DIY Home Automation, p. 26 • Build a Network Service Monitor, p. 36 • Custom MCU-Based Machine Control, p. 60

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#238 May 2010

MEASUREMENT & SENSORS

Inertial Measurement Unit Design

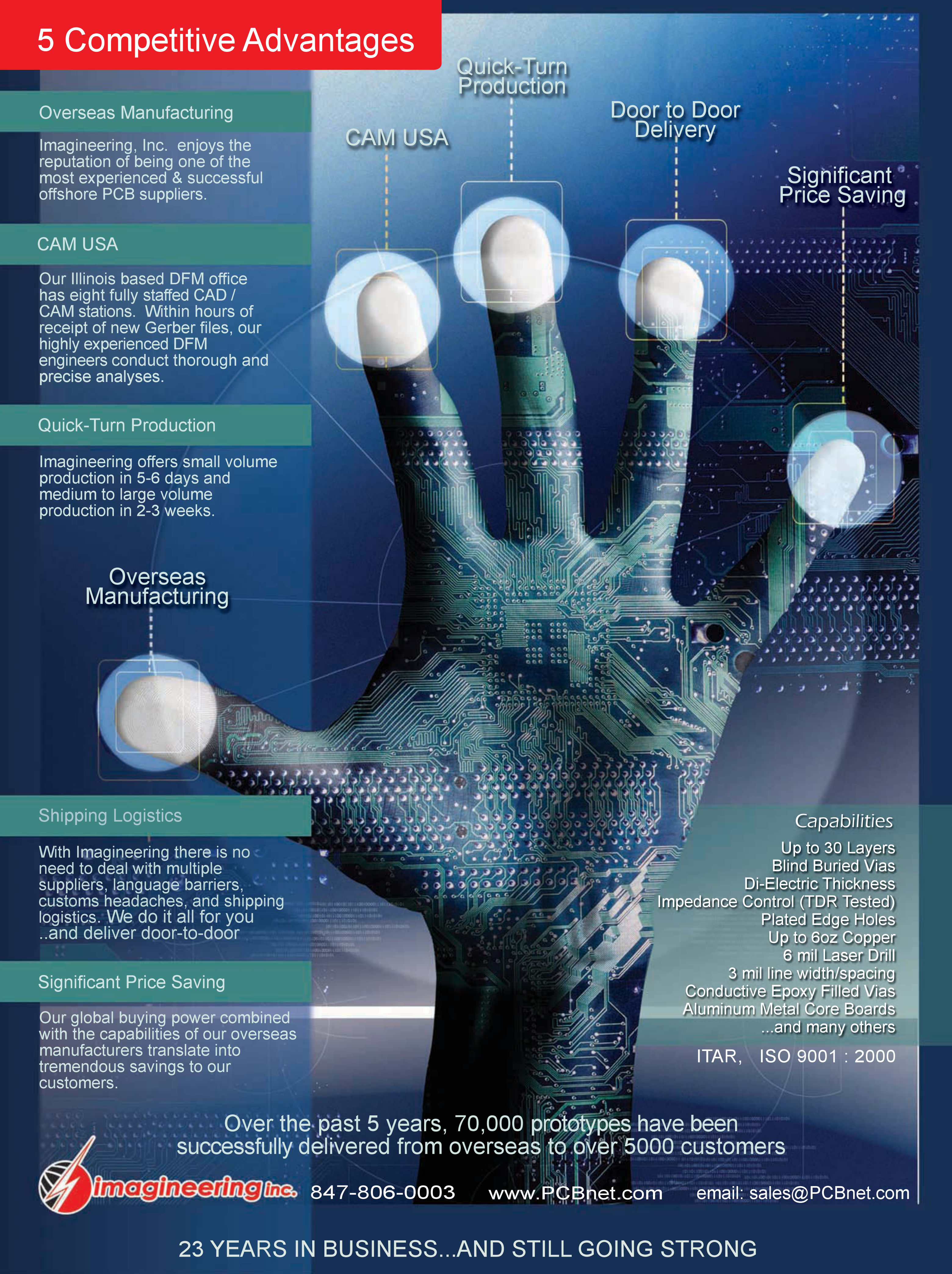
Infrared Tech & Next-Generation Sensing Applications

An OAE Probe Amp/Intercom System

C Language Tutorial: Phase 2

$5.95 U.S. ($6.95 Canada)

www.circuitcellar.com



5 Competitive Advantages
Overseas Manufacturing
Imagineering, Inc. enjoys the reputation of being one of the most experienced & successful offshore PCB suppliers.
CAM USA
Our Illinois based DFM office has eight fully staffed CAD / CAM stations. Within hours of receipt of new Gerber files, our highly experienced DFM engineers conduct thorough and precise analyses.
Quick-Turn Production
Imagineering offers small volume production in 5-6 days and medium to large volume production in 2-3 weeks.
Shipping Logistics
With Imagineering there is no need to deal with multiple suppliers, language barriers, customs headaches, and shipping logistics. We do it all for you ..and deliver door-to-door
Significant Price Saving
Our global buying power combined with the capabilities of our overseas manufacturers translate into tremendous savings to our customers.
CAM USA
Quick-Turn Production
Door to Door Delivery
Significant Price Saving
Overseas Manufacturing
Capabilities
Up to 30 Layers
Blind Buried Vias
Di-Electric Thickness
Impedance Control (TDR Tested)
Plated Edge Holes
Up to 6oz Copper
6 mil Laser Drill
3 mil line width/spacing
Conductive Epoxy Filled Vias
Aluminum Metal Core Boards
...and many others
ITAR, ISO 9001 : 2000
Over the past 5 years, 70,000 prototypes have been successfully delivered from overseas to over 5000 customers
imagineering inc.
847-806-0003
www.PCBnet.com
email: sales@PCBnet.com
23 YEARS IN BUSINESS...AND STILL GOING STRONG

TASK MANAGER

The Future Is Now

Every issue of *Circuit Cellar* is special. But this issue strikes me as particularly extraordinary. Why? Here we present articles about some of the most innovative, talked-about, 21st-century technologies. These are concepts that your EE forefathers couldn't have imagined—and even if a few of them did dream of something similar to such technologies, they likely treated the ideas as more fiction than fact. Which technologies am I talking about? An inertial measurement unit (think: unmanned aircraft), a home automation system for an energy-efficient home (think: home control), portable monitoring devices (think: real-time server maintenance), and custom machine control circuits (think: programmed system operation) are examples.

If you take nothing else from this issue (which is doubtful), remember this: *the future is now*. Technology is so advanced that you can make most far-out ideas a reality. The key is being able to quickly gather quality information about cutting-edge technology so you can start working on the designs that were things of fiction only a few years ago. We're here to help.

Start with TJ Bordelon's article titled "The FreeSpace IMU" (p. 14). You learn not only about inertial measurement units, but also about a quaternion-based algorithm for attitude estimation. If you don't understand these topics, you probably shouldn't be designing an aircraft-monitoring system.

Turn to page 26 for an article about an exciting "intelligent energy solution," which is one of the hottest topics of our time. Stefan Siegel describes the home automation system for his energy-efficient house. In this article, he focuses on system design and server software.

On page 36, Alexander Popov and Peter Popov describe a portable network service monitor. You can build your own and use it to pull data from any server in a local network. Another DIY project is Chris Paiano's OAE Probe Amp and Intercom (p. 44). He describes how he planned the system and started the design. This article proves that having a thorough foundation in technical theory (otoacoustic emission theory in this instance) can lead to an effective end product (the probe amp and intercom system).

On page 54, George Martin gives you another dose of programming tips. This is the second article in his series about using C language. The article details some program specifics. In the next column, Jeff Bachiochi touches on a topic that has probably been on your mind since childhood: machine control. Turn to page 60 to learn how he customized control circuitry for tread speed mechanism. Tom Cantrell wraps up the issue with an article about two hot topics that have intrigued designers for decades: touch sensing and proximity sensing (p. 66). Today, these technologies are used in virtually every industry, from consumer electronics to industrial to military. Staying on top of the advances in these technologies will enable you to continue designing relevant systems.

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.
1187 Washington 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

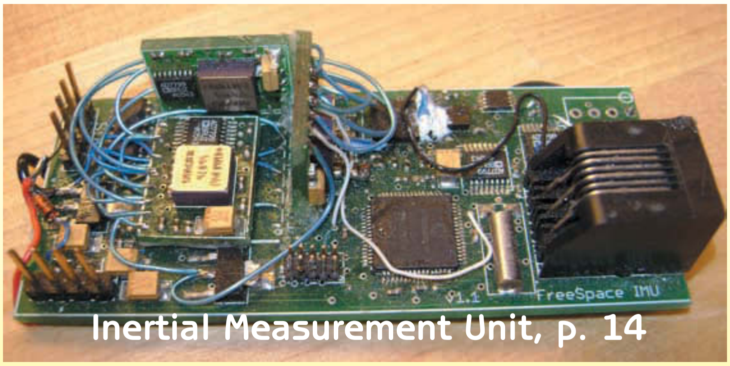
AUTHORS
Authors' e-mail addresses (when available) are included at the end of each article.

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.**

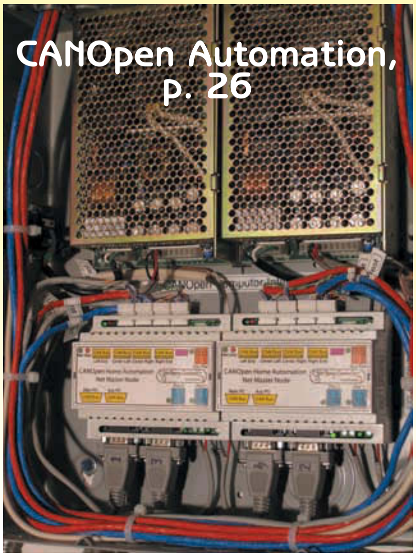
Postmaster: Send address changes to Circuit Cellar, Circulation Dept., P.O. Box 5650, Hanover, NH 03755-5650.

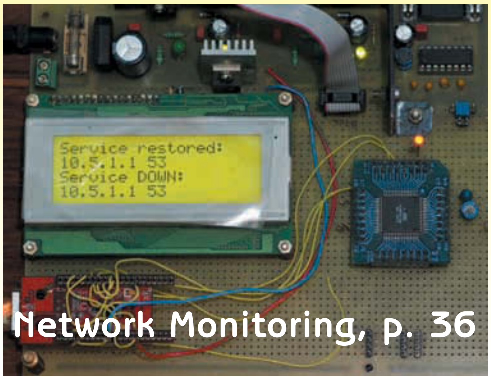
INSIDE ISSUE 238

May 2010 • Measurement & Sensors



Inertial Measurement Unit, p. 14


CANOpen Automation, p. 26



Network Monitoring, p. 36

Amp & Intercom, p. 44

TINY MCUs FOR POWER, SPACE, AND COST-SENSITIVE APPLICATIONS

The EFM32 "Tiny Gecko" (TG) microcontrollers are entry-level devices for low-power applications subject to stringent PCB space and product cost limitations. Costing less than $1, the Tiny Gecko microcontrollers offer the energy-efficient performance of the company's bigger Gecko products combined with smaller flash and RAM combinations and a choice of space-saving QFN20, QFN32, and QFN64 packaging.

The 32-bit, Cortex-M3 processors use less than a quarter of the energy needed by alternative 8-, 16-, or 32-bit microcontrollers. Their current consumption is extremely low: in Active mode typically 180 µA per megahertz executing code from flash; 900 nA in deep Sleep mode; and 20 nA in Shut Off mode. Low-current performance and wake-up times of less than 2 µs enable the devices to extend battery cell life by a factor of at least four.

The smallest, lowest-cost device—the EFM32TG100—provides flash and RAM capacities of 4 to 32 KB and 1 to 4 KB, respectively, and 12 GPIO pins. To simplify application scaling, the QFN32 packaged TG200 and QFN64 packaged TG230 and TG840 are also pin-compatible with the bigger Gecko family of products and provide flash options of 8, 16, and 32 KB, either 2 or 4 KB of RAM, and 24 or 56 GPIO.

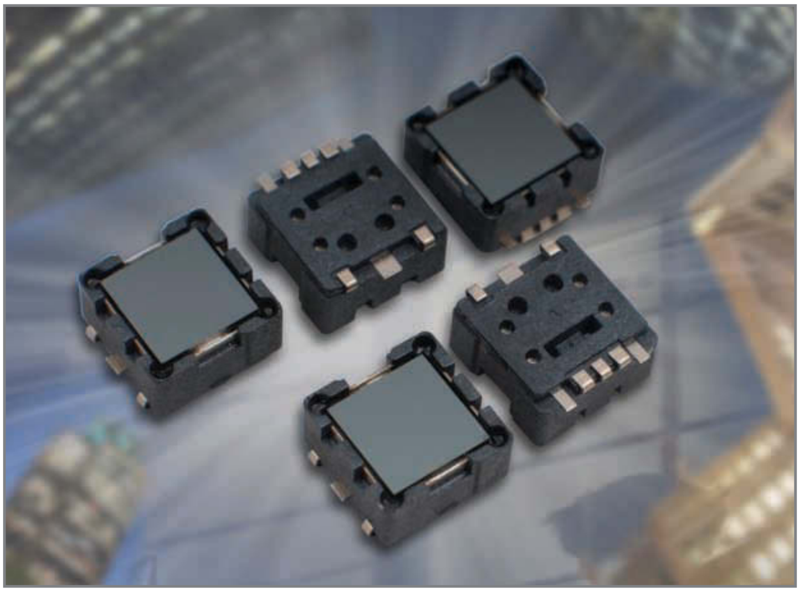
At the top of the range, the TG840 devices embed a 4 × 24 segment LCD controller consuming less than 900 nA, while all TG parts offer an AES encryption/decryption engine as standard, a prerequisite for many wireless applications.

Thirteen EFM32 Tiny Gecko microcontrollers are scheduled to enter full production during Q4 2010. Pricing starts at **$0.99** each in 100,000-piece quantities.

Energy Micro AS
www.energymicro.com

WORLD'S SMALLEST PYROELECTRIC INFRARED SENSOR

The **IRS-B210ST01-R1** is the world's smallest pyroelectric infrared (PIR) sensor. The new sensor measures 5.0 mm × 4.7 mm × 2.4 mm, making it ideal for use in compact, low-profile electronic equipment and energy-saving devices in particular. As the only surface-mount PIR sensor available on the market today, it provides numerous features, including high sensitivity and signal-to-noise ratio, excellent stability against temperature changes and white light, and enhanced radio frequency interference (RFI) immunity.



The many design benefits of the compact component make it well-suited for a variety of security and lighting applications to further help homeowners and businesses decrease energy consumption. For example, the sensors have the ability to detect infrared emitted by the human body for security-sensing applications. Also, the component can be utilized in household appliances, such as lighting, which switch off automatically when no one is in proximity to further save energy.

The PIR sensor includes dual 0.85 mm × 1.2 mm electrodes with a field of vision of ±70° (horizontal) and ±50° (vertical). Its operating temperature range is –40° to 70°C and the device is driven by supply voltage in the range of 2 to 15 V.

The IRS-B210ST01-R1 sample price is **$1.88**, and the lead-time is approximately 12 weeks.

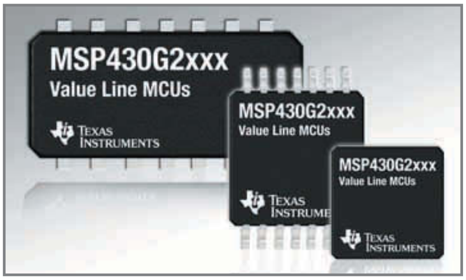
Murata Electronics North America
www.murata-northamerica.com

NEW PRODUCT NEWS

Edited by John Gorsky

MSP430 VALUE LINE OFFERS 10x PERFORMANCE AND BATTERY LIFE

The **MSP430 Value Line** offers superior 16-bit MCU performance and industry-leading ultra-low-power consumption at ultra-low 8-bit MCU prices. The Value Line ensures 8-bit developers no longer need to sacrifice performance, power efficiency, or scalability because of price. The roadmap includes more than 100 MCUs to be released over the next 15 months, providing developers with a broad portfolio to best fit their memory, peripheral, and packaging configuration needs. The new MSP430G2xx devices are code compatible across the MSP430 MCU platform, enabling easy code migration and upgrades to higher-end devices as application requirements evolve. The new MCUs are also supported by Texas Instruments's easy-to-use MSP430 tools, free software, and broad third-party support network, driving faster time-to-market across a broad range of cost-sensitive applications, including safety, security, and touch sense.



Key features include: up to 10x performance, including true 16-MIPS operation and 50% greater code density; five power modes with ultra-low stand-by power of 0.4 µA; and less than 1 µs wake-up time. Integrated intelligent peripherals—such as 10-bit ADCs, UART, comparator and serial communication—offload CPU tasks for increased power efficiency.

The new MSP430G2xx MCUs are priced from **$0.25** for 100,000 units.

Texas Instruments, Inc.
www.ti.com

NEW BLUETOOTH MODULES FOR EMBEDDED APPLICATIONS

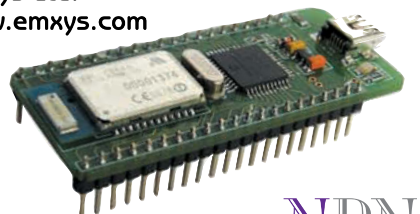
With the brand "WirelessEmbedded," emxys is launching a renewed portfolio of Bluetooth modules, specifically targeting the embedded applications industry. The line's core are the **ulceBlue2** and **BTswitch** embedded modules.

ulceBlue2 is an embedded Bluetooth module based in the PIC18F4550 microcontroller and PearlBlue Bluetooth module. The device (a new version of former ulceBlue) is enhanced with this USB PIC micro, which enables the possibility to implement simultaneous USB and Bluetooth communications through a unique device. The module has a micro-USB connector included with the possibility to get the module powered directly from USB port.

BTSwitch is a Bluetooth PearlBlue-based board with four power relays enabled for applications up to 240 VAC at 1 A. With this module, it's possible to monitor and change the state of the switches and to get the logic state of the included optoisolated digital input. It is well suited for remote control applications in fields as domestic, industrial control, and more.

The portfolio includes extensive documentation, application software, source code for complex applications, and more. The ulceBlue2 costs approximately **$160.** The BTSwitch costs approximately **$175**.

emxys Ltd.
www.emxys.com



NPN

WORLD'S SMALLEST FINGERPRINT-MATCHING MODULE

The new **MK67Q5250** fingerprint-matching module is designed to speed the implementation of automated fingerprint identification and access control in applications ranging from locks and garage door openers to portable devices, computer workstations, and POS terminals. This low-power module—measuring less than 1 square inch in size—utilizes the ML67Q5250 LSI MCU that has been factory-programmed to control the module's AES2510 fingerprint imaging sensor from Authentec. It can perform fingerprint processing and matching, store biometric control code, application code and templates, and communicate with a host processor. SPI is the default communications interface to the host processor from the module. The module can be configured for other communications protocols such as UART or USB.

The ML67Q5250 is the industry's only single-chip fully-integrated fingerprint authentication system. Alternative solutions typically employ costly and power-consuming high-performance microcontrollers or processors that require external memory. By using the available software development kit the designer does not need to write or verify any biometric software. The designer—using standard code development tools—can write application specific code for the on-chip 32-MHz ARM7 processor and combine it with the SDK-generated biometric software to meet the needs of their specific application.

The MK67Q5250 module costs **$90** in small quantities, and the ML67Q5250 LSI MCU is **$10** in small quantities.

OKI Semiconductor
www2.okisemi.com

NPN

STM32 VALUE LINE FOR COST-SENSITIVE APPLICATIONS

The **STM32 Value Line** are new ultra-low-cost, 32-bit microcontrollers bringing the advantages of the STM32's 32-bit advanced industry-standard core into applications served by lower-performing devices that lock developers into proprietary processor architectures.

Until now, engineers upgrading legacy 16-bit designs for higher performance and flexibility have had to choose from high-end alternatives that are featured for more complex applications. The STM32 Value Line remedies this by combining a 24-MHz ARM Cortex-M3 processor core with peripheral features optimized for typical 16-bit applications, such as home-entertainment products, appliances, and industrial equipment. At 24-MHz with zero wait state embedded flash memory access, the devices deliver up to 30 DMIPS, outperforming most of the 16-bit processors.

Features embedded in the line include up to 12 16-bit timers, including a PWM timer for motor-control applications. As many as 26 timer channels are available, with package options providing up to 100 pins, as well as a 12-bit high-speed ADC. Free software support—a complete three-phase motor-control library for appliance and motor-control applications and free VDE-certified libraries to help household appliances achieve necessary approval to EN/IEC60335-1 Class B—is also included. In addition, a 12-bit DAC allows developers to add audio support in a variety of applications (e.g., security devices or home automation systems) and to address wider opportunities requiring analog waveform generation and control.

One-thousand-piece prices range from **$0.85** for 16-KB devices in a 48-pin LQFP48 package to **$1.44** for 64-KB devices in the 64-pin LQFP64 package.

STMicroelectronics
www.st.com

NPN

CIRCUIT CELLAR

Test Your EQ

Edited by David Tweed

Problem 1—*Audio tone detection is a common application for DSP, and the Fourier Transform (FT), which converts a time-domain representation of a signal into a frequency-domain representation, is often considered as an implementation method. Let's explore some questions related to this topic.*

Describe the differences among the FT, the DFT, and the FFT.

Problem 2—*What information from an FT is not important in a tone-detection application? What does this imply about the implementation?*

Problem 3—*What effects does varying the block size of an FFT algorithm have?*

Problem 4—*What is the difference between the DFT and the Goertzel algorithm? What advantage does the DFT have over the FFT or Goertzel algorithms in tone-detection applications?*

What's your EQ?—The answers are posted at www.circuitcellar.com/eq/

You may contact the quizmasters at eq@circuitcellar.com

Contributed by David Tweed

by TJ Bordelon (USA)

The FreeSpace IMU

A Quaternion-Based Algorithm for Attitude Estimation

An unmanned robotic vehicle requires a working inertial measurement unit (IMU), which outputs an estimation of the attitude, or orientation, of a vehicle in 3-D space. For attitude estimation, you need MEMS sensors (e.g., gyros, accelerometers, and magnetometers) and a sufficient algorithm to "fuse" them together. This article covers a simple quaternion-based algorithm for an IMU project.

The Holy Grail for hobbyists and engineers wanting to get into the exciting world of unmanned robotic vehicles is a working inertial measurement unit (IMU) of their own design. An IMU outputs an estimation of the attitude (or orientation) of a vehicle in 3-D space, and it is the foundation for implementing autopilots or remote instrument piloting for small robots (e.g., aerial or underwater systems). In recent years, the cost of purchasing such devices has dropped from many thousands to a few hundred dollars; however, rolling your own gives you control over the estimation parameters and how the sensor data is weighted in ways impossible with present off-the-shelf units. Plus, you can take advantage of your unique situation when you have complete control over the design.

