sensors require 39 ms for each measure. Most sensors of this kind require a settling command followed by a reading command after the time required to measure to be executed. The program acts as a state machine with a 14-ms clock, alternating sets, and reads cycles as described in Table 1. The entire cycle completes in about 100 ms to read three range finders, three IR distance sensors, three light sensors, the battery voltage, and gas sensors. The digital compass value can be read at any moment without any settling; therefore, this measure is executed in the last cycle.

A robot running at 50 cm/s travels for 5 cm in 100 ms, which could be too much. In order to have the fastest possible response when avoiding obstacles, a check is performed at each cycle for dangerous distances. If the measured distance on any side is less than the threshold, or if any of the three bumpers is active, an immediate alarm is sent to navigation board. Instead of waiting the whole 100-ms cycle to communicate distance, the measuring packet is sent immediately after the measure during an emergency. In this way, the robot travels a maximum of 0.7 cm before braking, instead of 5 cm as in case of worst condition. Therefore, the robot is "blind" for only 0.7 cm instead of 5 cm.

READING & TRANSMISSION

The SRF08 sensors can return the distance already converted in centimeters or inches. While reading the distance value, the light values from SRF08 modules are also read. This can be done within a single I^{2}C reading cycle on registers 1 and 2. Sharp IR sensors instead return an analog value inversely proportional to the distance of the object. According to the documentation, the response curve can be approximated with a 1/x trend. A simple conversion can be obtained with the following formula:

$$\text{Distance} = \frac{K}{\text{ADC value} - \text{offset}}$$

K and offset constants can be computed by just reading two values in the quasi-linear portion of the response curve. I chose values of 4 cm for Distance 1 and 40 cm for Distance 2 (still within the measurable range). In my configuration, this returns: $ADC_{VAL1} = 2{,}492$ and $ADC_{VAL2} = 278$. Therefore:

$$K = \left(ADC_{VAL2} - ADC_{VAL1}\right) \times \text{Dist1} \times \frac{\text{Dist2}}{\text{Dist2} - \text{Dist 1}} = 9{,}840$$

$$\text{Offset} = \frac{\left(\text{Dist2} \times ADC_{VAL2} - \text{Dist1} \times ADC_{VAL1}\right)}{\text{Dist2} - \text{Dist 1}} = 32$$

Of course, using centimeters when calculating constants returns centimeters for "Dist" in the formula. Using inches returns inches.

When you compare the converted reading from the Sharp sensors with direct readings from the SRF08 sonar, the differences are within 1 cm from flat and mat objects. Under regular conditions, the actual distance from the object is considered the shortest distance between the three sensors on the same side (bumper = on means Dist = 0). This scenario is best for safe obstacle avoidance. The maximum distance is 255 cm, which allows a 1-byte transmission packet.

When the voltage of the battery falls below a given threshold, an alert sounds and a light illuminates on the sensor board. Reading the digital compass is simple. The bearing value is transmitted as is (splitting the 2-byte variable) to the navigation board.

A simple protocol is used to transmit the data from the sensor board to the navigation board. This protocol is the same that's used for telemetry on the navigation board I described in my previous articles (*Circuit Cellar* 224 and 225, 2009).

DESIGN SUCCESS

Do you need your robot to be able to navigate safely in an obstacle-ridden environment? This project is for you. If you follow my lead, you can build a similar system without breaking the bank. But this project is just a starting point. Of course, you can use many other sensors to achieve your goals—such as locating stairs for a cleaning robot or enabling your robot to follow a line. This board has enough I/O and computation power to get the job done. ■

Guido Ottaviani (guido@guiott.com) has experience working as an analog and digital designer for a communications company in Italy. He also has worked as a system integrator and a technical manager for a large Italian publishing group. In his spare time, Guido enjoys working with his scope and soldering iron on autonomous robotics projects. He's an active member in Italian robotics groups.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/236.

REFERENCES

[1] Arduino, Shields, www.arduino.cc/en/Main/ArduinoShields.

[2] Arduino, Libraries, www.arduino.cc/en/Reference/Libraries.

[3] J. Borenstein and L. Feng, "Gyrodometry: A New Method for Combining Data from Gyros and Odometry in Mobile Robots," The University of Michigan, 1996, www-personal.umich.edu/~johannb/Papers/paper63.pdf.

RESOURCES

Captain's Universe, "Active Filter Calculator: Band-pass with Op-Amp Designer in Javascript," www.captain.at/electronics/active-filter/.

B. Carter, "A Single Supply Op Amp Circuit Collection", SLOA058, Texas Instruments, 2000, www.ti.com/sc/docs/psheets/abstract/apps/sloa058.htm.

NXP Semiconductors, "Tone Decoder/Phase-Locked Loop," NE567/SE567, 2002, www.nxp.com/acrobat_download/datasheets/NE567_SE567_2.pdf.

Sharp Corp, "GP2D120 Optoelectronic Device," SMA06008, 2006, www.sharpsma.com/Page.aspx/americas/en/part/GP2D120/.

SOURCES

Diecimila board
Arduino | www.arduino.cc

TGS822 Gas sensor
Figaro USA, Inc. | www.figarosensor.com

CMPS03 Electronic compass and SRF08 ultrasonic rangefinder
Devantech Ltd | www.robot-electronics.co.uk

Philips NE567 tone decoder/PLL
NXP Semiconductors | www.nxp.com

GP2D120 Infrared module
Sharp Microelectronics of the Americas | www.sharpsma.com

OPA4244 Op-amp
Texas Instruments, Inc. | www.ti.com