At first glance, producing an attitude estimation using microelectromechanical systems (MEMS) sensors—such as gyros, accelerometers, and magnetometers—may seem like a trivial task. But, these sensors each have their own strengths and weaknesses. The complexity arises in the algorithm that fuses the sensors together to produce the final result. In this article, I'll present my Microchip Technology dsPIC33FJ256GP506-based "FreeSpace IMU" and describe an algorithm that you can use for an IMU project of your own (see Photo 1).

PROJECT EVOLUTION

In 2001, I set out to make my own drone aircraft completely controllable from a laptop. Obviously, I wanted to display the primary instruments available to real pilots, and that included an "artificial horizon." At first, I thought simply using an accelerometer and magnetometer triad (compass and gravity sensor) would be enough. But after talking to a few pilots

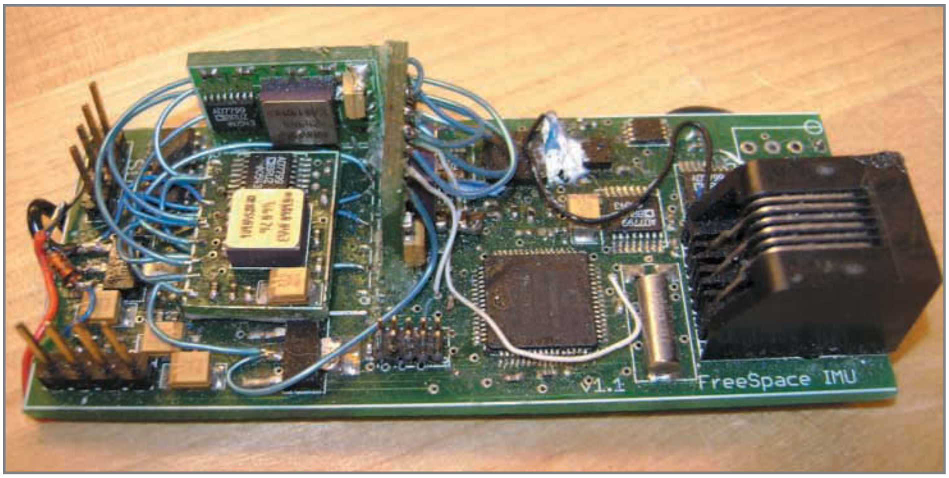


Photo 1—This is the very first prototype of the FreeSpace IMU, which has managed to survive a few crash landings. To the left, three gyro modules are installed orthogonally to one another. The remaining sensors are mounted above the larger microcontroller package, with the programming connector to the right.

Photo 2a—The IMU was extensively flown and tested on a modified off-the-shelf "Easy Star" model airplane. After the algorithm was validated, an autopilot was developed simply by closing a few feedback loops around the IMU outputs. b—The IMU is mounted in the wing to keep it away from the nearby transmitter and vibrating motor.

about why they seemed to use gyroscopes in their instrumentation, the problem didn't seem so simple. It turns out, the centrifugal force makes gravity seem to come from places other than down! So, I obviously needed some gyros.

Within a week, I threw together a circuit board with a full array of sensors,

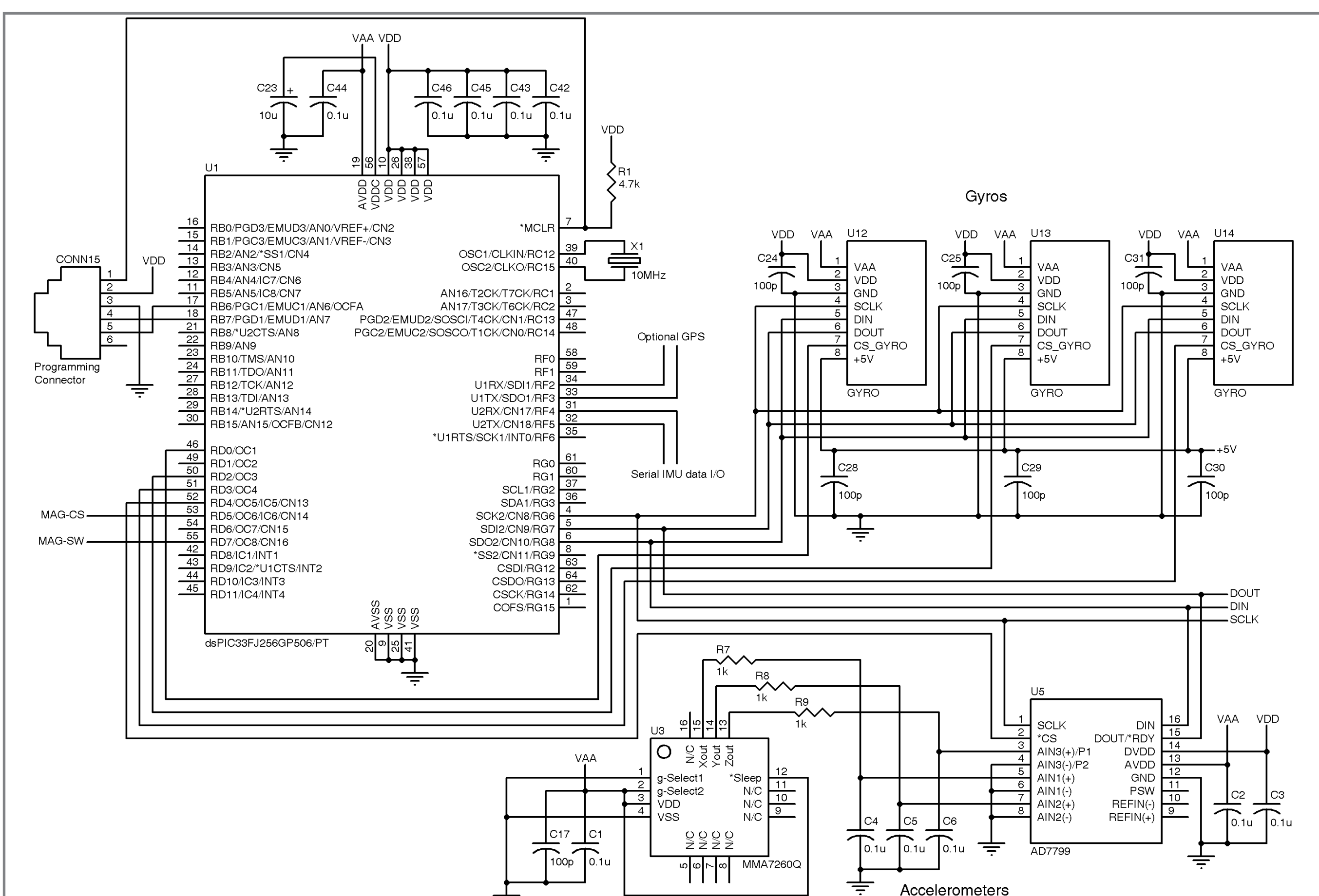


Figure 1—The main IMU schematic. The SPI bus is used to communicate with the ADC chips on each sensor. Each gyro is mounted on a separate module mounted orthogonally on this main board. Special care is taken to separate analog and digital supplies (and grounds). The key to a successful design is primarily in accurate sensor placement and a clean power supply.

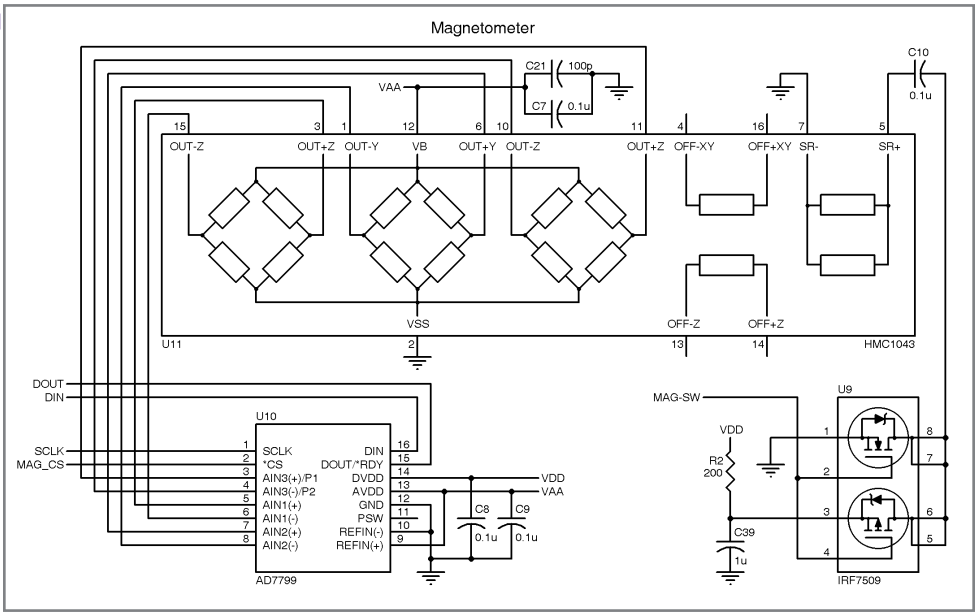


Figure 2—The magnetometer is also part of the main board. This measures Earth's magnetic vector in 3-D. The MOSFETs are utilized to send a high-current set/reset pulse, which is necessary at power-up or after exposure to strong magnetic fields.

including some rate gyros and a three-axis accelerometer module. A few lines of code were all that stood in the way of success! Unfortunately, it wasn't that easy, and I quickly became frustrated. The gyros were incredibly drifty, and the algorithms I was finding in various papers and publications were confusing. So, I hit the Internet and noticed that almost everywhere I went folks were saying Kalman filtering was a magic bullet. People seemed to be saying that if you drop in this magic bullet, you could solve all these problems, and then the sky would clear up and rainbows would appear.

Well, six months of learning new math and reading a dry PhD thesis or two didn't help much. At that point, I had a closet of broken planes, and I'd been kicked out of a few local model airplane flying fields. I obviously had a lot to learn. It was then that a small local company, Continental Controls and Design, asked me to help write the software for a UAV called "Locust." This opportunity couldn't have come at a better time. Over the next few years, I worked with industry veterans and learned how to solve my problems. Everything I learned helped me make a rather cool backpack-deployable 18" flying wing with the world's smallest (in 2001) autopilot in 1 cubic inch of volume.

Armed with this experience, I set out to make my own IMU. This time I was determined to make the design even simpler than the one for the Locust, and I hoped to save others the years of frustration and crashes that I had endured.

One of the biggest complaints I had with the earlier IMU algorithms were their reliance on Euler angles (yaw, pitch, and roll), and their insistence on implementing "offset/bias tracking," which seemed to add a level of complexity above what felt necessary for the accuracy I was looking for. I didn't care about squeezing the theoretical maximum out of the hardware. I merely wanted to obtain reasonable accuracy within a couple of degrees.

AN IMU EMERGES

What I'll share with you here is an IMU of my own design (see Figure 1, Figure 2, and Figure 3). It's quaternion-based, which means I don't have the problem of gimbal lock. It uses no bias tracking, and has been flown dozens of times on a model airplane pulling 4 Gs in turns and hitting speeds of 70 MPH without losing its brains (see Photo 2). And best of all, you don't have to be a mathematician to implement the algorithm I developed.

To implement this design, I needed a triad of sensors (on X, Y, and Z) measuring rotation rate, the Earth's magnetic vector, and acceleration. Fortunately, all these quantities can be measured with commonly available MEMS sensors. We'll simply refer to these sensors as the gyro, magnetometer, and accelerometer, respectively.

Figure 4 is a block diagram of my IMU hardware. It's nothing more than careful voltage regulation and a handful of sensors feeding a microcontroller via an

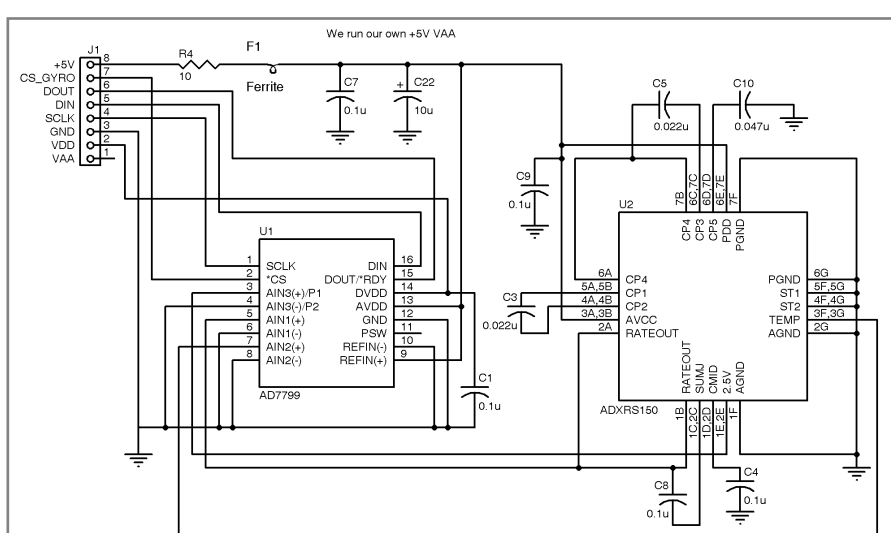


Figure 3—Three of these boards are mounted to the main board (see Figure 1), each measuring a single axis of rotation. It is very important to keep the analog supply as clean as possible, especially for these sensors. The ADXRS150 is not "ratiometric," so we must scale the RATEOUT voltage in proportion to the 2.5-V reference provided by the gyro.

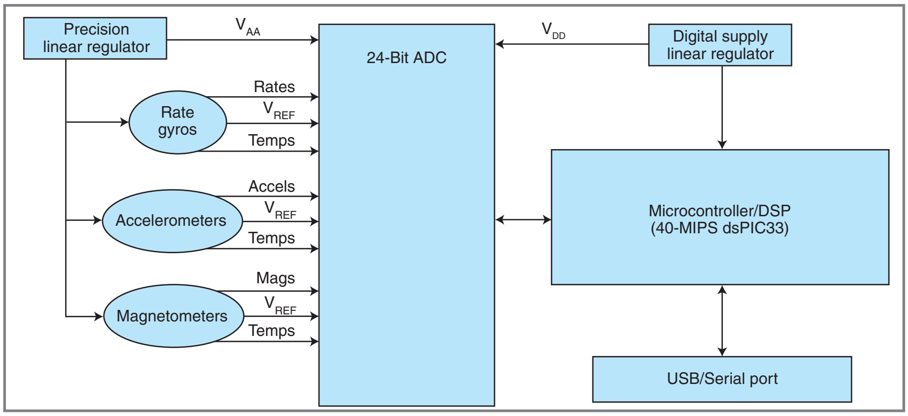


Figure 4—A simplified "high-level" view of the IMU design. Precision voltage regulators ensure clean sensor outputs into a high-precision ADC. The IMU algorithm running in the microcontroller continuously reads the ADC and runs the "FreeSpace" algorithm to produce an attitude estimation. V_{REF} is utilized to account for any ripples or variation in voltage.

Analog Devices AD7799 ADC. Selecting the specific components isn't complicated beyond finding the most accurate and temperature-immune ones you can afford. Be careful to watch out for certain sensors with built-in digital temperature calibration. I've found they tend to be less desirable because the calibration is usually worse than what you can do on your own, and you wind up with unrepeatable results. Keep your analog and digital grounds and supplies separate, and remember that some of these sensors are millivolt-level, so you need to pay special attention to signal routing in the analog areas.

Make sure each sensor has a built-in temperature sensor, or be sure to mount one near each sensor yourself. Additionally, make sure the sensors are either ratiometric (the output scales with the supply voltage), or make sure you do the scaling yourself in software by reading a voltage reference (V_{REF}) provided by the sensor. This detail cannot be stressed enough, or you will wind up with sensors that seemingly produce unrepeatable results over temperature.

Your microcontroller must do quite a bit of floating-point math and run the algorithm at the fastest rate possible. I've had success with 4-MHz processors running with a sample rate as slow as 36 Hz and with faster processors running up to 100 Hz and beyond. You need to do some benchmarking to find the perfect fit for your application, but make sure floating-point is fast. This simple choice will enable you to avoid endless nights of hand-optimizing code and writing in assembly language.

As you might imagine, the complexities arise not in the schematic, but rather in the algorithm that fuses the various sensors into the final result: vehicle attitude. The algorithm I used models how older mechanical artificial horizon units work on full-sized aircraft. The final output from the algorithm is R, which is the rotation of the aircraft in world space. R is a quaternion, which is simply a rotation describing how to rotate from aircraft space into world space in what amounts to axis-angle form. If you love Euler angles, you can still take the quaternion and break it into yaw, pitch, and roll quite easily. But by keeping all internal calculations quaternion-based, you completely avoid the gimbal lock issue.

WHY QUATERNIONS?

Most of us are familiar with the usual Euler angles yaw, pitch, and roll to express the orientation of a vehicle. Unfortunately, this simplicity comes at the cost of something known as *gimbal lock.* This occurs during situations such as one axis aligning to another and causing rotations about the others to become ambiguous. To avoid this, another option for expressing rotations is a 3×3 matrix of orthogonal vectors. Such a matrix needs to be renormalized and orthogonalized frequently or it will become degenerate. Quaternions have none of these drawbacks yet seem to suffer from the stigma of being too complex to be worth the trouble.

Although quaternions *can* be a rather

complex subject, they can be explained quite simply as an axis-angle rotation. Quaternions can take a vector (X, Y, Z) in one coordinate space and put them into another, or the inverse of the same quaternion can do the opposite. A quaternion is expressed as a four-vector (W, X, Y, Z), where X, Y, Z is the axis of rotation whose magnitude is the sine of half the rotation and W is the cosine of half the rotation. That's it. It's basically axis-angle. The only operations you need to do with quaternions are multiply (concatenation of rotations), vector rotate (rotate a vector using a quaternion), normalization (which keeps the quaternion from becoming degenerate), and invert. The pseudocode for these operations is expressed in Listing 1.

Listing 1—Quaternion and vector math pseudocode

Function Quat_Multiply (Q_1 , Q_2)
Resulting quat will act as if it does Q_2 first then Q_1

```
A = (Q1.W + Q1.X)*(Q2.W + Q2.X)
B = (Q1.Z - Q1.Y)*(Q2.Y - Q2.Z)
C = (Q1.W - Q1.X)*(Q2.Y + Q2.Z)
D = (Q1.Y + Q1.Z)*(Q2.W - Q2.X)
E = (Q1.X + Q1.Z)*(Q2.X + Q2.Y)
F = (Q1.X - Q1.Z)*(Q2.X - Q2.Y)
G = (Q1.W + Q1.Y)*(Q2.W - Q2.Z)
H = (Q1.W - Q1.Y)*(Q2.W + Q2.Z)
RESULT.W = B + (-E - F + G + H) * 0.5
RESULT.X = A - (E + F + G + H) * 0.5
RESULT.Y = C + (E - F + G - H) * 0.5
RESULT.Z = D + (E - F - G + H) * 0.5
```

Function Quat_RotateVector(Q, V)
Rotates vector V by quat Q

```
QV.X = V.X, QV.Y = V.Y
QV.Z = V.Z, QV.W = 0
IQ = Quat_Invert(Q)
R =  Quat_Multiply (Q,  QV)
RESULT =Quat_Multiply(R, IQ)
```

Function IMU_IntegrateGyros(dt, RATES, R)
Euler integrates RATES into quat R

```
rads_sec = vector_length(RATES)
axis = vector_normalize(RATES)

q_dot = Quat_SetAxisAndAngle(
        axis, rads_sec * dt)

R = Quat_Multiply(R, q_dot);
RESULT = Quat_Normalize( R );
```

Function Quat_Invert(Q)
Inverts the given quaternion

```
N = Q.X * Q.X + Q.Y*Q.Y +
    Q.Z * Q.Z + Q.W * Q.W

S =  (1.0 / N)

RESULT.X = -Q.X * S
RESULT.Y = -Q.Y * S
RESULT.Z = -Q.Z * S
RESULT.W =  Q.W * S
```

Function Quat_Normalize(Q)
Normalizes the given quaternion

```
N = Q.X * Q.X + Q.Y*Q.Y +
    Q.Z * Q.Z + Q.W * Q.W

if  N > 0.0
    S = 1.0 / sqrt(N)
else
    S = 1

RESULT.X = Q.X * S
RESULT.Y = Q.Y * S
RESULT.Z = Q.Z * S
RESULT.W = Q.W * S
```

Function Quat_SetAxisAndAngle(AXIS, ANGLE)
Creates a quaternion from a unit length axis and angle in radians

```
HALF_ANGLE = 0.5 * ANGLE
S = sin(HALF_ANGLE)

RESULT.X = AXIS.X * S
RESULT.Y = AXIS.Y * S
RESULT.Z = AXIS.Z * S
RESULT.W = cos( HALF_ANGLE )
```

Function vector_length(V)
Returns the length of the given vector

```
RESULT = sqrt( V.X * V.X +
         V.Y *V.Y + V.Z * V.Z)
```

Function vector_normalize(V)
Scales the vector to make it unit length

```
LENGTH = vector_length( V )
S = 1.0 / LENGTH;
RESULT.X = V.X * S
RESULT.Y = V.Y * S
RESULT.Z = V.Z * S
```

Function vector_cross(A, B)
Returns the vector cross product of A x B

```
RESULT.X = A.Y * B.Z - A.Z * B.Y
RESULT.Y = A.Z * B.X - A.X * B.Z
RESULT.Z =  A.X * B.Y - A.Y * B.X
```

Function vector_dot(A,B)
Returns the dot product of A . B

```
RESULT =
   A.X * B.X + A.Y * B.Y + A.Z * B.Z
```

VOLTS TO UNITS

Taking sensor voltages and obtaining engineering units (e.g., radians/sec and 3-D vectors) is done during the "scale/offset correction" step. I'll cover the details later. For now, just assume your gyros output a 3-D angular velocity vector, your magnetometer outputs a 3-D unit length vector, and your accelerometer outputs a 3-D vector in Gs.

For the rate gyros, we take the independent X, Y, and Z rotation rates, and place them into single 3-D vector, obtaining the same angular velocity vector commonly used in physics texts. The vector direction is the axis of rotation, and the magnitude is the rate of rotation in radians/sec.

The magnetometer units are irrelevant for this application. You simply normalize this vector to obtain a 3-D "North" of unit length. The accelerometer measures G force magnitude and direction, where (0, 0, 1) is the IMU sitting completely level in a static state. (In aerospace, the positive Z-axis is down.) The accelerometer in this application acts like a pendulum and generally points down and swings slightly outward in a turn.

THE ALGORITHM

Start with the rate gyroscope triad. Mathematically, you simply take these components as a single angular velocity vector and integrate them into the output quaternion R. Ideally, this would be the only step and you'd be done; however, even the best gyros have random walk or "drift." Over time, errors build up and this estimate will move off in a random direction and won't be much of an estimate. This is where the mechanical artificial horizon units do what is called an "erect", which simply means they erect their output estimate by effectively allowing gravity to pull the estimated "down" towards actual "down" over time, where down is assumed to be where gravity is coming from. You can simulate this same operation with your accelerometer. Using this sensor, you get a 1-G vector due to the acceleration of gravity, and you can slowly rotate your estimate over time so the estimated down is really pointing down. For my IMU, this correction rate is in the neighborhood of

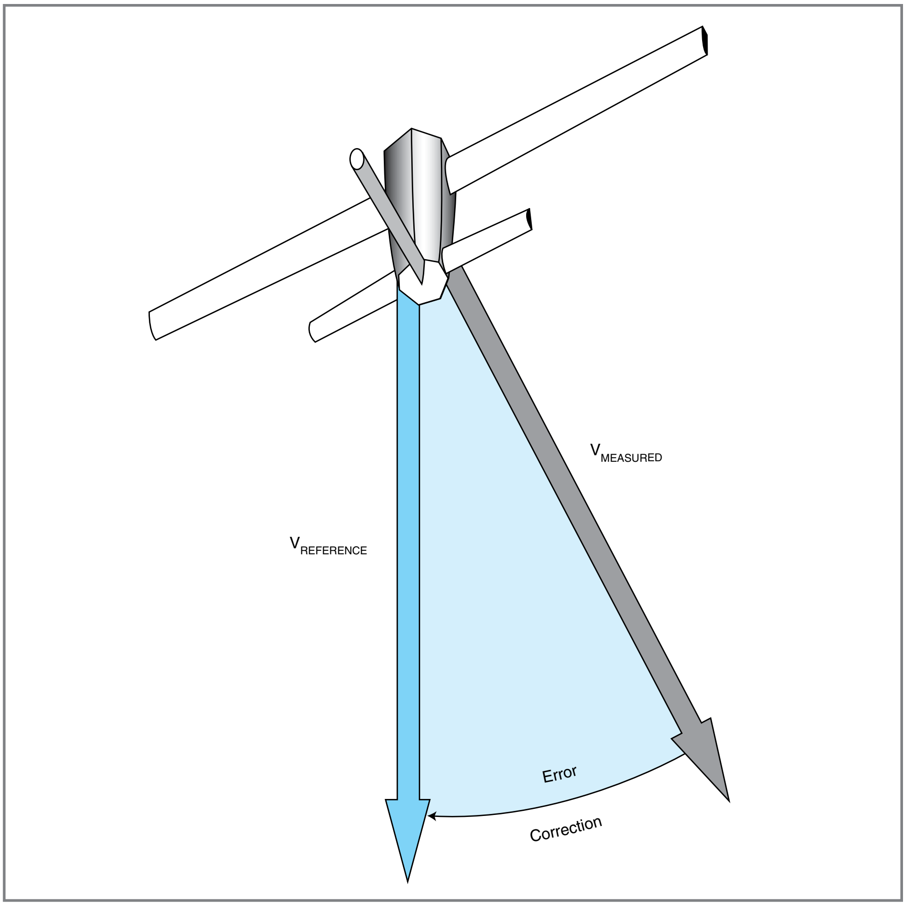


Figure 5—A hypothetical error in roll and pitch. The aircraft is actually flying straight and level, but the IMU has an incorrect estimate. Gravity is supposed to be coming from "$V_{REFERENCE}$," but it is actually coming from "$V_{MEASURED}$." We correct our estimate by rotating such that $V_{MEASURED}$ moves toward $V_{REFERENCE}$.

0.3°/sec, which is just above the offset error and drift rate.

To make this a bit clearer, Figure 5 shows a hypothetically incorrect attitude estimate, incorrectly showing the aircraft in a bank due to gyro drift. In

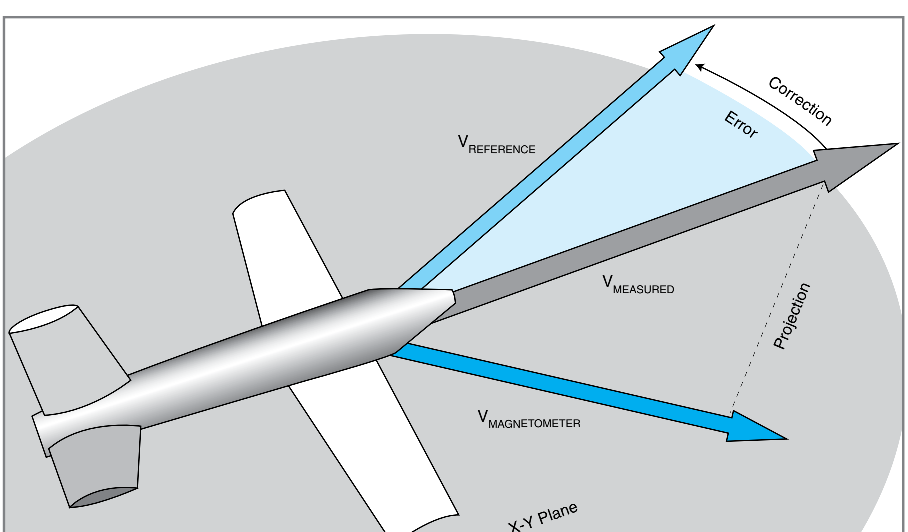


Figure 6—A hypothetical error in yaw. $V_{REFERENCE}$ is pointing "north," yet we're measuring "north" from a different direction. We once again correct our estimate by rotating such that $V_{MEASURED}$ moves toward $V_{REFERENCE}$, but this time only in yaw, or about the z-axis.

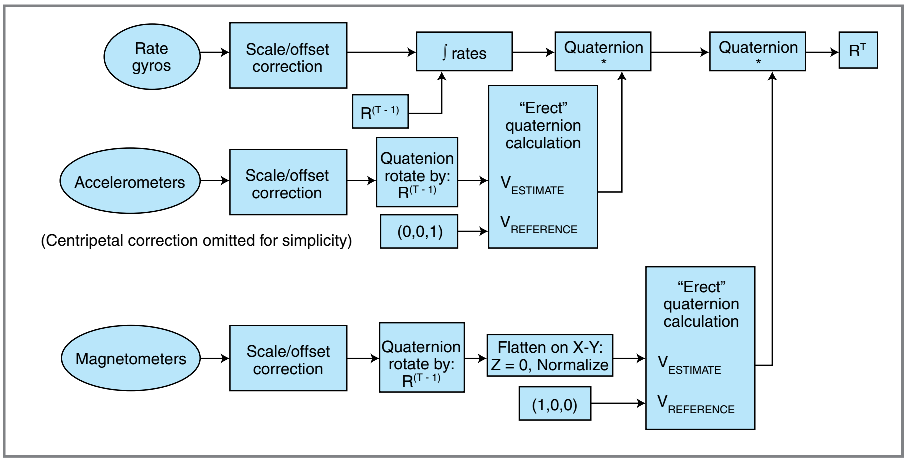


Figure 7—The FreeSpace IMU algorithm. Gyros are integrated into a drifting estimate. Accelerometers produce an "erect" quaternion to correct for drift on pitch/roll. Magnetometers produce an "erect" quaternion for correcting drift on yaw. The output is a quaternion representing orientation in world space.

this example, the aircraft is, in reality, flying straight and level; however, the IMU has it placed at a bogus bank angle. If you retrieve the vector from the accelerometer and run it through the estimated attitude R and into world space, you obtain your estimated "down vector" ($V_{MEASURED}$). Notice it isn't pointing down at all. You know, however, that this vector really should be coming from down (0, 0, 1), so the angle difference between the two is your error. Over time, you will correct, or "erect," your estimate R toward making these vectors match, and making sure your correction rate is faster than gyro drift.

It is probably worthwhile to note at this stage that the accelerometer not only measures gravity, but also any centripetal accelerations due to perhaps a banking turn. In the short-term, this could present a problem, however, since the corrections are slow enough, and because the average accelerometer output over the long term *does* generally reflect gravity exclusively. In practice, this is not as much of a concern as you might think. However, if centripetal accelerations are frequent enough, of long duration, and repeatedly in the same direction, this will cause the average accelerometer vector to point at a slight angle. Fortunately, if velocity is available (e.g., from a GPS or pitot tube), one can compute the centripetal accelerations from the following equations and subtract them out from the accelerometer:

Centripetal acceleration = Linear velocity × Angular velocity

Note that this is vector math with X, Y, Z components. It is simply the cross product between linear velocity and angular velocity in IMU coordinate space. The resulting quantity then can be directly subtracted from the accelerometers in order to eliminate this effect and save a bit of error. After doing this additional work, you end up with accelerometers that reflect only acceleration due to gravity, and perhaps a small error, but accuracy will be significantly improved.

The "erect" operation using the accelerometer will take care of drift on pitch and roll; however, the estimate will still drift in yaw. This is where our magnetometer comes into play. The magnetometer provides a 3-D vector pointing along Earth's magnetic field toward North. Most of us are used to the usual 2-D compass bearings; however, it will surprise some that in most parts of the United States, this vector is not horizontal, but mostly vertical, pointing nearly straight down! Using this vector, you'll perform a similar "erect" operation, but on yaw, as shown in Figure 6. In this example, the aircraft thinks it's slightly yawed to the East, as shown. But in reality, it is pointing North along $V_{REFRENCE}$. You take the value retrieved from the magnetometer, use the estimate quaternion R to put this into world space, and obtain $V_{MAGNETOMETER}$. We then project this vector to the X-Y plane by zeroing the Z value and renormalizing to obtain the world space estimated North vector $V_{MEASURED}$. You know North should be coming from $V_{REFRENCE}$. So, as before, the angle difference between the two is our error. At a rate faster than drift, you correct your estimate R toward making these vectors match by rotating R on the Z-axis. Over time, this operation will minimize any detected error in yaw. The time constant needed will depend on the drift rate of your gyros, as well as how good of a job you do calibrating them. For example, if your gyros drift

Listing 2—This calculates the quaternion to rotate the measured vector to the reference vector at a fixed correction rate.

```
Function IMU_GetErectorQuat( Vmeasured, Vreference, RADS_SEC, DT )
Calculates the quaternion needed to rotate the measured vector to the
reference vector (world space) at a fixed correction rate.

// Get the normalized rotation axis we'll use to erect.
C = vector_cross( Vmeasured, Vreference)
C = vector_normalize(C)
// Get the angle between the two vectors, and clamp to that limit.
rads = RADS_SEC * DT;
a = abs(clamped_acos( vector_dot( Vmeasured, Vreference)))
IF rads > a THEN rads = a
// Get the quat that rotates our sensor Vmeasured toward the
Vreference vector by the specified amount
RESULT = Quat_SetAxisAndAngle( C, rads);
```

Listing 3—High-level pseudocode that keeps the estimate R updated

```
Function IMU_UPDATE( R, Gyros, Accels, Mags, dt )
Update our vehicle attitude R with new calibrated sensor values.
This is the "high level" update.

  // Euler Integrate our angluar rate gyros into R
  R = IntegrateGyros( dt, Gyros, R)

  // Get the down vector in world space.
  Vdown = vector_normalize(Accels)
  Vmeasured = Quat_RotateVector( R, Vdown)

  // Get erector quaternion from measured and "reference" down.
  Qerector = GetErectorQuat(  Vmeasured, [0,0,1] , dt)

  // Apply the erector rotation to our estimate
  R = Quat_Multiply( Qerector, R )

  // Get the north vector in world space, flattened
  Vmeasured = Quat_RotateVector( R,  Mags)
  Vmeasured.z = 0;
  Vmeasured = vector_normalize( Vmeasured )

  // Get erector quaternion from measured and "reference" north.
  Qerector = GetErectorQuat(  Vmeasured, [1,0,0] , dt )

  // Apply the erector rotation to our estimate
  Result = R = Quat_Multiply( Qerector, R )
```

randomly at 0.2°/second and your bias is off by a maximum of 0.3°/second over temperature, you'd want to correct at least as fast as 0.5°/second plus a safety margin. I tend to be conservative and double this value; and in this case, the erecting would occur at a constant speed of 0.6°/second.

For review, the block diagram of this whole process is outlined in Figure 7. Each sensor is fed through Scale/Offset tables, which yield calibrated engineering units you can work with. The rotation velocity vector provided by the gyros is then integrated into your estimate quaternion R using the computed rotation from the last time step R^{T-1} and the current angular velocity. This produces an attitude estimate that has drift from the gyro's random walk and calibration errors. Next, you compute two erector quaternions, which are then multiplied into this estimate to produce the final, corrected estimate for this time step. The first erector quaternion is produced by calculating the rotation that will move between your estimated "down" vector and the real down vector in world space. This is then multiplied into the estimate. The second "erector" quaternion takes the X-Y flattened estimated world space "North" vector and real North (1, 0, 0), and it also computes a rotation that will move from the estimate toward the real vector. Again, these operations take place in world space. After the corrections are applied, you end up with R, your estimated orientation quaternion.

BLOCKS

Integrating a rotational velocity vector from the gyros into a quaternion using Euler integration is relatively straightforward. You first generate a Euler "time step" quaternion whose rotation axis is the angular velocity vector, and whose angle is the rotation amount in radians/second scaled by the time step *dt*. Next, you multiply the current estimate by this new quaternion, integrating the current time step into the estimate. The Listing 1 function `IntegrateGyros` shows the pseudocode for this operation.

As you can see in Figure 7, the block is used twice. It simply takes a "measured" vector from a sensor in world space, and a "reference" vector in world space, and computes a quaternion that would rotate the "measured" vector towards the "reference" vector.

Quaternions are axis-angle rotations, so first you need to compute the axis which would rotate from one vector to the other. This is done with the cross product of the vector portion (X, Y, Z) of the two, producing a perpendicular vector. The magnitude of the rotation about this vector is fixed at the desired correction rate. For example, if we want to do a 1°/second correction factor, the quaternion produced only rotates 1° about this axis, scaled by time step *dt*. Be careful not to overshoot, so the angle between the two is calculated with the arc-cosine of the dot product of the vectors and the rotation clamped to be below this threshold. We also only compute positive angles, since the rotation vector already flips depending on the direction of rotation. The pseudocode algorithm is in Listing 2 function `GetErectorQuat`.

PUT IT TOGETHER

The high-level pseudocode, which keeps the estimate R updated, is shown in Listing 3. Your program simply needs to call the `imu_update` function repeatedly with calibrated sensor vectors (again, each has X, Y, and Z components), the timestep dt, as well as the estimate quaternion R to be updated by the function. Units are as we discussed: gyroscopes are in radians/second, accelerometers are in Gs, and the magnetometer is a normalized vector pointing North in 3-D. Note that while "reference" world space down is always the Z-axis [0, 0, 1], "reference" North can be either purely the X-axis [1, 0, 0] or true magnetic North with your local magnetic declination [cos(decl), sin(decl), 0].

That's really all there is too it. Chances are you'll find this process much easier to visualize and debug than a huge matrix with nonintuitive coefficients.

CALIBRATION

Many papers seem to gloss over calibration, and I found this to be the single most critical aspect in determining the accuracy of the IMU. It also happens to be the most boring. Each sensor must be calibrated for scale and offset over the operating temperature range. This simply means, for example, that the gyros must output 0°/second when there is no rotation, and temperature should have no scaling effect on the output. The formula I use to produce meaningful, temperature-invariant output from each sensor is simply:

$$\text{Value} = (\text{sensor\_output\_voltage} + \text{OFFSET\_TABLE[temperature]}) \times \text{SCALE\_TABLE[temperature]}$$

The offset and scale on a particular sensor tend to change over temperature, so we need to generate a look-up table we can use. For a given temperature, we need to look up offset and scale and apply. Fortunately, generating this table is a one-time operation done only after the initial build of a new IMU, or sometimes after you drop it.

The rate gyroscopes are the most error-sensitive sensors. Since you will be integrating the output, it is paramount that error be as low as possible. The other sensors are fairly forgiving and may not need calibration if temperature tests demonstrate they don't have error above what is tolerable for your specific application. I had cheap accelerometers that varied 20% over temperature, so punting on calibration was not an option for me.

For rate gyro calibration, my methods are caveman-centric but effective. To make this process easier, I begin by mounting my IMU in a cube. This simply allows me to rotate it with ease on each primary axis (X, Y, or Z). I also place a thermal mass brick in my kitchen freezer for an hour or so. Once cold, I then place the brick on a hot plate inside an ice cooler, followed by placing the IMU cube on the brick and close the lid on the cooler. Then the fun begins.

I let the unit warm up to room temperature over many hours, followed by switching on the hot plate to further heat up the unit to the maximum expected operating temperature. The goal here is to simply expose the IMU to all expected operating temperatures slowly over time. The temperature starts off low with the cold brick and gradually heats up. During the ramp-up, the IMU records an average voltage from all sensors at each temperature. The specific speed of the temperature ramp-up is not important, but should be as slow as practical in order to get the "center" of the random walk for each temperature sampled. For my setup, I tended to run overnight, or roughly 8 hours, for the gyros, and faster for the other less-critical sensors.

For the accelerometer and magnetometer, each axis needs two of these temperature runs per axis. Run the calibration on each axis with maximum positive and negative inputs. For example, if your IMU has the Z-axis running normal to the PCB (up and down), do the test once with the IMU resting flat and upright, and the second test with the IMU upside down. For each temperature, two voltages result from these two runs. The average of the two is the offset, and half the difference between the two is the scale.

The magnetometer is slightly more difficult because the magnetic vector doesn't come from a convenient direction. Rotate the unit until you have the highest reading from the sensor and calibrate each axis the same way as before.

The rate gyroscopes need slightly special treatment. Offset calibration is the same as for the other two sensors, but scale is not. For scale, it is adequate to simply plug in a constant value from the datasheets if this quantity doesn't vary significantly. However, if you need the extra accuracy as I did, a fixed rotation rate turntable is one solution. After calibrating for offset, place the IMU on a turntable rotating at a fixed rate (I used 1 radian/second) and run a calibration over temperature. The values you get after this run are the scale factors. Don't think for a second that this requires expensive equipment. My solution was a "Lazy Susan" with an attached stepper motor and rubber wheel. This works, providing your sensors are linear and most try to be.

As a check, rerun these tests with your calibration tables active. Make sure everything is flat as temperature varies and your gyros have a zero-rotation offset of near 0 radians/second. If all goes well, calibration takes the temperature variant sensor voltages and gives engineering quantities you can work with.

A final consideration for the magnetometers should be mentioned at this point. Once you install your IMU in a particular vehicle, perform an additional scale/offset calibration. This can be achieved by rotating the IMU in all directions while recording the min/max values on each axis. From these, compute a final scale and offset. During operation, apply this after the temperature calibration table. This removes any distortions in the magnetic field caused by the vehicle itself.

Mounting sensors orthogonally within a degree or less is the best way to avoid any need to do a "cross coupling" calibration. It is possible to calibrate out any misalignment of sensors due to the sensitive axis not being precisely aligned in software. However, I've found that the easiest solution is to simply design your PCB to mechanically ensure this aliment. Since the gyro drift with modern MEMS gyros is high enough, in practice I've found this error is lost in the noise.

As if all this weren't enough, MEMS gyroscopes tend to be sensitive to accelerations. You can squeeze out some additional accuracy by recording the offset shift of the gyro outputs caused by 1 G of acceleration on each axis. You can then take these offsets and multiply by the accelerometer vector (in Gs) and subtract the result from the gyro vector, which will subtract this effect out. This calibration is probably worthwhile once you have everything working and need a bit of extra accuracy.

The ultimate level of calibration is entirely up to you. I suggest starting with little to no calibration and adding complexity in this area as you achieve success.

ALGORITHM SUCCESS

While the subject of IMU fusion algorithms can be complex, I believe this is the simplest and easiest to understand implementation to date. Most importantly, it has been proven to work in the high dynamic environment encountered on my own UAV. I have also successfully used this algorithm to create an autopilot able to take off, fly to waypoints, and land completely autonomously.

I hope this helps you with your drone aircraft (UAV, RPV, MAV, and whatever acronyms they think of next), underwater exploring rover, blimp, or other interesting robotics projects. Just try to refrain from using these ideas in any plans for world domination. The up-front work has certainly paid off in the level of control I now have over the inner workings of my project. Plus, I didn't have to pay hundreds of dollars per unit for someone else's module. Hopefully, it will help you in your endeavors.

TJ Bordelon (tj@bordelon.net) lives in Natick, MA, and is the Director of Core Technologies for the video game company High Impact Games. He enjoys writing low-level, high-performance code for challenging applications, such as high-performance physics and 3-D graphics engines used in cutting-edge games. TJ consulted for several UAV companies and helped develop the IMU firmware and ground station for Continental Controls and Design's Locust UAV. When he's not writing or playing games, TJ is usually in his garage working on his own autonomous UAV project or preferably spending time with his family.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/238.

RESOURCES

J. C. Hart, J. K. Francis, and L. H. Kauffman, "Visualizing Quaternion Rotation," *ACM Transactions on Graphics*, Vol 13, No 3, 1994.

A. Hanson, "Visualizing Quaternions," *The Morgan Kaufmann Series in Interactive 3D Technology*.

SOURCES

AD7799 ADC and gyros
Analog Devices, Inc. | www.analog.com

HMC1043 Magnetometer
Honeywell | www.ssec.honeywell.com/magnetic/

dsPIC33FJ256GP506 DSC
Microchip Technology, Inc. | www.microchip.com

CHIMU AHRS
Ryan Mechatronics | www.ryanmechatronics.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*.

For more need-to-know information about topics covered in TJ Bordelon's Issue 238 article, the *Circuit Cellar* editorial staff highly recommends the following content:

—

Inertial Rolling Robot
by Jeff Bingham and Lee Magnusson
***Circuit Cellar* 200, 2007**

This H8/3664-based rolling robot is capable of inertial movement. A DC electric motor is attached to a pendulum and suspended inside an inflated ball. Topics: Robot Navigation, Inertial Measurement, Sensors, Velocity

Go to: www.circuitcellar.com/magazine/200.html

—

Mini Rover 7
by Joseph Miller
***Circuit Cellar* 165, 2004**

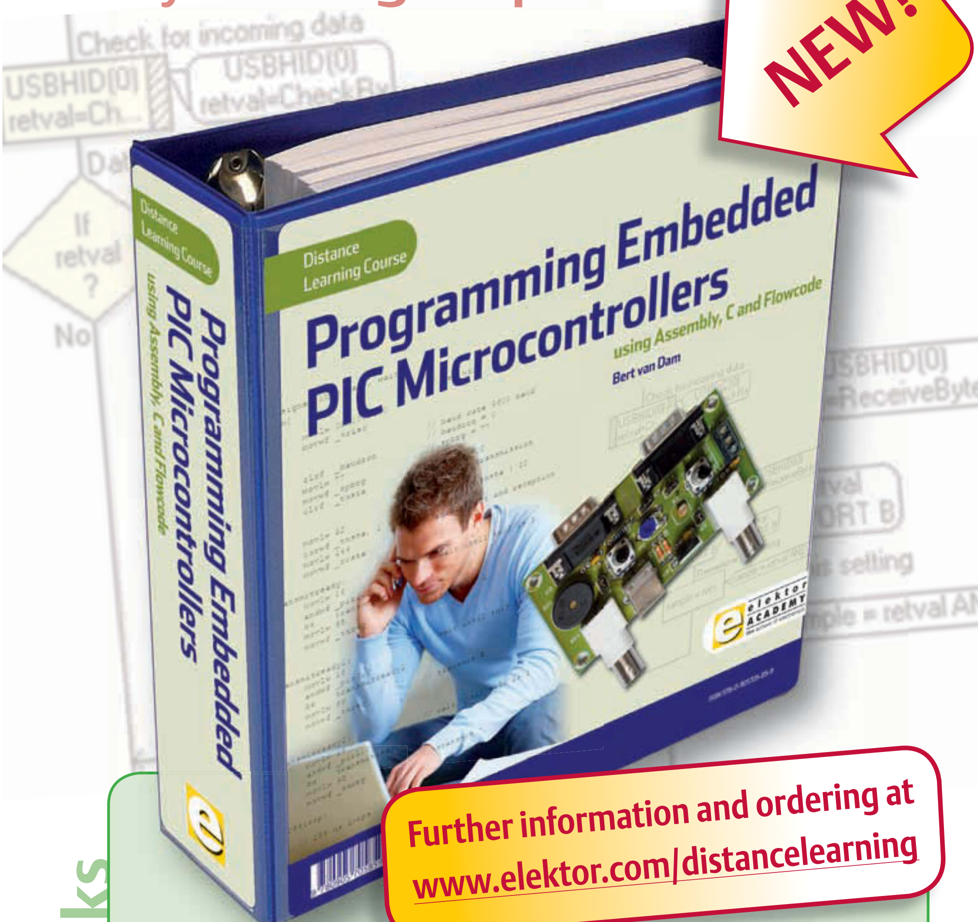
You can use electronic compassing to obtain heading info for a robotics application. This design was modeled after the NASA/JPL Rocky 7 Mars rover. Topics: Robot Navigation, Measurement, Sensors, Velocity

Go to: www.circuitcellar.com/magazine/165toc.htm

Elektor Shop

The world of electronics at your fingertips!

NEW!



Further information and ordering at www.elektor.com/distancelearning

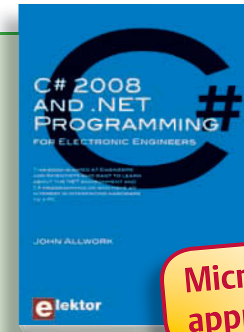
Books

Distance Learning Course

Programming Embedded Microcontrollers

In this course you will learn how to program an embedded microcontroller. We will start with the absolute basics and we will go into a lot of detail. You cannot learn about software without understanding the hardware so we will also take a close look at the components and schematics. Gradually the course will become more complex as we move into the more advanced subjects. At the end of the course you will be able to design your own embedded applications and write the appropriate software for it. From a flashing LED to a USB controlled LCD screen, from analog signals to interrupt driven high frequency sound, from frequency sampling to a digital thermometer, you will master it all. Your course package consists of a Courseware Ring Binder, a CD-ROM including software and example files and a special application board. On-line support is available at the Elektor Forum.

747 pages • Art.# 15033 • $645.00



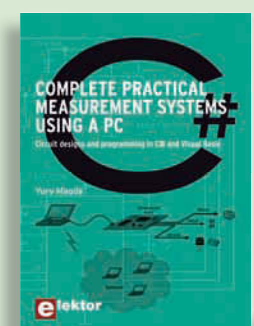
Microsoft approved!

Learn more about C# programming and .NET

C# 2008 and .NET programming for Electronic Engineers

This book is aimed at Engineers and Scientists who want to learn about the .NET environment and C# programming or who have an interest in interfacing hardware to a PC. The book covers the Visual Studio 2008 development environment, the .NET framework and C# programming language from data types and program flow to more advanced concepts including object oriented programming.

240 pages • ISBN 978-0-905705-81-1 • $47.60



Circuit design and programming

Complete practical measurement systems using a PC

This book covers both hardware and software aspects of designing typical embedded systems based on personal computers running the Windows operating system. With an emphasis on learning by doing, readers are encouraged by examples to program with ease; the book provides clear guidelines as to the appropriate programming techniques "on the fly". Complete and well-documented source code is provided for all projects.

292 pages • ISBN 978-0-905705-79-8 • $46.00

Prices and item descriptions subject to change. E. & O.E

CD/DVD-ROMs

See the light on Solid State Lighting

DVD LED Toolbox

This DVD-ROM contains carefully-sorted comprehensive technical documentation about and around LEDs. For standard models, and for a selection of LED modules, this Toolbox gathers together data sheets from all the manufacturers, application notes, design guides, white papers and so on. It offers several hundred drivers for powering and controlling LEDs in different configurations, along with ready-to-use modules (power supply units, DMX controllers, dimmers, etc.). In addition to optical systems, light detectors, hardware, etc., this DVD also addresses the main shortcoming of power LEDs: heating. This DVD contains several Elektor articles (more than 100) on the subject of LEDs.

ISBN 978-90-5381-245-7 • $46.00

110 issues, more than 2,100 articles

DVD Elektor 1990 through 1999

This DVD-ROM contains the full range of 1990-1999 volumes (all 110 issues) of Elektor Electronics magazine (PDF). The more than 2,100 separate articles have been classified chronologically by their dates of publication (month/year), but are also listed alphabetically by topic. A comprehensive index enables you to search the entire DVD.

ISBN 978-0-905705-76-7 • $111.30

A whole year onto a single disk

DVD Elektor 2009

The year volume DVD/CD-ROMs are among the most popular items in Elektor's product range. This DVD-ROM contains all editorial articles published in Volume 2009 of the English, American, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favourite graphics program, zoom in /out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-251-8 • $28.30

**Elektor is more than just your favorite electronics magazine.
It's your one-stop shop for Elektor Books, CDs, DVDs, Kits & Modules and much more!**

www.elektor.com/shop

elektor

Elektor US
PO Box 180
Vernon CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: sales@elektor.com

Kits & Modules

Winamp Controller

A variety of remote control devices for Winamp and other PC-based media players have been available for a good while. This project implements a physical progress bar for Winamp. Here a small ATmega microcontroller uses the USB interface to provide a bidirectional link between the Winamp software and a hardware studio fader, which acts as a combined indicator and entry device.

Kit of parts, including PCB

Art.# 090531-71 • $143.60

Software Defined Radio

SD radio receivers use a bare minimum of hardware, relying instead on their software capabilities. The Elektor SDR project (by Burkhard Kainka) demonstrates what's achievable, in this case a multi-purpose receiver covering all bands from 150 kHz to 30 MHz. It's been optimised for receiving DRM and AM broadcasts but is also suitable for listening in to the world of amateur transmissions. The designer's aim for this project was to create a receiver displaying high linearity and phase accuracy. Development was focussed on the characteristics that were most important for a top-notch DRM receiver and the end result is a receiver with remarkable interference rejection characteristics!

Ready-populated and tested board

Art. # 070039-91 • $139.60

Home Automation for an Energy-Efficient House (Part 1)

System Design and Server Software

A 21st-century energy-efficient house requires more than extra insulation and a solar panel kit. A home control system based on some serious embedded hardware is essential. This series details how to build a home automation system based on a CAN bus running a CANOpen software stack. Here you learn about the overall design and server software.

by Stefan Siegel (USA)

During the process of designing an energy-efficient house, it became clear that the house would need control features that were not available in any off-the-shelf commercially available home automation system. While industrial factory control systems certainly would've been up to the task, their sheer cost proved prohibitive. This prompted me to design my own system from scratch, which is based on a controller area network (CAN) bus running a CANOpen software stack. It consists of more than 80 embedded microcontroller nodes distributed around the house (and connected to each other) and a Linux server. The first part of this article covers some details of my energy-efficient house's features, the resulting automation requirements, the overall system design, and the server software. In the second part, I'll cover the hardware and firmware of one of the embedded nodes in detail.

LOCATION

When my wife and I set out to design and subsequently build a new house for our family of five in Pueblo, Colorado, we decided from the get-go that it would have to be far more energy-efficient than the conventional, stick-frame and plaster board house that we lived in at the time. From a "big-picture" perspective, the most energy use in a home is for heating and cooling; thus, this is the most obvious area to address if you want to reduce energy consumption. Our goal was to design and build an energy-efficient house that would be well-adjusted to the landscape and climate in southern Colorado. The

Photo 1—An exterior view of our house, taken from the southeast. Note the passive solar design features: the large south-facing windows, the roof overhang and shade between the ground and first floor, as well as the sliding exterior shutters on the east-facing windows.

area gets warm in the summer and quite cold in the winter. It has little precipitation and cloud cover throughout the year. We have more than 330 sunny days per year in Pueblo, so it is a high desert climate with dry air, which features wide temperature fluctuations (sometimes even within a single day).

PASSIVE SOLAR & THERMAL MASS

To address the large temperature swings from day to night, the house was designed with a large thermal mass that keeps the temperature inside mostly constant without any additional cooling or heating. This is implemented in the form of both concrete floors and concrete exterior walls, which are insulated by 3″ of closed cell foam to the outside. We make use of the plentiful sunshine using large windows on the south side, and very minimal windows on the north side of the house.

To store the sun's heating power during the day and release this heat at night, a trombe wall is used on most south-facing windows. A trombe wall is in essence a wall that will be heated by the sun during the day and then release this heat to the inside during the night. Photo 1 shows the trombe walls behind the large glass front. Roof overhangs of 3′ (90 cm) on the south side keep the sun out during the hot summer months when the sun is high in the sky, while allowing it to heat the house during winter when the sun only rises slightly above the horizon. This passive solar design covers most of the heating needs and prevents tremendous amounts of cooling load with little or no added construction cost.

EXTERIOR SLIDING SHUTTERS

While the roof overhangs shield the south facing windows, on the east and west the building would experience tremendous heat gains in summer through these windows; in fact, far more than the currently installed capacity of the cooling system! Thus, the plan from the get-go was to keep the sun out using sliding shutters also seen in Photo 1. As these are larger and heavier than any commercially available drive system, I had to design hardware, motorization and control electronics for these from scratch. The automation goal here is to close them during cooling season and while there is direct sun hitting them, and open them during the heating season to get the passive solar heating gains.

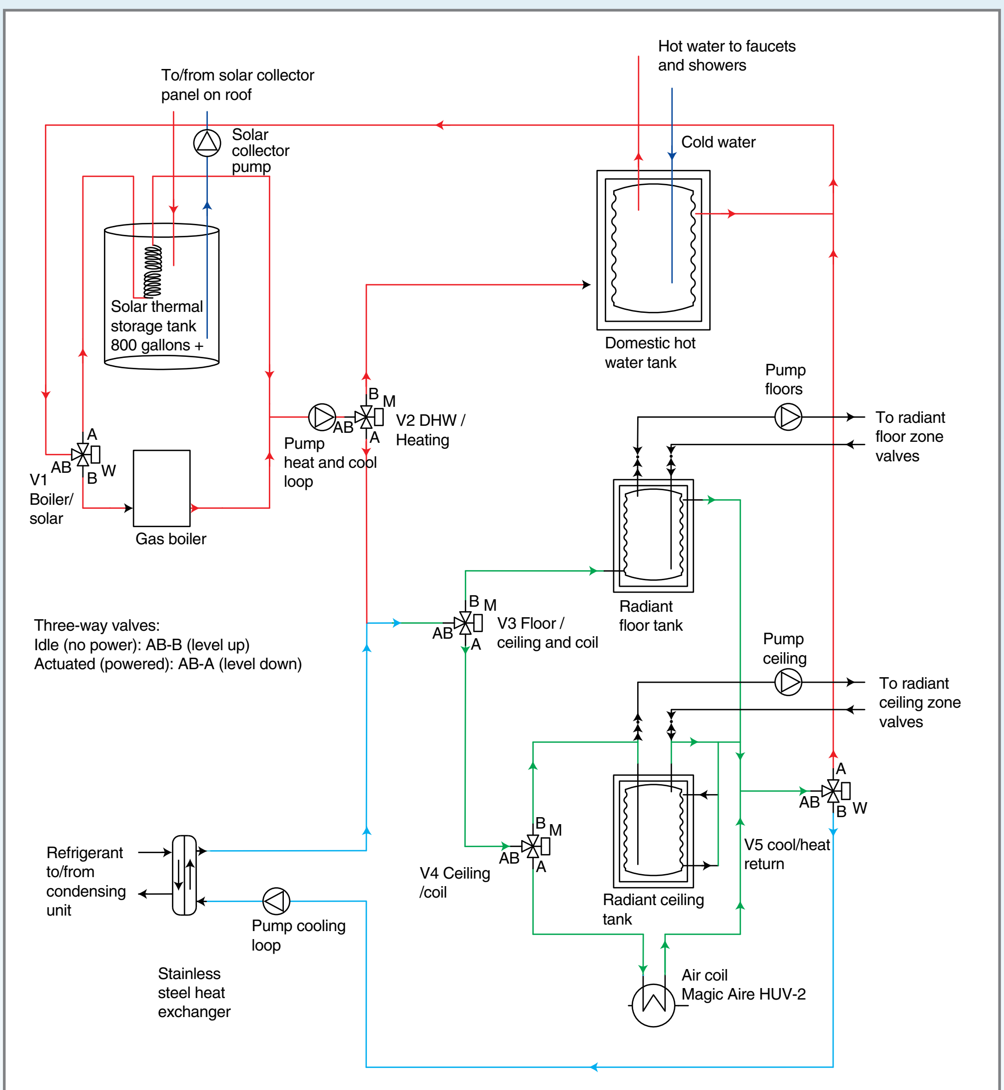


Figure 1—A diagram of the heating and cooling system's plumbing. Red pipes carry hot water, cyan pipes carry chilled water, and green pipes can carry either hot or chilled water depending on the season and the system's operating mode.

RADIANT HEATING & COOLING

Even the best passive design including smart shutter control will meet its limits during extreme hot and cold spells. In winter, a set of solar thermal collectors stores heat in a large storage tank during the day, which can then be used for additional heating with a radiant heating system, or to heat the domestic hot water for showers and more. This system is designed to cover all heating and hot water needs down to 0°F weather—as long as the sun shines. If not, an efficient gas-fired water boiler kicks in to fill the need for heating and hot water during overcast periods. To determine the exact heating and cooling loads, I used Energy Plus, a free software package from the U.S. Department of Energy that performs time-resolved simulations, not just of heat transfer by conduction, but also radiation through windows, which is essential for a passive solar design.

In summer, the daytime temperatures in Pueblo do exceed 100°F (38°C) regularly. Instead of an inefficient air-based HVAC system, we installed a radiant ceiling cooling system to cool the house. This system works just like a radiant floor heating system, only that the water is chilled instead of heated and circulates through the ceilings instead of the floors. It has similar advantages as a radiant heating system, which are better comfort (no air drafts) and higher efficiency (less cooling costs). Unlike radiant heating systems, this radiant cooling system is rarely used in residential buildings in the U.S., so a special permit had to be requested before implementing this feature in our house.

All of these active features require both sophisticated plumbing and control to operate. Figure 1 shows the plumbing located in the basement. On the left top, you see the solar storage tank, which is heated during the day by circulating water through the collectors with the solar collector pump. Temperature sensors at the collectors activate the pump when the water there is warmer than in the tank, and they turn it off when the set temperature has been reached—or there is no temperature increase anymore because the sun does not supply sufficient energy. The hot water in the storage tank then can be used to heat one of four heat consumers: the domestic hot water tank, a radiant ceiling tank, a radiant floor tank, or an air coil in the ventilation system. If the water in the solar tank is not hot enough for that task, the water in this heat and cool loop can be routed through the gas-fired boiler. During summer operation, an air-based refrigeration unit chills water through a refrigerant-to-water heat exchanger (bottom left in Figure 1). This water then can be used to chill the floor tank, ceiling tank, or the air coil in the ventilation system by setting the valves accordingly and using the cooling loop pump to circulate the water.

With temperature sensors in all tanks, and a number of pumps and three-way valves to be controlled, this setup requires some custom programming to operate reliably. The fact that none of the commercially available controllers were up to that task is what really started this project.

VENTILATION

An additional energy-saving feature in this house is an energy-recovery ventilation system. It serves three purposes in our house. One, it allows fresh air to enter without opening any windows, and it recovers both latent and sensible heat in that mode. Then, there needs to be a means of controlling humidity. As the cooling system does not dry the air like a conventional HVAC system, this functionality is provided by the ventilation system and an air coil. Last but not least, during cool summer

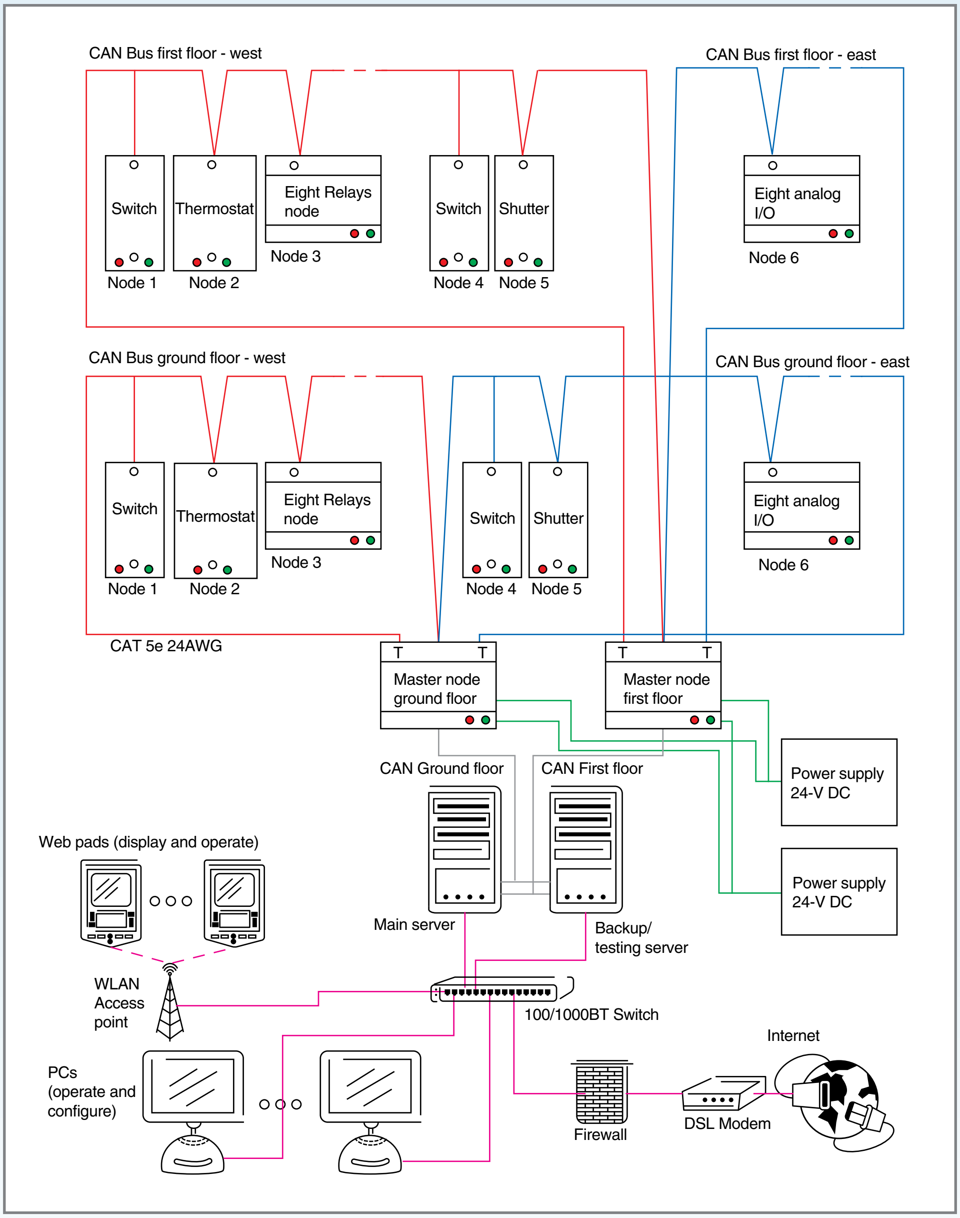


Figure 2—Home Automation hardware layout. Red/blue/gray lines denote CAN Bus wires. The pink and dashed pink are LAN and wireless LAN. The DC power supply lines are green.

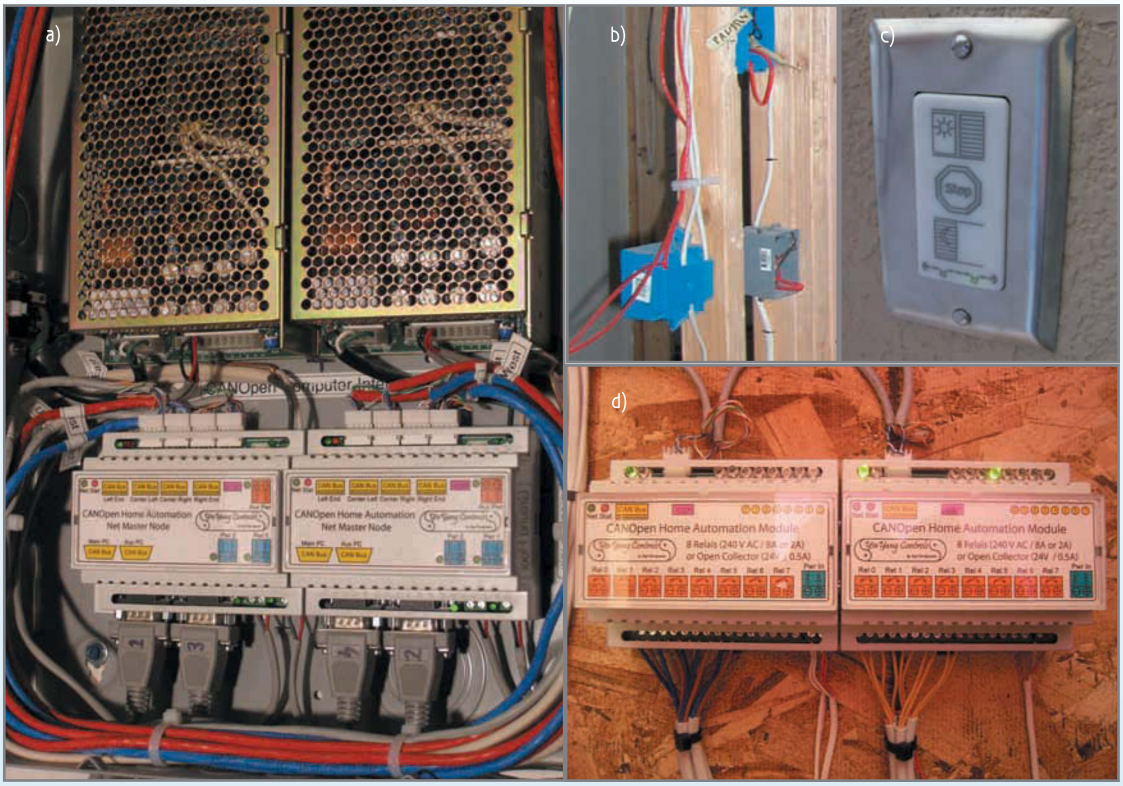


Photo 2—Here you see some of the installed home automation components. a—Two master nodes (bottom) and two power supplies (top) are installed in a steel enclosure. b—Wall mounting worked well. The bus wire (red) loops through three rough-in boxes. c—A shutter node is installed. d—These are two eight-port open-collector DIN rail-mounted nodes for zone valves.

nights, the ventilation system helps to cool the house by moving cool outside air into the house and exhausting the warmer air inside, thus cooling without any use of refrigeration.

The energy-recovery system we bought comes without any sophisticated controller. Thus, the home automation system provides this functionality as well.

AUTOMATION & CONTROL NEEDS

Beyond the shutters and the heating and cooling system in Figure 1, the design called for 14 heating and cooling zones, each with its own thermostat and hygrostat (to prevent condensation on the cooling ceilings) and actuators for radiant floors and ceilings. There are keypads for entry, control for lighting and essentials like the security system with motion detectors, glass break sensors and more. Lastly, there is a weather station on the roof that measures wind, rain, solar irradiation, and so on.

NETWORK HARDWARE

Given the size of the house and the number of different components, a traditional centralized system with wiring home runs from each item to be controlled was ruled out from the beginning. This leaves two choices: a wireless network or a wire-based network. Wireless is all the rage these days; but in my opinion, it is severely hampered by the fact that we have yet to invent wireless power supplies. Thus, I started looking into wired networks—the most popular of which is, of course, Ethernet. The problem is that this is a home run type of system (from each component to a router or switch), and it's relatively expensive to implement in terms of hardware components.

Software-wise, it also puts a high demand on resources in terms of memory and CPU power. I was looking for a simpler solution and found it with the CAN bus. Two wires and a ground, daisy chain, cheap silicon from many suppliers, any node can communicate with any other node (without a bus master), proven and tested for decades and ultra reliable—What else can you ask for? An undetected error at a bus load of 25% would only occur once every 220 years if the bus is operated continuously.[1]

The bus length is limited based on the bus speed. For our home, I picked 125 kbps, which allows for at least 500 m (1,500′) of bus length. On this length of wire, up to 112 nodes can be connected to the network. If that is not enough, one can always install more than one network—our house uses two individual CAN bus networks. From a data transmission point of view, the CAN bus transports messages consisting of an 11-bit message identifier and a data payload of 1 to 8 bytes. That may not sound like much, but it is more than enough to transmit typical home automation data like the state of a light switch (1 bit), a temperature or humidity measurement (4-byte float), or the command to open or close a valve (1 bit).

Figure 2 the two separate CAN bus runs installed in our house. One of these serves the upper level, the other the ground floor. While the bus seems to form a loop, it is in actuality a straight run from a termination resistor at each end and a center tap connecting to the servers in the basement. The master nodes for each network provide bus termination, connection to the servers, and power supply management for each net. Power can be provided from two separate independent power supplies, and the master node measures both voltage and current and can disconnect the

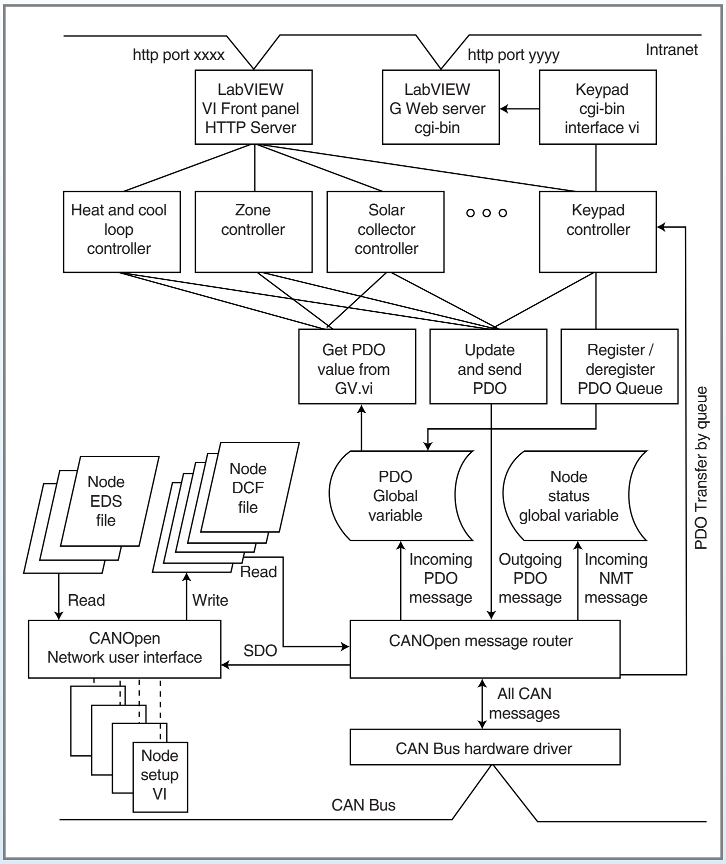


Figure 3—The home automation server's software layout

bus in case of shorts. It will switch from one power supply to the other in case of a power supply failure.

Along the length of each bus, only a few nodes are shown in Figure 2, while in actuality there are about 40 nodes (and counting) right now. Only a small subset of the nodes and functionality were there when we moved in. The system is currently still being expanded in terms of nodes and functionality. This is made possible by the fact that nodes can be hot plugged, and all network wires were in place before the walls were closed off. The bus wires typically run in the ceiling following the walls, and dip into every single rough-in box housing a switch or other home automation component. Physically, the bus wiring is one long stretch of CAT 5e cable, with one pair used for the CAN bus differential signals and another for 24-V DC power to power the individual nodes. Thus, there are two spare pairs available for expansion or other purposes. Photo 2 shows the system during installation.

The two redundant servers are energy-efficient off-the-shelve PCs running Linux and LabVIEW as a programming environment. They each have a CAN bus PCI card with two CAN ports that allow them to connect to both CAN networks. As you might suspect, they are powered through two independent uninterruptible power supplies (UPS) that also power the 24-V bus power supplies. Thus, the system is fully redundant for servers and power, as one power supply and one server are all that is necessary to keep things running. The other can be a hot or cold spare. Beyond redundancy, the double setup makes upgrades of hard and software easier as the system can still operate with the other server while one is tinkering with things.

The servers run the main control software that manages the nodes, allows configuration of the nodes over the CAN bus, and all individual controllers that require user interaction. This can be achieved by operating the controllers from the actual server console. More often, though, controller settings are made through any other PC on the intranet through web browser-based access. Small web pads connect to the servers through WLAN and are used to display and operate the system, again using an HTTP-based connection to the servers. If you are not at home and get a complaint from your wife that the house is too hot or cold, you can also gain secure access to the servers via the Internet. Use a secure socket shell (ssh) connection through the firewall and tunneling. This will enable you to find out and change what is going on at home from anywhere in the world as long as you have Internet connectivity and a laptop or better.

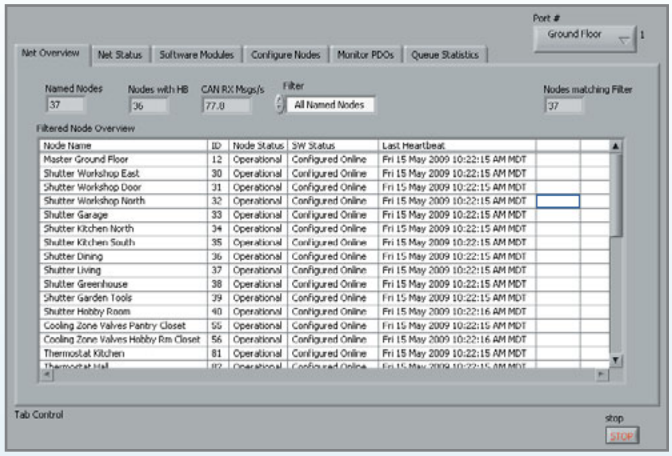


Photo 3—This is a screenshot of the CANOpen network user interface.

the advantage of being nonproprietary and reasonably well documented.

A detailed introduction to CANOpen is beyond the scope of this article, so I suggest you do some reading on CANOpen if you are not familiar with it before reading the rest of this paragraph—or skip on to the next one. The big picture of the control software is shown in Figure 3. Starting at the bottom, there is a CAN bus driver interface between the CAN bus hardware and the CANOpen message router software. This interface is standardized so that different CAN bus adapters can be accommodated without any change to the other software. Currently there are drivers for all National Instruments interfaces under Windows and all Peak Interfaces both under Windows and Linux. The CANOpen message router inspects each incoming message and maintains a global variable of both the data transmitted on the network (PDOs, or "process data objects," in CANOpen terminology) and the status of all the nodes based on the transmitted network management transmissions (NMTs). Service data objects (SDOs) form a

SERVER SOFTWARE DESIGN

Since one of the main purposes of the software running on the server is user interaction, I used National Instruments's LabVIEW as the programming environment. The main advantages for this project are the availability on different operating systems, the ease of user interface design, and the modular interaction of software pieces without having to relink code. A built-in web server comes in handy for remote control. With the CAN bus selected as the bus medium, one can choose to use a number of different software protocols to run on the bus. The most straightforward choice seems to be to develop one's own message transmission standard—but I did not feel up to the task, which is somewhat like reinventing the wheel (since there are nice protocols available) and making it possible to wind up with a flat tire in the process. Instead, I picked CANOpen as the network protocol, which has third category of CANOpen messages. These are exclusively used to adjust and configure the nodes settings, they and are routed to the CANOpen Network User Interface. The latter is only used for maintenance and setup of the network and the nodes. A screenshot of the overview tab is shown in Photo 3. You can see 37 nodes in operation on the ground floor network, transmitting about 80 messages per second. In accordance with the CANOpen standard, for each type of node there is an electronic datasheet (EDS). This is a text file containing a description of all data available on a given node, as well as how it is transmitted over the network. This electronic data EDS is parsed by the user interface when a node is being configured. The user interface will dynamically load a setup VI that is specific for the particular type of node to be configured, and allow the user to adjust all software configurable parameters. Then, a copy of the EDS with the actual parameters of the node as configured is saved as a device configuration file (DCF). When a node shows up on the network by broadcasting its heartbeat message, the CANOpen message router loads the respective DCF for that node and is thus able to correctly interpret the data sent by the node.

The controllers that provide all the home automation functionality interact with the CAN bus through a simple API (see Photo 3). Two options for receiving messages exist. In the asynchronous model, a controller can read the last transmitted value of a PDO as it is stored in the global variable using the "Get Value from GV.VI." In synchronous operation, as shown for the keypad controller, the controller registers to have all incoming PDOs of interest to be transmitted to the controller using a queue, which eliminates the need to poll the global variable for changes. In either case, the controllers transmit messages to the CAN bus through the "Update and Send PDO.VI."

The controllers can be accessed by logging in at the server console, where each controller allows the monitoring and adjustment of parameters through the front panel of the VI. This front panel also can be remotely accessed through a web server that is part of LabVIEW, allowing for the same functionality. However, web access relies on a plug-in that is only available for PCs with a standard operating system (Windows, Linux, or Mac), but not for small web pads or smart phones. Since I picked up a bunch of Nokia 770 web pads for cheap and wall-mounted them throughout the house for home automation monitoring, I added a CGI-based web interface that does not need a plug-in on the browser side. It transmits data as XML in AJAX style. While this requires more programming on the server side, it has the advantage of running on any device with a JavaScript-capable web browser.

NODES & PERFORMANCE

I hope I've whetted your appetite for the second part of this article series. In the next article, I will detail the hardware and firmware associated with the home automation nodes. I will also provide some of the operational and system performance data I've gathered so far. ◾

Author's Note: More details, as well as a software download for this system, is posted at www.siegels.us/SH_Automation.html.

Stefan Siegel (stefan@siegels.us) holds a PhD in Aerospace Engineering with a minor in Electrical Engineering. During the day, he performs fluid dynamics and alternative energy research at the U.S. Air Force Academy in Colorado Springs, Colorado. Stefan developed his home automation system in his spare time.

REFERENCE

[1] O. Pfeiffer, A. Ayre, and C. Keydel, *Embedded Networking with CAN and CANOpen*, RTC Books, 2003.

RESOURCES

K. Etschberger, *Controller Area Network: Basics, Protocols, Chips and Applications*, IXXAT Press, 2001.

U.S. Department of Energy, Energy Plus Thermal Modeling Software, http://apps1.eere.energy.gov/buildings/energyplus/.

SOURCES

Peak PCI bus CAN adapter
Grid Connect, Inc. | www.gridconnect.com

LabVIEW
National Instruments | www.ni.com/labview


Saelig
UNIQUE PRODUCTS & SUPPORT
www.saelig.com
Color LCD Scopes
Best Seller
2-ch + trigger standalone USB bench scope. $287 / $499
PDS5022S / PDS6062T
2-ch 1GSa/s Scopes
NEW! RIGOL
2-ch 1GSa/s (25GSa/s equiv.) 50/100 MHz scope. ~~$595~~ / ~~$795~~
DS1000E
Scope + Analyzer
NEW! RIGOL
25MHz 2-ch /16 logic scope and logic analyzer. $1195
DS1052D
Amazing 7 in 1 Scope! $180
CircuitGear CGR-101™ is a unique new, low-cost PC-based instrument which provides the features of seven devices in one USB-powered compact box: 2-ch 10-bit 20MS/sec 2MHz oscilloscope, 2-ch spectrum-analyzer, 3MHz 8-bit arbitrary-waveform/standard-function generator with 8 digital I/O lines. It also functions as a Network Analyzer, a Noise Generator and a PWM Output source. What's more – its' open-source software runs with Windows, Linux and Mac OS's! Only $180!
"I really like this scope adapter - it's meant for teaching electronic experiments but it's ideal for engineers too."
Alan Lowne
Saelig CEO
CGR-101
www.saelig.com
Handheld Scopes
20MHz / 60MHz rugged handheld USB 2-ch scope. $593 / $699
HDS1022MN / HDS2062M
Pen Scopes
10/25MHz USB powered scope-in-a-probe! Up to 100MS/s. $193 / $280
PS2104/PS2105
XP Emb Touchpanel
Touch-input 10.2" LCD 12V-powered Windows PC. LX800/512MB/4GB/ Ethernet/3xUSB/SD.
CUPC-80
Low-Cost Scopes
2-ch 40/100/200MS/s 8-bit scope range with 5/10/25MHz. $246 +
PS2203/4/5
Scope/Logger
2-ch DSO 16-bit DSO, FFT, VM, logic analyzer, standalone + 24 I/O.
Mephisto
Mixed-Signal Scopes
100MHz Scope, + Spectrum/Logic Analyzer and Signal Generator. $1259+
CS328
microCAM
Compact compressed-serial output camera module for any host system.
uCAM-232 / uCAM-TTL
Multichannel DAQ
12/16 input 1kS/s 10/12-bit PC-connected voltage logger.
PL1012 / PL1216
FREE COFFEE
Call 1-888-772-3544 to get a free Starbucks Card with your >$50 order!
While supplies last - not available with any other offers
Mention offer# SBW
USB Bus Analyzers
Best Value
Packet-Master™ - USB 1.1/2.0 analyzers and generators. $699 +
USB12 / 480+ / 500AG
16-Ch Logic Analyzer
Intuitive full-featured 16-ch 4MB 200MHz sampling memory. $240
LAD-16128U
SPI Bus Analyzer
Protocol exerciser/analyzer for standard SPI and non-standard 4-wire and 3-wire serial protocol interfaces up to 50 Mbps.
SPI Xpress
EMC Spectrum Analyzer
RF & EMF Spectrum Analyzer 1Hz to 7GHz for WiFi, mikes, etc.
EMC RF & EMC Spectrum
EMC Spectrum Analyzer
Handheld Palm PC-based 2.7GHz Spectrum Analyzer.
PSA2701T
Worlds's Fastest
2-ch 12GHz sampling scope for high-speed electrical signals.
PS9200
Waveform Generator
USB2.0 speed 16-bit digital pattern or arbitrary waveform generator.
Wave Xpress
I2C Xpress
Versatile USB 2.0 I2C protocol exerciser and analyzer.
I2C Xpress
Automotive Testing
Kits turn your PC into vehicle-electrics diagnostic tool.
PS3423 / KLARI-MOD
CANminiBOX
Embedded controller series: 2 x CANbus, Ehternet, USB2.0, CF.
emPC-x133
RF Generator
High-res, extremely low-noise, portable 6GHz RF generator.
APSIN6000
RF Testing/EMI Tents
Portable RF test enclosures & shielding tents with external frame.
RF Testing / EMI Tents
Wireless Data Loggers
Log and display temp, hum, volt, event-time or pulse-counting data
RTR-50
Multiparameter Loggers
Mini-logger with built-in temp/hum/ pressure/3-axis accel sensors.
MSR145S
USB Loggers
Standalone USB temp / hum / volt / current loop data logger. $49+
EL-USB-1/2/3/4
Electronic DC Load
Const. current, resistance, conductance, voltage & power modes
LD300
60/100/120MHz AWG
NEW! RIGOL
60/100/120MHz USB 14-bit ARB with USB RS-232, LAN/GPIB.
DG3061A/3101A/3121A
TorqSense
Configurable, patented USB-output non-contact SAW digital rotary torque transducers with integral electronics.
RWT320
USB to I2C
"Drop-in" solution connects PC to I2C/SMBUS + 32 I/O lines. $89
USBI2CIO
FTDI USB ICs
Lowest Prices
Popular UART and FIFO chips. Upgrade Legacy designs to USB.
FT232RL
CAN-USB
Intelligent CAN connection from PC's USB port. $299
CAN-USB
Serial-Ethernet Cable
Network serial product easily without a PC using this 28" cable. $89
eCOV-110-P
Lorlin Switches
Fantastic array of stock and custom switching devices.
Lorlin
Sound Module
NEW!
14-pin module plays back pre-stored audio files from microSD card
SOMO-14D
Keyboard Simulator
USB board adds 55 I/O and 5 x 10-bit A/D inputs, 1 x 10-bit analog O/P.
PoKeys55T
Instant Ethernet
No OS needed. TCP/IP offload, ICs improve system performance.
WIZ110SR / W5100
Ethernet-IO
UDP/IP-controlled 24 digital I/O board 3 x 8-bit TTL ports.
Ether-IO 24
FPGA Systems
Ready-to-go out-of-the-box FPGA/DSP designs for beginners and experts!
RTG005
Wireless Solutions
Analog input, bluetooth wireless modules 433/868/915MHz.
EmbedRF / Adeunis
Temp/RH Sensors
Novel ambient sensors & modules accurately measure temp/RH.
UPSICAP / DLP-TH1
.NET Board
Small (2.2" x 2.2") lowest cost .NET Micro Framework dev system.
USBizi
Easy OLED Display
Compact, economical smart OLED with graphics drive from USB or RS232.
uOLED-96-G1
RF Modules
Simultaneously transmit composite video and stereo audio signals.
AWM6XX TX/RX
RS232 to 422/485
9p-9p or 25p-25p self-pwrd, isolated RS232-RS422/485
KK Systems
1/2/4/8/16 x RS232
Add 1-16 COMports via your PC's USB Port easily.
USB-COM
USB-Serial
A complete CP2102 USB-serial converter in a DB9 shell. $26
CE-USB
Saelig .com
unique electronics
888-75AELIG info@saelig.com
Above are some of our best-selling, unique, time-saving products - see our website for 100s more: WiFi/910MHz antennas, wireless boards, LCD display kits, Ethernet/IO, USB/RS232/485, USB-OTG, instant Ethernet-serial, CAN/LINbus, USB cables/extenders, line testers, logic analyzers, color sensors, motion controllers, eng. software, wireless boards, SMD adapters, I2C adapters, GPS loggers, automotive testing, security dongles, video motion detectors, crystals/oscillators, custom switches, barcode scanners, DSP filters, PLCs, Remote MP3 players, etc. FREE Starbucks card with your $50 order!
Check www.saelig.com often for special offers, bargains, business hints, blog, etc.

by Alexander Popov & Peter Popov (Bulgaria)

Portable Network Service Monitor

This portable network service monitor was developed to help network administrators supervise datacenters. The handy monitor is equipped with a 4 × 20 LCD that can display important messages from any configured server in a local network. It also continuously checks connectivity to predefined services on different machines. When problems occur, it triggers an alarm.

Ensuring service health is the cornerstone of successful datacenter operation. In recent years, investments have increased in disaster recovery and back-up solutions for critical system maintenance. Redundant power supplies, RAID arrays, back-up links, and other systems have lost their high-profile images and have become commonplace. Once the infrastructure is in place to ensure proper redundancy, the obvious need arises to know when to perform a service failover.

Enter service monitors. These come in two main flavors: local and remote. The local variety is installed on the same platform as the monitored service and it tracks system resources—such as CPU, RAM, and disk space usage, as well as whether all required services are running and healthy (e.g., they respond to requests, they don't consume abnormal amounts of resources, etc.). These are convenient when multiple services reside on the same platform, but they suffer from a few major drawbacks: one, a preferable monitoring tool may not exist for the desired platform, it may have to be licensed separately, or it may have to be specifically tweaked for a specific application; two, it has to be installed on the actual platform, meaning that in the case of virtual servers, multiple copies have to be installed, licensed, configured, and maintained; and, three, it is not possible to know whether, or how well, the service is accessible from a remote location.

A remote service monitor solves all of the aforementioned problems. (But it has limitations of its own, so ideally both must be used.) Put simply, it is a piece of software or hardware that emulates client behavior to an extent sufficient to certify whether the monitored service is up and running (and eventually how well it responds). It is usually integrated with the failover infrastructure to ensure minimal interruption. For example, a network load balancer may periodically send simple requests to a web server (e.g., GET /robots.txt). If no response is received,

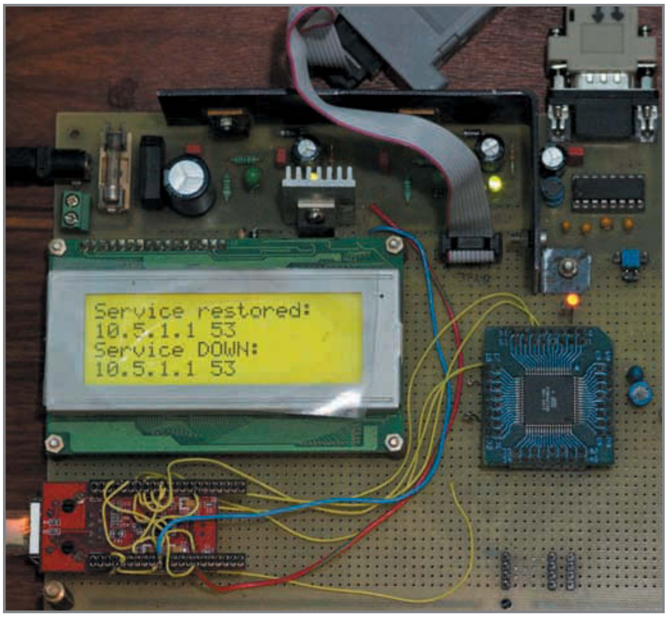


Photo 1—This is our portable network service monitor prototype. The power supply is on top. The RS-232 interface is on the right. In the middle are the LCD and the CPU. The "gem"—that is, the WIZnet module—is at the bottom. It's upsidedown because the 2-mm connector does not match the 2.54-mm board.

redirect incoming client requests to a back-up server.

Virtually all load balancers have the capability to perform the aforementioned "ping" (as it is widely yet incorrectly called). However, to the best of our knowledge, there weren't any accessible embedded network service monitors available when we began building our own (see Photo 1). It may look like just another gadget at first glance, but it's surprisingly handy. If you are like us, you often wish there was an easier way to do this than carrying a 17" laptop, keyboard, and monitor to the datacenter. And there's never a keyboard-video-mouse (KVM) when you need it.

Not only is a portable network service monitor an obviously affordable solution, but it is also extremely flexible. It may be cheaper to get free software to do the job; however, an appropriate platform may not be available in certain segments of the network, or it may not be available at all as enterprise environments are often vendor-locked. Moreover, installing software randomly is usually out of the question. Using the existing network infrastructure—such as load-balancers or smart switches—may require complex routing and firewall rules, which may not even be possible in the DMZ or behind an air gap. All of these require change requests and are subject to change freezes, so it is not always possible to get the work done immediately.

With a portable network service monitor, you can just plug it in the nearest switch port in the desired network segment, and after just a few minutes of configuration, you can determine connectivity and service availability. It also fits perfectly when setting up a rack cabinet and not all the required equipment is available. During a sales demo, simply plugging in a preconfigured device and making sure everything is up and running can save a lot of embarrassment.

ETHERNET CONTROL

We used a WIZnet W5100 hardwired TCP/IP Ethernet controller in this project (see Figure 1). Like similar products on the market, it has both an Ethernet MAC and a physical Ethernet interface. The CPU connection is through either the memory-mapped direct access interface (many wires) or the much simpler SPI mode (fewer wires). Nothing new here. You can connect an 8-bit processor with an SPI and have Ethernet connectivity. But we needed more: IP, TCP, UDP, SNMP, and DHCP. Usually, these protocols are implemented in software.

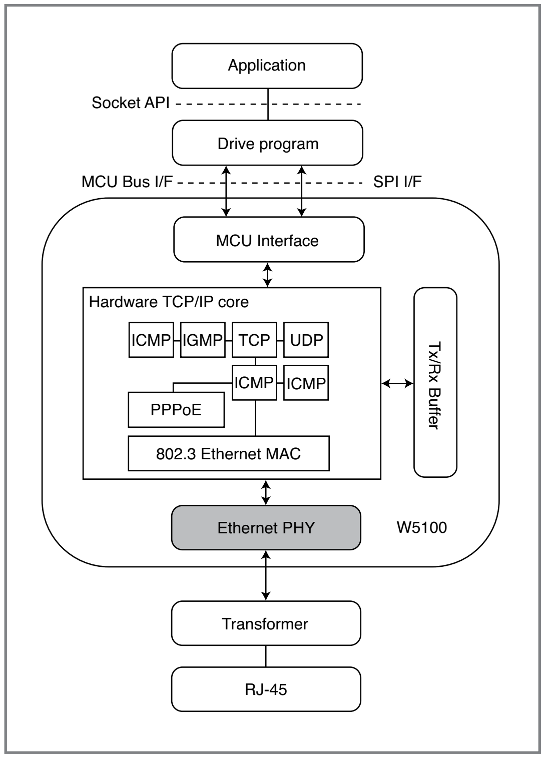


Figure 1—The useful W5100 chip has an Ethernet MAC and a physical Ethernet interface.

The W5100 provides a hardware solution for the most commonly used network protocols: IP, TCP, UDP, ARP, IGMP, ICMP, and PPPoE. In addition to this, the chip has 16-KB memory dedicated to transmit and receive buffers. The hardware supports four different channels for socket operations. Each channel can work in either Client or Server mode. WIZnet provides a software library, which abstracts the hardware registers and simplifies interfacing with the user application.

DESIGN DEVELOPMENT

The design features a WIZnet WIZ810MJ module (with a W5100 chip) that's controlled by an Atmel ATmega128 microcontroller. It also includes a power supply, a 4 × 20 LCD, and a TTL-to-RS-232 converter for the debug console. Usually, providing TCP/IP support over Ethernet requires a decent amount of resources that certainly pushes the limits of a small embedded system. There are many commercial and free network stack solutions implemented entirely in software, but they are inherently complex and usually require an underlying operating system and a fair chunk of code and data memory. This poses the need to use external memories, further increasing device complexity and cost. However, due to the very integrated WIZnet module, which gives us access to a built-in network stack right out of the box, a simple 8-bit microcontroller is sufficient for this design. The ATmega128 has enough internal flash memory (128 KB) and RAM (4 KB) for this project, so external memory is not needed. And since the TCP/IP state machines are implemented in hardware, there is no need for complicated software and an operating system.

For the device's IP initialization, we used a DHCP client. DHCP servers are present in most networks I've seen. If a specific IP in a defined range is needed, it always can be leased based on a MAC address or simply requested by the client. Implementing a static IP address is equally trivial if required.

To manage device configuration, we implemented a simple TCP Telnet-based console with a few simple commands, such as viewing or editing the list of services to monitor. The list of network services (IP:PORT) can be configured with a text editor of choice and piped through a Telnet client on virtually any conceivable platform. A more interactive interface could be used (e.g., with a menu), but accounting for various virtual terminal emulation models is beyond the scope of this project. We purposefully didn't use the sample web server. We could use it to edit the list of servers, either interactively or by simply uploading it via an HTTP PUT or POST request, but a web browser won't be readily available in many of the locations where the device will be used. Many datacenters don't even have KVMs as most server vendors have some way to provide remote KVM emulation. A Telnet client, on the other hand, is available on any platform imaginable, from home routers to mainframes, so this is how we chose to interface with the device.

Once configured, the device continuously runs a simple health check iterating over all servers in the configured list. The actual health check performed is a `connect()` call, which is a complete three-way TCP handshake. We have seen many issues with the typical "SYN-SYN/ACK-RST" (i.e., TCP half-open) health check that most load balancers perform. These issues ranged from

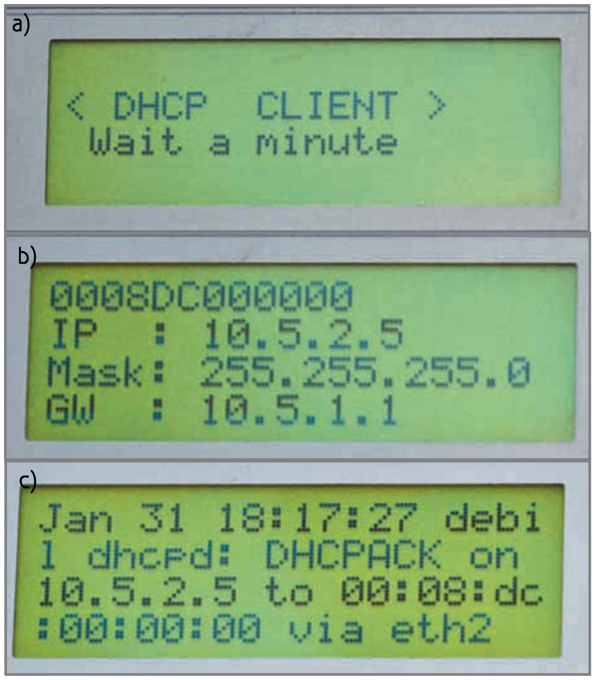


Photo 2a—During start-up, it tries to get its network settings through a DHCP server in the network. **b**—Here you see the successful initialization of device network parameters via DHCP. The first row shows the device's MAC address. **c**—This is the DHCP server log message of the response to the device's request for network parameters. (Printed via lcdprintd service.)

raising false alarms, which filled server logs with pointless exceptions, to having the load balancer IP throttled or even blocked because the behavior was considered malicious and akin to a SYN flood attack. We avoided these problems by using a proper three-way handshake.

The W5100 has four available socket channels. So far, we used three of them. The last one listens for feedback from selected servers. It is basically a TCP console that echoes on the 4 × 20 LCD (with vertical scrolling) anything that it receives on a specific TCP port. SNMP would have been better, of course, but this approach is good enough to demonstrate the device's capabilities. SNMP could be implemented with a library of choice (as required), provided it is stripped down to size. The way we implemented it, sending a message from a server to the monitor is trivial when using readily available tools, such as Netcat or a Telnet client.

We built the device on our universal development board. We designed it several years ago when we realized every embedded project needs at least one power supply (we included two) and an RS-232 connection (hence the two MAX3232 circuits). Some small but important components include a power connector, a fuse, a diode bridge, and a Reset button. We've used the board successfully in many projects, including a few award-winning *Circuit Cellar* design contest entries. We had a board with a 4 × 20 LCD and an Atmel ATmega128 left over from one such project, and it proved to be the perfect design bed for our needs.

FUNCTIONALITY

When the device first starts, it uses DHCP to get an IP address (see Photo 2a). If the DHCP server sends us a valid lease, the device uses the assigned IP; otherwise, a default is substituted, which is useful for networks without DHCP. Photo 2b

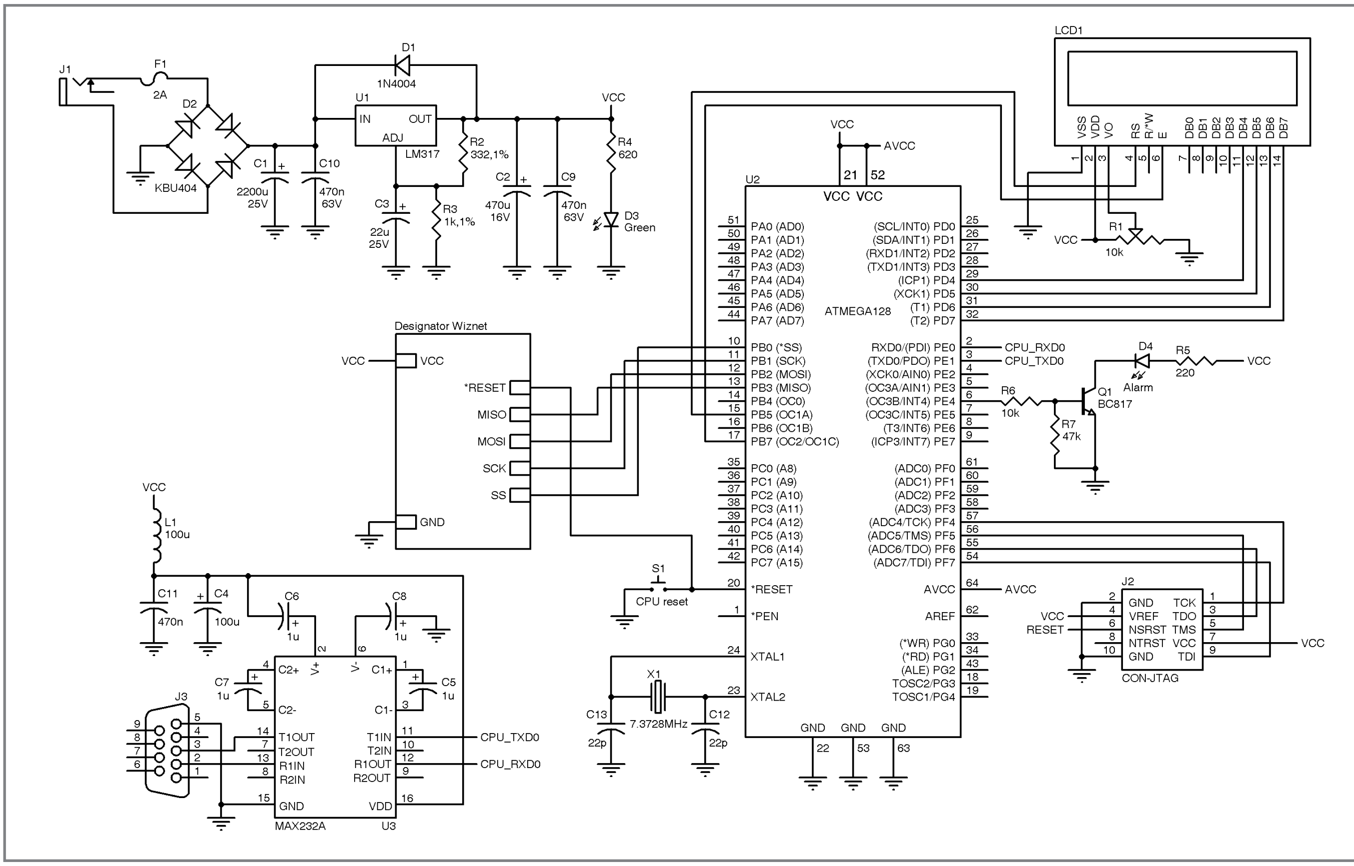


Figure 2—It's easy to perform TCP/IP networking with an 8-bit microcontroller by using the WIZnet module in SPI mode.

shows a sample configuration including a valid DHCP response.

Once an IP has been acquired, the lcdprintd and telnetd servers are started and wait for connections. Two service entries are hardcoded, but the aforementioned Telnet interface can be used to override them. After initialization, the probing loop is initiated. If a status change is detected in any service, an appropriate message is displayed on the LCD.

If the software is compiled with debug flags for some of the components, the information is sent over the serial console. The device then can be connected to any available box with a serial port, and its debug output redirected to file, syslog, or sendmail. This is not required for normal operation, but it is available for debugging purposes.

HARDWARE

Figure 2 shows the entire design. The most interesting and exotic part is the WIZ810MJ Ethernet module, which consists of the highly integrated W5100

Ethernet PHY+hardware TCP/IP stack IC, a transformer, an RJ-45 connector, and a multiplexer for the SPI bus. The first revisions of the WIZ810MJ module were with 2-mm pitch, which was not always convenient for prototyping because it required the module to be mounted "belly up" and soldered to the board via the RJ-45 connector. But WIZnet quickly responded to user requests and changed the connector to the standard 2.54-mm (100 mil) type.

The ATmega128L microcontroller's internal data and code memory is enough for this project, so external memory modules are unnecessary. (We can add them for more complex solutions if needed.) We chose this MCU for three main reasons. One, we'd worked with it on several projects. Two, the WIZnet software library for the W5100 is specifically made for this MCU. And three, there are freeware tools for software development: AVR Studio and WinAVR. In-system debugging is possible with an inexpensive JTAG programmer. That's the purpose of the J2 connector. There is an affordable programming-only solution—via the ISP interface. There are a lot of easy-to-build programmers on the Internet. Most consist of just a buffer chip between a serial or parallel PC port and the MCU. There is one inconvenience associated with using an ISP interface for this particular project. It uses the SPI pins, so extra care must be taken to disable the WIZnet module during programming.

The processor is connected to the module via a SPI connection. This reduces the number of wires required and also simplifies the hardware. The other alternative is parallel memory-mapped mode with a lot of data, address, and control lines. It may be faster, but we didn't need it. Also, we wanted to prove we could use even more affordable, easier-to-mount microcontrollers that don't provide so many pins for memory-mapped mode. Of course, if a smaller microcontroller is going to be used, it should at least provide enough code and data memory for the application. As it turned out, the SPI worked out just fine.

The device is powered through a standard DC (10 to 24 V) or AC (7 to

Listing 1—The code for testing if a service is active

```
switch (sm_st) {
   case ST_IDLE: return;
   case ST_TESTING:   break;
   case ST_TO:
   case ST_OK:
      if(sm_st==ST_OK) {
         led_off(0);
         services[iservice].up=1;
      } else {
         led_on(0);
         services[iservice].up=0;
      }
      if(services[iservice].up != services[iservice].upold) {
            if(services[iservice].up)
                fprintf_P(flcd, PSTR("\nService restored:"));
         else
                fprintf_P(flcd, PSTR("\nService DOWN:"));
         fprintf_P(flcd, PSTR("\n%s %d"),
         inet_ntoa(ntohl(services[iservice].ip)),
         services[iservice].port);
         services[iservice].upold = services[iservice].up;
      }
      services_print(stdin);
      sm_st=ST_IDLE;
      return;
   }
switch (getSn_SR(ch)) {
   case SOCK_ESTABLISHED:
      if(bchannel_start[ch]==1)       bchannel_start[ch] = 2;
      if ((len = getSn_RX_RSR(ch)) > 0) { // check Rx data
         if (len > TX_RX_MAX_BUF_SIZE) len = TX_RX_MAX_BUF_SIZE;
         len = recv(ch, data_buf, len);// throw the received data
      }
      sm_st=ST_OK;
      disconnect(ch);
      bchannel_start[ch] = 0;
      break;
   case SOCK_CLOSE_WAIT:     /* If the client requests to close */
      disconnect(ch);
      bchannel_start[ch] = 0;
      break;
   case SOCK_CLOSED:          /* if a socket is closed */
      if(!bchannel_start[ch]) { /* Service monitor client started */
         PRINTLN1("%d : Service Monitor TCP Client Started.",ch);
         bchannel_start[ch] = 1;
      }
      if(socket(ch,Sn_MR_TCP,DEFAULT_SOURCE_PORT,0x00) == 0) {
   /* reinitialize the socket */
         bchannel_start[ch] = 0;
      } else {
         connect(ch,(u_char *)&services[iservice].ip,
services[iservice].port); // try to connect to the tested service
      }
      break;
}
```

Listing 2—The code for finding items in a command line

```
uint8_t nextitem(char **ppc) { // Returns non-zero if there is an
item available
        if((*ppc!=NULL) && (**ppc!=0))
                (*ppc)++;
        while((*ppc!=NULL) && (*ppc!=0) && isblank(**ppc))
                (*ppc)++;
        return ((*ppc!=NULL) && (*ppc!=0));
}
```

Listing 3—This is a command-line session that shows: adding a service, clearing all the services, a listing of all the services monitored, and changing of service state.

```
Monitored services:
10.5.1.2 21 0
10.5.1.1 53 0
A 192.168.0.1 80

Adding service: 192.168.0.1 80
Success!
L

Monitored services:
10.5.1.2 21 0
10.5.1.1 53 0
192.168.0.1 80 0
--------------
C

Services list cleared
L

Monitored services:
--------------
A 10.5.1.2 21

Adding service: 10.5.1.2 21
Success!
L

Monitored services:
10.5.1.2 21 1
--------------
```

17 V) power adapter. A fuse protects the adapter from a short circuit on the board—something that can happen especially during development. Naturally, the fuse-rated current should be lower than the adapter's peak current lest the MCU turns into the proverbial TV that burns out first to save a $0.10 fuse!

The diode bridge D2 enables us to use AC or DC adapters with any polarity. The supply voltage V_{CC} is generated by an LM317 linear voltage regulator. The voltage divider formed by the precise resistors R2 and R3 sets the output voltage to 3.3 V. The diode D1 protects the power transistor inside U1 from reverse voltage after powering off the device.

When the power supply is cut off, the input capacitance (C1) may discharge faster than the output one (C2), especially in a case when there are several voltage regulators connected to it. This is a common problem that often goes unnoticed. In this situation, the voltage in the output is larger than the voltage in the input and the power transistor inside the chip is reversed, which often causes it to fail. The diode connected in the reverse direction minimizes this voltage to its forward junction value, thus protecting the transistor.

Listing 4—This code supports vertical scrolling. Due to the write-only hardware connection to the LCD, a buffer in RAM is used to cache screen content.

```
unsigned char xi,yi;

for(yi=0; yi<LCD_LINES-1; yi++) {
        lcd_gotoxy(0,yi);
        for(xi=0; xi<LCD_CHARS_PER_LINE; xi++) {
                lcd_write(lcdscreen[yi+1][xi], DATA_MODE);
                lcdscreen[yi][xi]=lcdscreen[yi+1][xi];
        }
}
```

A MAX3232 is used to convert between CMOS and RS-232 levels. Since V_{CC} is 3.3 V, the well-known MAX232 powered with 5 V is not suitable and its lower voltage is used. The MAX3232 is powered through a simple LC filter to suppress the noise it generates through switching capacitor circuits. RS-232 signals are led out to a standard DE9 male connector, so a cross-cable should be used when connecting to a PC. The hardware permits the use of RTS/CTS signals for flow control. This is not implemented in this version of the software, since it isn't required for a simple serial console.

A standard 4 × 20 LCD connected through a six-wire interface (four Data, Enable, and Register Select) is used for alphanumeric output. The LCD is powered by a second linear regulator built with an LM317 (not shown on the schematic). The only difference is the resistor divider correspondent to R2-R3, which are 332 Ω and 1 kΩ, respectively, to provide 5 V. All the signals between the MCU and LCD are in a direction from a 3.3-V device (MCU) to a 5-V device (LCD), so no level shifters are needed.

SOFTWARE

The software is written in C. The code is built around the WIZnet driver, using the Arduino implementation. Internet protocols require specific byte ordering. Dedicated functions are needed to convert between host and network Endianness. The following new components were developed.

The main service testing function is `servmon_tcpc()`. It uses client mode TCP to check the availability of a service. If current state of the service (up/down) doesn't match the old state, a message with the server IP/port and state is written on the LCD. The Alarm LED illuminates when service is down. (You can use a beeping sound in place of the diode if necessary.)

The `servmon_timer()` function is called periodically every second. First, it checks whether TIME_BETWEEN_TESTS seconds have elapsed since the last connection attempt, and it sets a timeout state in that case. Next, it iterates to the the next service to test if its TIME_WAITING_FOR_CONNECTION seconds interval has elapsed.

There are two important parameters to be configured (in servmon.h):

```
#define TIME_WAITING_FOR_CONNECTION
  5 /* in seconds */
#define TIME_BETWEEN_TESTS 10 /* in
  seconds */
```

The main code for testing a service is provided in Listing 1.

The `telnetd` server is used to configure the device remotely. It uses TCP port 5006, which can be changed in the config.h file (`DEFAULT_CONSOLE_PORT`). Currently, the following commands are implemented. All commands are case-insensitive.

With the `A <IP> <PORT>` command, new services can be added to the list. Only numeric IPv4 addresses are supported because a DNS server may not be accessible in some environments. IPv6 is not supported by the WIZnet part's hardware TCP stack. Some services can be predefined in the code if needed.

Blank characters are permitted between items in a command. The code in Listing 2 parses the next item, with `ppc` pointing to the last item.

In Listing 3, the `L` command prints out the current list of services and their last known states (0-service down, 1-service operating). Use the `C` command to clear the entire list of services to monitor.

The `lcdprintd` server simply prints whatever it receives from the TCP socket. This is useful for the network administrator to have a peek at important messages from all servers in the network (see Photo 2c). The following sample command in Linux will send the hard drive temperature of all internal hard drives to our device (assuming 10.5.1.100 is the IP assigned to it by DHCP):

```
watch -n 60 'hddtemp /dev/sd? | telnet
  10.5.1.100 5005'
```

The port of this server is defined in config.h:

```
define DEFAULT_LCDPRINTD_PORT 5005
```

The `lcd_io` module controls the LCD module using a six-wire interface. It is mainly built on top of Chris Efstathiou's LCD library, with some improvements like LCD vertical scrolling. For the vertical scrolling to be implemented, the rows below the first one should be copied one row up. This can be easily done by reading the current symbol from the LCD module memory and writing it back above. However, this implementation requires one more wire to be connected between the MCU and the LCD. For small character screens, such as the one we used (4 × 20), a buffer in the MCU memory can hold the symbols currently on the LCD. This way, a write-only connection to the LCD module is sufficient. The code for vertical scrolling is lcd_io.c (see Listing 4).

FUTURE DEVELOPMENT

In the future, we'd like to be able to store configuration information in the internal microcontroller's EEPROM, which would be convenient if we have to disconnect the device from the power supply on a frequent basis. By adding additional hardware and a specific compatible magnetics, the device could be powered by power-over-Ethernet, which is usually found in office networks with VoIP phones.

We could also expand the design with SNMP support for better integration with any existing monitoring infrastructure. Another advantage would be to integrate an external GPRS module for monitoring network segments 24/7—even behind the air gap!

Alexander Popov (sasho@popovbrothers.com) has been designing electronics for more than 20 years. He has a Master's degree in Telecommunications and has been working in the telematics field for companies like IBM. Alexander designs the software and hardware aspects of Linux and uCLinux-based embedded

devices. He is currently searching for new opportunities.

Peter Popov (peter@popovbrothers.com) is an experienced IT professional with a background in networking, operating systems, and application support for critical IT infrastructures. He holds Master's degrees in Industrial Engineering as well as in Communication and Information Systems. Peter is currently the head of the Professional Services and Integration Development Department at Software Technologies, Ltd.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/238.

RESOURCES

P. Lindsay, "Learning About Arduino and WIZ810MJ," http://code.rancidbacon.com/LearningAboutArduinoWIZ810MJ.

WinAVR, http://winavr.sourceforge.net/.

SOURCES

ATmega128 Microcontroller
Atmel Corp. | www.atmel.com

WIZ810MJ Module
WIZnet | www.wiznet.co.kr/en/

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*.

For more need-to-know information about topics covered in Alexander and Peter Popov's Issue 238 article, the *Circuit Cellar* editorial staff highly recommends the following content:

—

iEthernet Bootcamp
Get Started with the W5100
by Fred Eady
***Circuit Cellar* 208, 2007**
Ready to work with a hardwired TCP/IP embedded Ethernet controller? Fred helps you get started on your first W5100-based design. Topics: W5100, Ethernet, TCP/IP, PPPoE, C compiler, MPLAB

Go to: www.circuitcellar.com/magazine/208.html

—

Automated Data Mining
Build an Embedded Server Application
by Matt Pennell and Aaron Thomas
***Circuit Cellar* 219, 2008**
This embedded server app finds airfare deals on the 'Net. It uses an online travel search engine to find data. Topics: Embedded Server, W5100, TCP/IP, XML

Go to: www.circuitcellar.com/magazine/219.html

by Chris Paiano (USA)

OAE Probe Amp and Intercom (Part 1)

System Planning and Design

This series covers the design of an otoacoustic emission (OAE) probe amp and touch-to-talk intercom system. The design is used for professional otoacoustic studies in a laboratory setting. Here you learn about the study of otoacoustics and how to perform hearing experiments with a standard PC.

A research lab recently hired me to build a touch-to-talk intercom system with an integrated high-gain otoacoustic probe amplifier. This project introduced me to the study of otoacoustic emissions (OAEs)—sounds originating from the inner ear. These emissions are directly related to ear health; a deaf ear produces no OAEs. Tests that purposely stimulate and measure OAEs are suitable to test the hearing of infants and small children who are otherwise too young to participate in a conventional hearing test.

My OAE probe amp/intercom system is used for performing highly accurate OAE-related experiments with low-noise, high-gain components (see Photo 1 and Figure 1). These experiments—which sometimes involve maintaining real-time communication between an isolated subject and the control room—are facilitated by the touch-to-talk intercom.

The otoacoustic probe amplifier is completely separate from the intercom. It's built into the subject's intercom box for convenience. The intercom portion of this project features a Cypress Semiconductor CY8C29466 PSoC microcontroller (see Figure 2). The control room and subject units are connected by an RJ-45/Ethernet cable. The use of this high-quality standard gives an experimenter freedom to purchase various cable lengths locally. Only the control room unit needs a power source.

An optional external 12-V lead-acid gel cell battery may

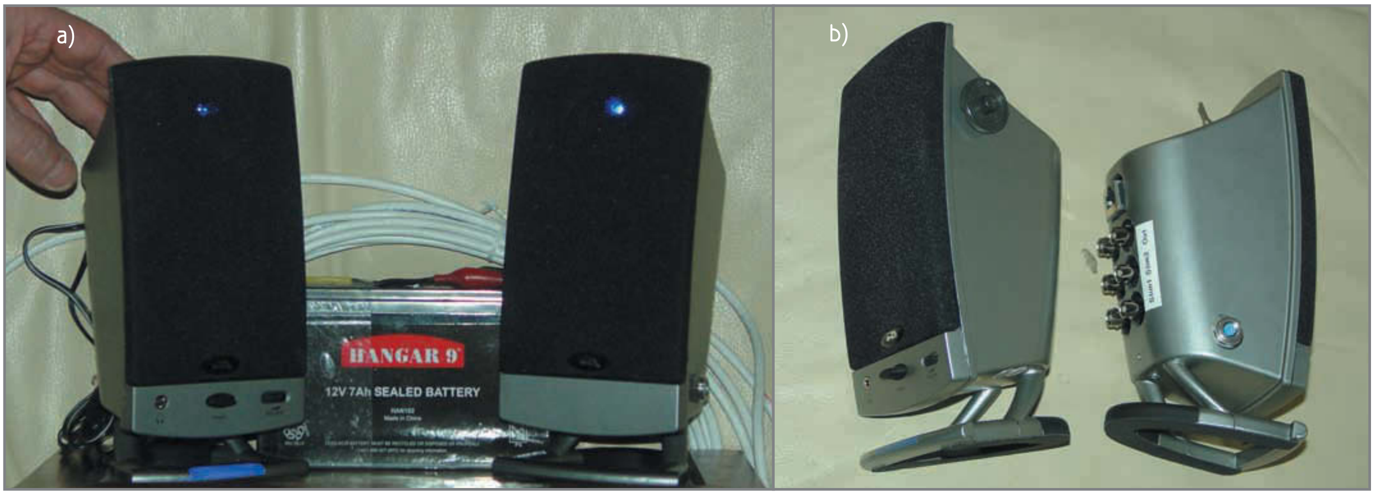


Photo 1a—This is the OAE probe amp/intercom with an optional battery detecting a touch. b—The OAE probe amp/intercom system features a control box, subject box, OAE probe I/O, and more.

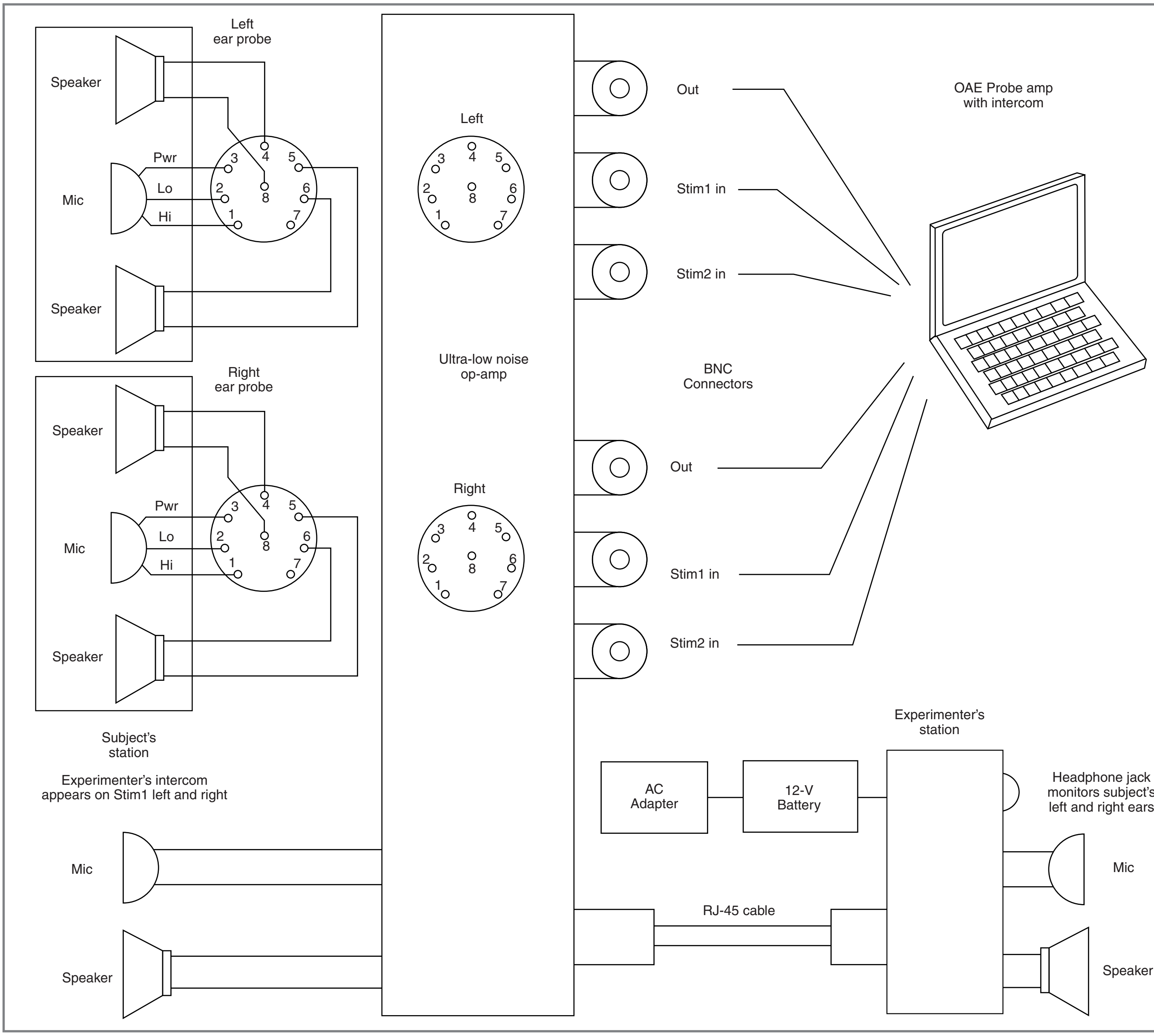


Figure 1—These are the main parts of the OAE probe amp/intercom system.

be used to power the system through the included battery/power cable. This is recommended for actual otoacoustic experiments. It eliminates any AC supply noise and is inherently safer—a good feature for any device using electricity around or on the human body. When the battery is connected at the same time as the included AC adapter, the battery will charge (whether the system is powered or not). If the AC adapter is removed, the battery will power the system. Without the battery, the AC adapter will power the system.

The professional experimenter can make use of any existing otoacoustic stimulation and recording devices with this probe amp/intercom system through standard BNC and TRIAD8 connectors. For the amateur, without access to this specialized equipment, any computer with a sound card can be used instead. Although it took a lot of time to find, configure, and learn to use, there is a freeware solution out there for amateur otoacoustic experimenters, which I will explain in detail later in this article.

INTERCOM

The intercom system and probe amplifier are both built into a set of computer speakers that I bought for less than $20 at a local computer store. There was enough room inside to fit a PCB in each speaker, and there was plenty of surface area to drill holes for connectors. This also gave me an intercom volume control, power button, aux headphone jack, power input jack, and some nice bright blue LEDs—all of which were utilized in the design.

The push-to-talk (touch-to-talk) hand sensor is on the top of the control box—on the inside, actually. It senses upwards through the case. Just place your hand over the top and get relatively close to it, or touch the top of the case.

The MP3 files included with this project demonstrate the simple audio feedback integrated into the intercom. The four sounds are (listed in the order

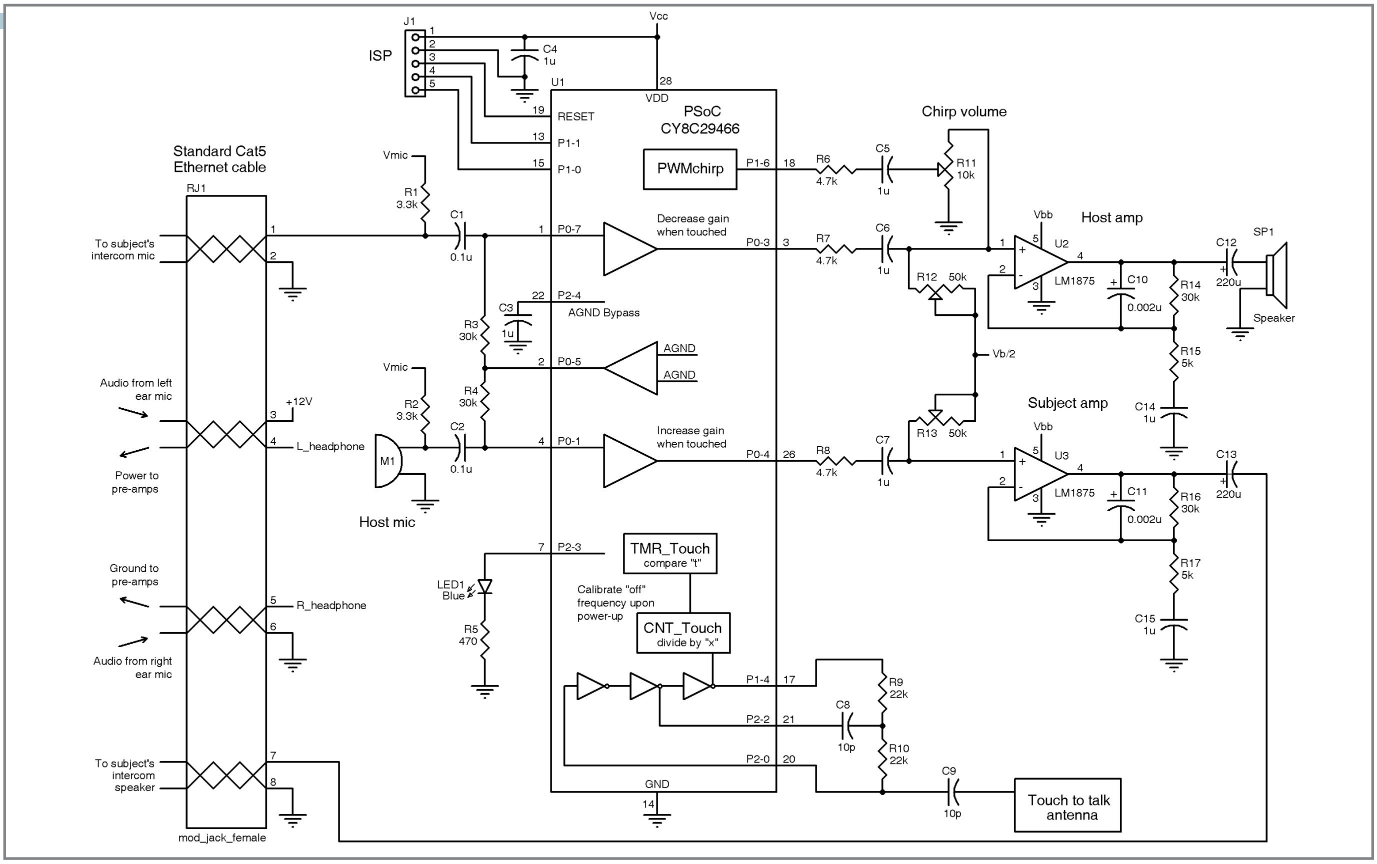


Figure 2—The OAE intercom includes Ethernet cable, a host microphone, a touch-to-talk antenna, a PSoC, and more.

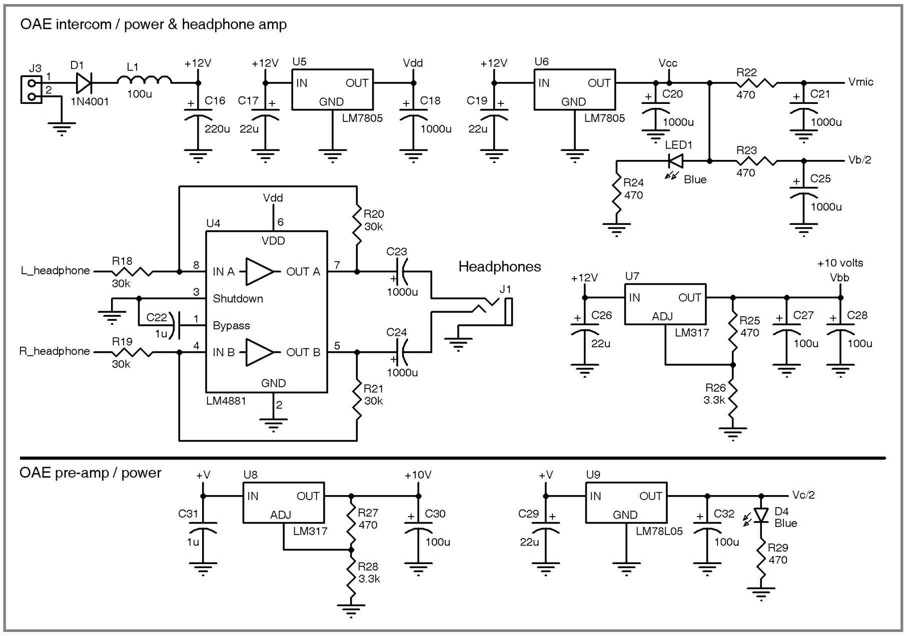


Figure 3—Here you see the intercom/power and headphone amp, as well as the OAE preamp power.

they will be heard during operation) StartupSlide.mp3, CalibrationComplete.mp3, ControlActive-TouchDetected.mp3, and SubjectActive-TouchReleased.mp3.

CalibrationComplete.mp3 is the sound played at start-up, indicating that the unit was just powered up and the touch sensor has begun calibrating itself. Touching the top of the control room unit before calibration is complete can skew the calibration—or at least increase the required time for calibration. The subject can speak to the control room freely during this time.

StartupSlide.mp3 is the sound played when the microcontroller is satisfied with its touch sensor calibration. The touch sensor is now active, and the control room can now push-to-talk (touch-to-talk) to enable communication to the subject.

ControlActive-TouchDetected.mp3 is the sound played when a hand is sensed close enough to the top of the control box, indicating that the control room can now speak to the subject (Touch Detected event). The blue LED above the control box speaker will light as long as a hand is detected.

SubjectActive-TouchReleased.mp3 is the sound played when a sensed hand moves far enough away from the control box after having triggered the Touch Detected event, indicating that the subject can now speak to the control room again (Touch Released event). The blue LED above the control box's speaker

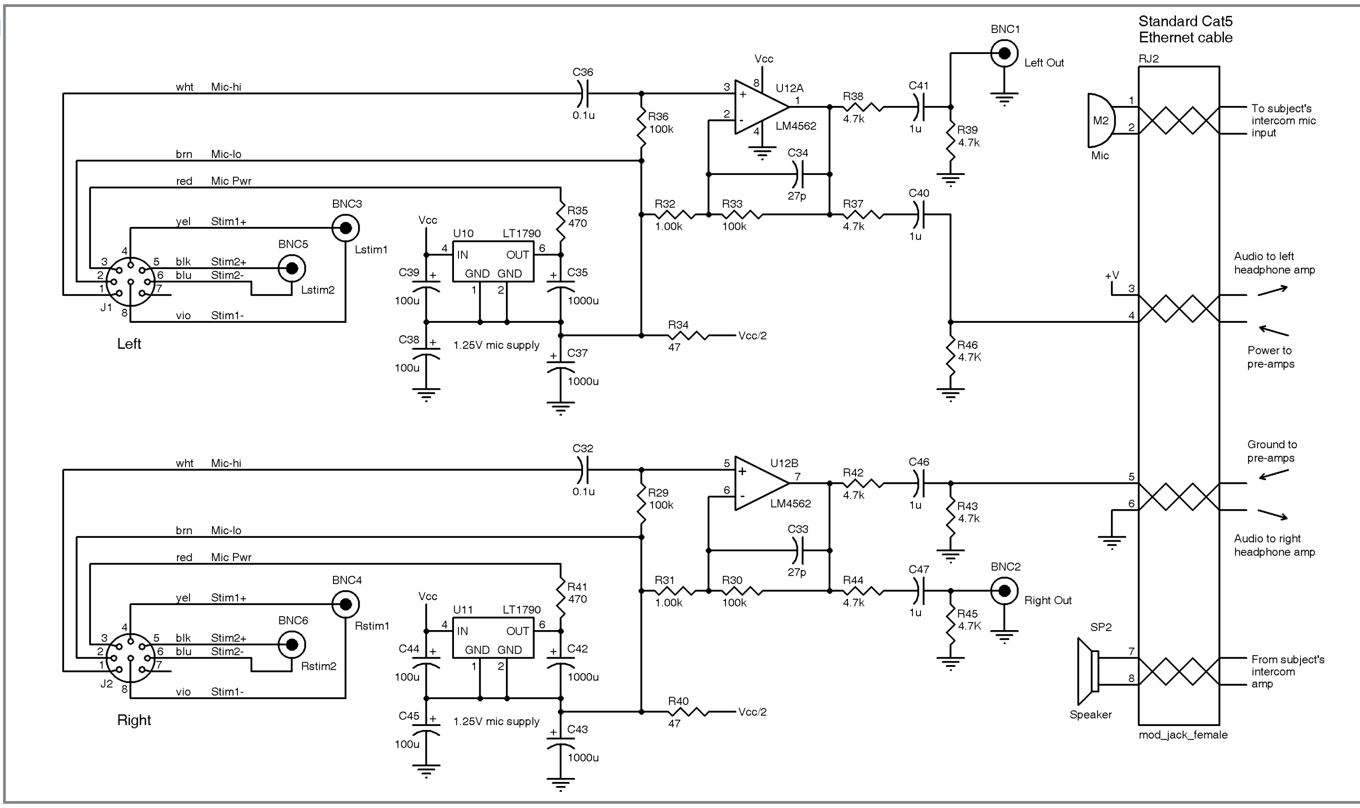


Figure 4—The otoacoustic probe pre-amp

will turn off.

These sound effects are mixed into the control room's speaker. Adjust the knob on the right side of the control box to set the relative volume (see Photo 1b).

Photo 2—Socketed gain resistors are behind the subject box's speaker.

TOUCH/PROXIMITY SENSING

I began evaluating touch-sensing ICs, naturally, with the Cypress PSoC CapSense family of ICs (for the intercom's touch-to-talk function). A CapSense can definitely do the job—several touch-sensitive buttons and even a touch slider can exist simultaneously. However, after some experiments, I had two issues. First, the CapSense seemed like overkill for this project. I needed only a single touch sensor, and I did not need to define the sensor area so specifically. In fact, for this particular application, it would be best to have a wide and unrestricted sensor area. Second, the effective distance at which proximity could be sensed left me unable to utilize the computer speakers' plastic case to house the sensor and still reliably sense capacitance.

I decided to build my own capacitive touch/proximity sensor using a standard CY8C27443 PSoC, which could also handle the intercom function. For convenience, I actually used a CY8C29466 PSoC for the extra eight digital blocks. This saved some development time, as I did not have to juggle digital module placement and routing to fit all the user modules into the top eight blocks. This intercom design can be reconfigured to fit in the less-expensive CY8C27443 PSoC.

Designing a capacitive touch/proximity sensor with standard PSoC user modules was not too difficult. Three digital inverter modules, linked in series with an external output to input loop, create an oscillator based on module transport delay. The frequency of this oscillation is partially dependent on the capacitance of the external loop connection—where I have attached an antenna.

Two timer modules are configured to measure the frequency of the oscillation. Then, firmware logic determines whether or not there is a hand near the antenna and activates the appropriate intercom communication direction. The intercom is shown in Figure 2, Figure 3, and Figure 4.

OTOACOUSTIC PROBE AMP

My client specified that this design work with a microphone that was

already integrated into his otoacoustic probes. The main requirement was to minimize the noise level. A quick search turned up the National Semiconductor LM4562, which was among some of the lowest noise op-amps available (rated at 2.7 nV/√Hz)—and I just happened to have a few in stock. I set up a breadboard with a dynamic mic as the input source and a high-impedance set of closed-ear headphones as the output load. I was truly amazed at how much gain could be applied with no discernible hiss.

When I tried to use the microphone as part of the ear probes, there was that familiar noise that I had expected in the previous test. Although I was initially dumbfounded, I deduced that there could be only two suspects: the mic itself or the phantom supply.

I reexamined the included mic's datasheet and noticed that it has a maximum input voltage of 1.6 V. I had used an LM317 adjustable regulator; but to my great surprise, it turned out to be the source of the noise. I installed a Linear Technology LT1790-1.25 precision reference, and the problem was solved. Figure 4 shows the probe amplifier.

You should be able to find a sensitive mic with a tube port, similar to the one I used, for around $30. That's an inexpensive solution for virtually anyone experimenting with otoacoustics. Speaking of which, experimenters using a computer's sound card and a homemade probe should note the following: The mic input should first route through the probe amplifier to boost sensitivity to a level appropriate to detect OAEs. Just listening for a response on a raw microphone signal will not work.

RJ-45 ETHERNET CABLE

I selected an RJ-45 Ethernet cable for a few important reasons: length (I could not get a good answer on how far apart the units would be in the lab; replacement (in case of trouble, a new cable could easily be obtained locally); cost (it's hard to compete with mass-produced components); and stock (the jacks and cables were already available in my parts bins).

The cables contain four twisted pairs—just enough to satisfy all the design's requirements. An independent pair each for the mic and the speaker allowed the intercom to be completely isolated from the probe amp.

The other two pairs were used as a separate audio feed from the left and right probes for a headphone amp in the control unit (for monitoring the probe signals). The common of those two signals is used for power and ground, so an artificial split supply is derived for the probe amp on the subject's side.

ADJUSTING AMP GAIN

Some resistors—the second (R_2) and fourth (R_4) resistors from the top—are socketed behind the subject box's speaker. These provide control over the OAE probe amp gain (see Photo 2). This makes adjustments quick and simple. It also enables me to experimentally determine the noise floor for each particular application or laboratory and apply the maximum possible gain to the test signal.

As shown in Photo 2, the effective gain applied to the OAE probe is

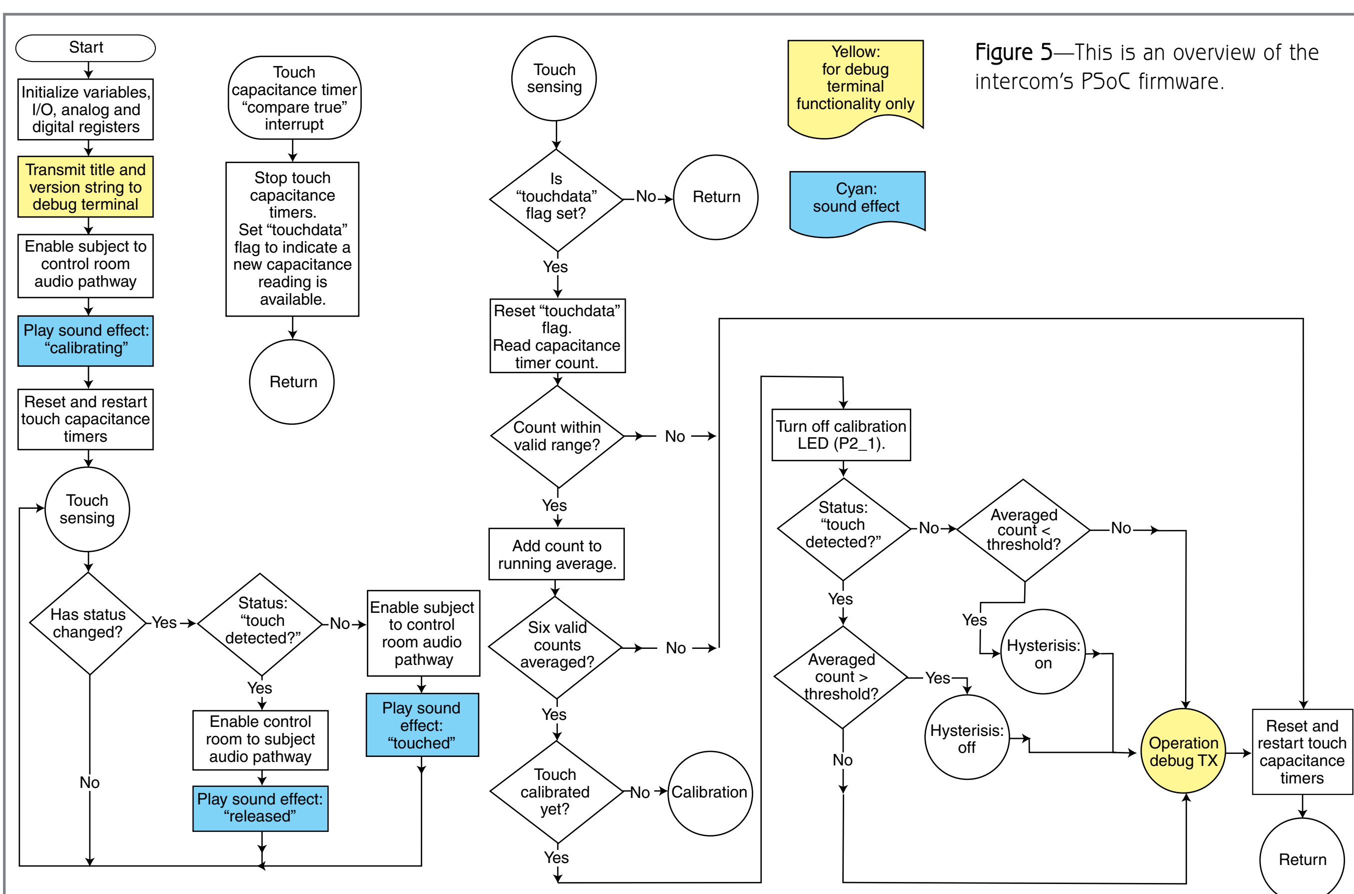


Figure 5—This is an overview of the intercom's PSoC firmware.

BRAIN FOOD
AT YOUR
FINGERTIPS
CIRCUIT CELLAR
THE MAGAZINE FOR COMPUTER APPLICATIONS
INTERNET & CONNECTIVITY
Internet-Based Weather Data Acquisition
A Compact Webcam Design From Start to Finish
Ethernet-Controlled HERMS
Text Library for Real-Time Translation
I2C Master Bus Controller
CIRCUIT CELLAR
DATA ACQUISITION
Very Low-Frequency Datalogger Design
A Simple Circuit for Energy Conservation
Sophisticated Embedded App Development
Battery-Monitoring Circuitry
Design & Then Program a Stand-Alone Timer
CELLAR
ANALOG TECHNIQUES
Low-Distortion FSK Generation Explained
Build a Precision Linear Encoded Display
Embedded Linux-Based Vehicle Tracking
Repurpose a Development Platform
Parameter Passing in C Language
CIRCUIT
PROGRAMMAB
Retrocomputing with Programmable Logic
Microprogramming with FPGAs
Addressing Memory Failures
Digital Modulation Theory
6LoWPAN Explained
Circuit Cellar's 2009
Archive CD Now Available
12 complete issues (plus BONUS ARTICLES!) on one CD
Comprehensive, cutting-edge project articles with source code and schematics
Searchable PDF format
Convenient month and year overview
Complete your collection for just $24.95!
FREE SHIPPING IF YOU ORDER BEFORE MAY 31!
VISIT WWW.CIRCUITCELLAR.COM/2009CD

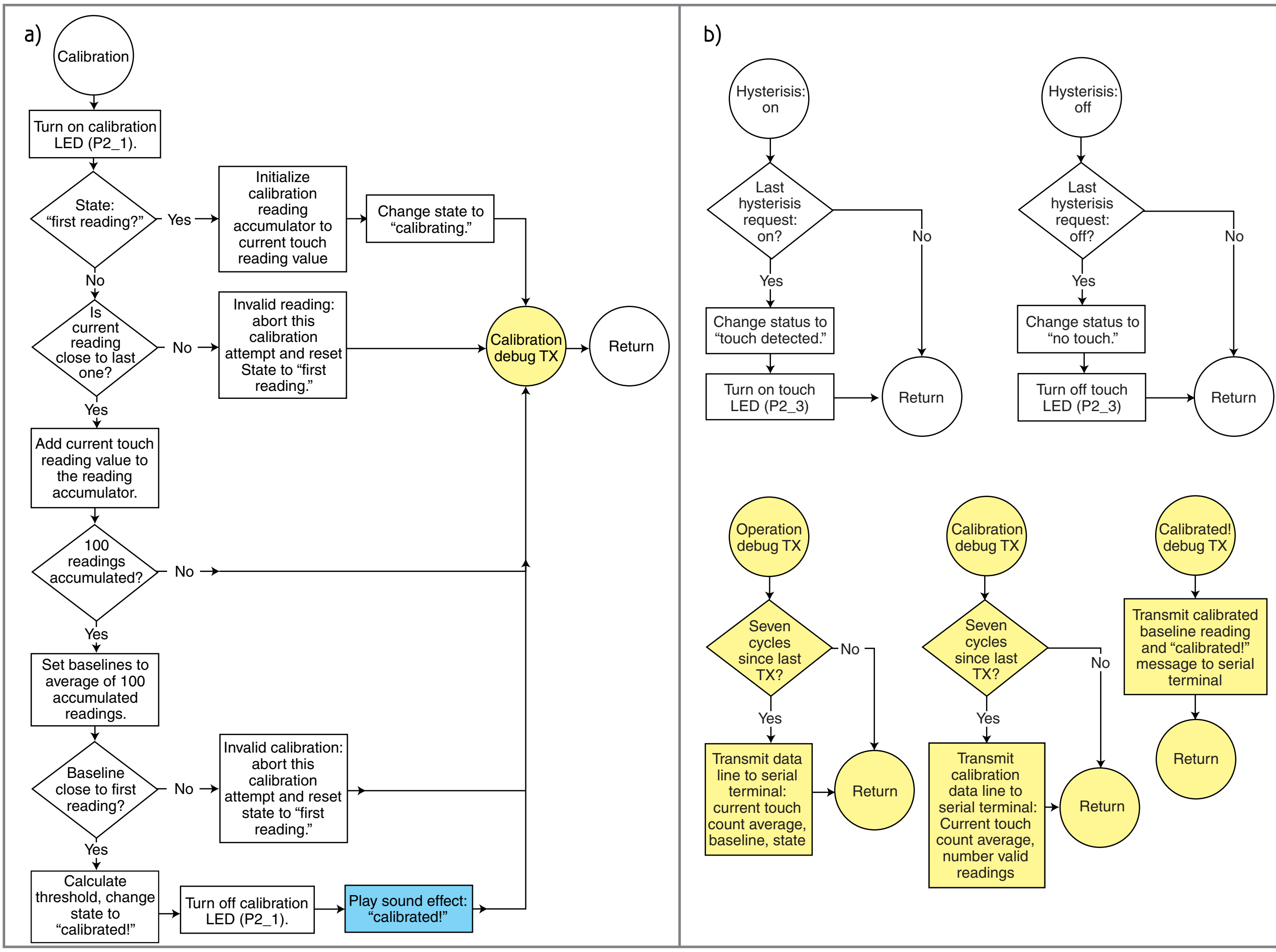


Figure 6a—This is the calibration flowchart. **b**—This is the hysteresis and terminal flowchart.

directly proportional to the resistances of R_2 and R_4. In particular, a simple and convenient formula applies:

$$\text{Gain} = \frac{R}{1{,}000}$$

Therefore, with R_2 and R_4 both at 100 kΩ, the effective gain is 100× (an 80-mV_{PP} signal input will result in an 8-V_{PP} amplified signal output). If R_2 and R_4 were changed to 1 MΩ, the effective gain would be 1,000×, which is within range.

INTERCOM FIRMWARE

The intercom system's PSoC firmware, programmed mainly in C (with assembly interrupt code), is represented in Figure 5 and Figure 6. The first section shows the main high-level operation flowchart, which begins by initializing user modules, variables, and the title line of the optional serial terminal display.

The audio pathway from the subject box to the control room box is enabled, and self-calibration begins. (The StartupSlide.mp3 sound effect is played.)

During calibration, a baseline touch sensor reading and an appropriate touch-sensing threshold are determined. One hundred consecutive readings are taken;

Listing 1—This is the intercom PSoC main loop code.

```
while(1)
    {
       mHandleTouchSensing
       if(CheckPttButt)
       {
           if(!TouchSensed)
           {
               PlayChirp(TouchSensedChirp);
               SubjectAudio;
           }
       }
       else
       {
           if(TouchSensed)
           {
               ControlAudio;
               Wait(CyclesForAnalogToSettleBeforePlayingSoundFx);
               PlayChirp(TouchReleasedChirp);
           }
       }
    }
```

at any point, if two readings are too far apart, this process is restarted (noisy readings or someone touched it). After 100 consecutive readings close to each other are taken, if the last reading is too far from the first reading, this process is restarted (baseline is still settling/drifting). After 100 consecutive readings close to each other are taken with the first and last readings close to each other, if the average of these readings is too far from the first reading, this process is restarted (noisy readings or someone touched it). If all the previous tests are passed, calibration is finally complete (typically takes 5 to 10 s) and the CalibrationComplete.mp3 sound effect is played.

Then an endless cycle begins, allowing the control room to touch the sensor area and enable the reverse audio pathway. (Control room to subject: the ControlActive-TouchDetected.mp3 sound effect is played.) When the hand is removed from the sensor area, the normal audio pathway is enabled. (Subject to control room: the SubjectActive-TouchReleased.mp3 sound effect is played.) The high-level main loop code is in Listing 1.

Touch-sensing is performed by measuring the frequency of the transport delay/capacitance oscillation created by the three digital inverter modules linked together in a loop. (The PSoC Designer module routing is shown in Photo 3.) This oscillation slows down when a hand approaches and touches the sensor area, which affects an antenna connected to the external part of this digital inverter loop.

The actual frequency measurement requires two timer modules, TMR_TouchCap (8-bit) and TMR_Touch (16-bit). The touch timer is clocked by the digital inverter oscillation, and the TouchCap timer will generate a capture pulse to read and reset the touch timer. This effectively measures the oscillation frequency produced by the digital inverter loop and provides a capacitive touch input.

EXPERIMENTS TO COME

In this article, I introduced an otoacoustic probe amp and touch-to-talk intercom system that I designed for a client to perform a wide variety of professional otoacoustic studies in a laboratory. I covered the concept of otoacoustic emissions and how they can be used to test the human ear without requiring conscious interaction with the subject.

In the second part of this article series, I will describe my amateur otoacoustic experiments in detail. Once I finished this product for my client, I was not satisfied because I had not seen any actual OAEs yet. I designed a low-cost solution to utilize my computer, a PSoC, and the high-gain probe amp to stimulate and capture OAEs for myself.

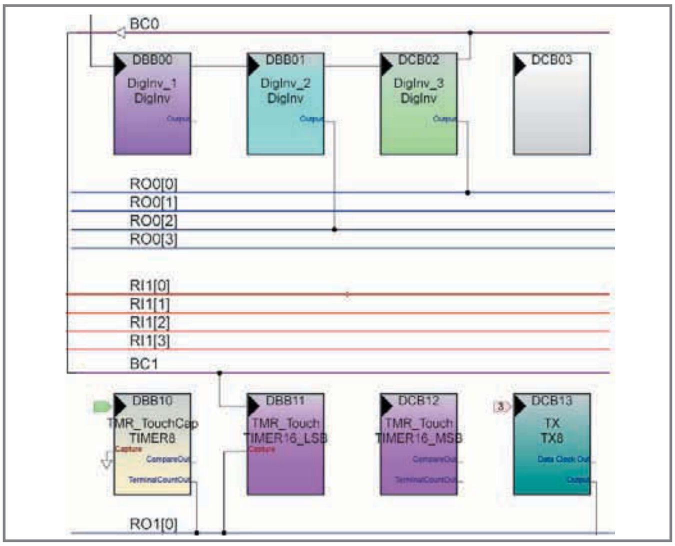


Photo 3—This is the PSoC touch-sensing user module routing.

Chris Paiano wrote over 30 application notes for the Cypress PSoC chipset, including such novelties as PongSoC and the Video RTA. Applicable links and information are available on his website (www.cpeproto.com).

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/238.

SOURCES

CY8C29466 PSoC
Cypress Semiconductor Corp. | www.cypress.com

LT1790-1.25 Precision reference
Linear Technology Corp. | www.linear.com

LM4562 Audio op-amp
National Semiconductor Corp. | www.national.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*.

For more need-to-know information about topics covered in Chris Paiano's Issue 238 article, the *Circuit Cellar* editorial staff highly recommends the following content:

—

Sound Effects Processing
by Robert Papp
***Circuit Cellar* 216, 2008**

Robert built a sound effects processor, and the system's keyboard is used for navigating through a user menu. You can set the volume level and select effects. Topics: Op-Amp, Audio, Sound Processing, Filter

Go to: www.circuitcellar.com/magazine/216.html

—

PSoC Design Techniques (Part 1)
Build an Eight-Channel Mixer
by Chris Paiano
***Circuit Cellar* 216, 2008**

Chris covers several useful PSoC design techniques. He also presents an eight-channel mixer with adjustment knobs. Topics: Audio, Mixer, Switched Capacitor, Sound Processing

Go to: www.circuitcellar.com/magazine/216.html

by George Martin (USA)

Put C Language to the Test (Part 2)

Program Specifics

In the first part of this series, you learned about an interesting project for implementing C language: a program for solving 2 × 2, 3 × 3, and 4 × 4 Sudoku puzzles. Now that you have a starting point (C language) and an ending point (a program), it's time to focus on the programming details.

Welcome back. Let me share an interesting story about the genesis of this project. I attended the 2009 Embedded Systems Conference (ESC) in Boston. I don't go each year, but I attend every two or three years because it makes the changes more apparent. This time around, I made two main observations. My first was that there wasn't a rush of high school students to the main exhibit floor in the afternoon. Where are they? Are they lost to our industry? These were questions for which no one in attendance had any answers. The second observation was the presence of *Elektor* magazine. "What's up with *Elektor* magazine?" I wondered. It reminded me of the fun projects I used to see a long time ago in this industry. I was intrigued and got a subscription. While I was in reading one of the free copies handed out at ESC, I came across their version of the Sudoku puzzle. I had never seen 4 × 4 using hex characters before. I used to be able to multiply hex numbers in my head, but now I can't even add decimal numbers. I'm surrounded by a host of computers that do that better than I ever could and I've lost the capability.

I was looking for a problem for our first C program. If I got too involved on hardware, I'd lose the readers who weren't familiar with the hardware I selected. Too much hardware design and I would lose readers too. Well, I ended up asking our editors if *Circuit Cellar* could reprint *Elektor*'s Hexadoku puzzle. Now, I didn't exactly say I was going to reverse-engineer the solution. The answer came back: yes. Now what was I going to do when they found out what I was up to? Oh well, it's been a good ride so far. And you know what ended up happening? Well, *Circuit Cellar* and *Elektor* subsequently joined forces—thanks to my efforts, of course. There I go wasting ink again.

On a serious note: congratulations to all.

Last time, I presented a problem: write a program to solve 2 × 2, 3 × 3, and 4 × 4 Sudoku puzzles. I also committed to get this running on a PC using C language. Next I found several free (as in beer) C compilers and selected one to use. I did a first pass at the design in my head and in notes. I then captured that design using a UML diagram. I also decided to use the PC's keyboard and display for input and output. I hinted at using files to aid in the development and testing processes. If you did not read the article, please look it over (G. Martin, "Put C Language to the Test (Part 1)," *Circuit Cellar* 236). I also presented a first pass at coding the implementation presented in the UML diagrams.

I initially thought a two-part article would be enough to explain the design. I was going to make this the last of my discussion about the problem, but then I thought about it a bit. Based on the emails I'd been receiving, I thought it best to explain more about the current design and what the next steps should be. I realized that this is a general class of problems that you might come across—one where the user enters data into a database and then that database is used to drive an embedded system. One such project is a scheduling system that I'm working with. The user enters scheduling data in the form of the day of the week and the time of the day. For example Monday, Wednesday, Friday at noon, schedule an event. And then the embedded system looks through that database and performs the appropriate action at the correct time. It's very similar to what we're talking about in this example.

UML

Let's look at the UML diagram for the original design (see Figure 1). Now compare that to the UML diagram in Figure 2 for the code as it stands today. It looks a lot different, but actually the aren't as many changes as a reorganization. I put the routines in the files where they are defined. I still have a yet-to-be-completed file of Rules, Errors, and Hints. I expect this to be broken into three separate files, one for each of the names.

So that's the overall design and how it took shape. Now let's look into the specifics. And let's start with Main. Sometimes my `main()` function is huge and other times it is tiny. Today, it's tiny—less than 100 lines in the file. I settled on a design approach that inputs characters from the keyboard and outputs characters to the screen. I read characters until a complete line is recognized and call the command-processing routine (`ProcessCmd(char *)`). I pass the command-processing routine a pointer to the string that holds the new line of characters. Remember: C strings are terminated using a NULL character, so be sure to save space for the terminator.

Refer to the `ProcessCmd(char *)` routine. (The code is posted with Issue 236 on the *Circuit Cellar* FTP site.) It's also very small. I kept it this way to reduce the parameter passing. Each command is tested, and if a match is found, a pointer to the same string of inputted characters is passed to the routine that will process the specific command. Those routines are found in the file that declares the database in memory. That file knows all about the data structures and the details are global to that file. Again this implementation has very little parameter passing and sharing of data. The routine `main()` reads the keyboard and passes a pointer to a string to `ProcessCmd(char *)`. It matches up input characters with commands and passes a pointer to that same string to the detailed commands. Each of them uses that same string and works with the database.

I settled on this approach because I knew I wanted to be able to work with files on this design. It will be very straightforward to plug in any file operations, and it's probably time to talk about file I/O.

FILE I/O

The C language supports reading and writing a file. This was once uncommon in embedded systems, but not so today. The file could be from a disk on a PC or a compact flash card in your embedded system. Look at the `ReadFile-ByName(char *c)` routine as I describe the process of reading a file. The first operation is to

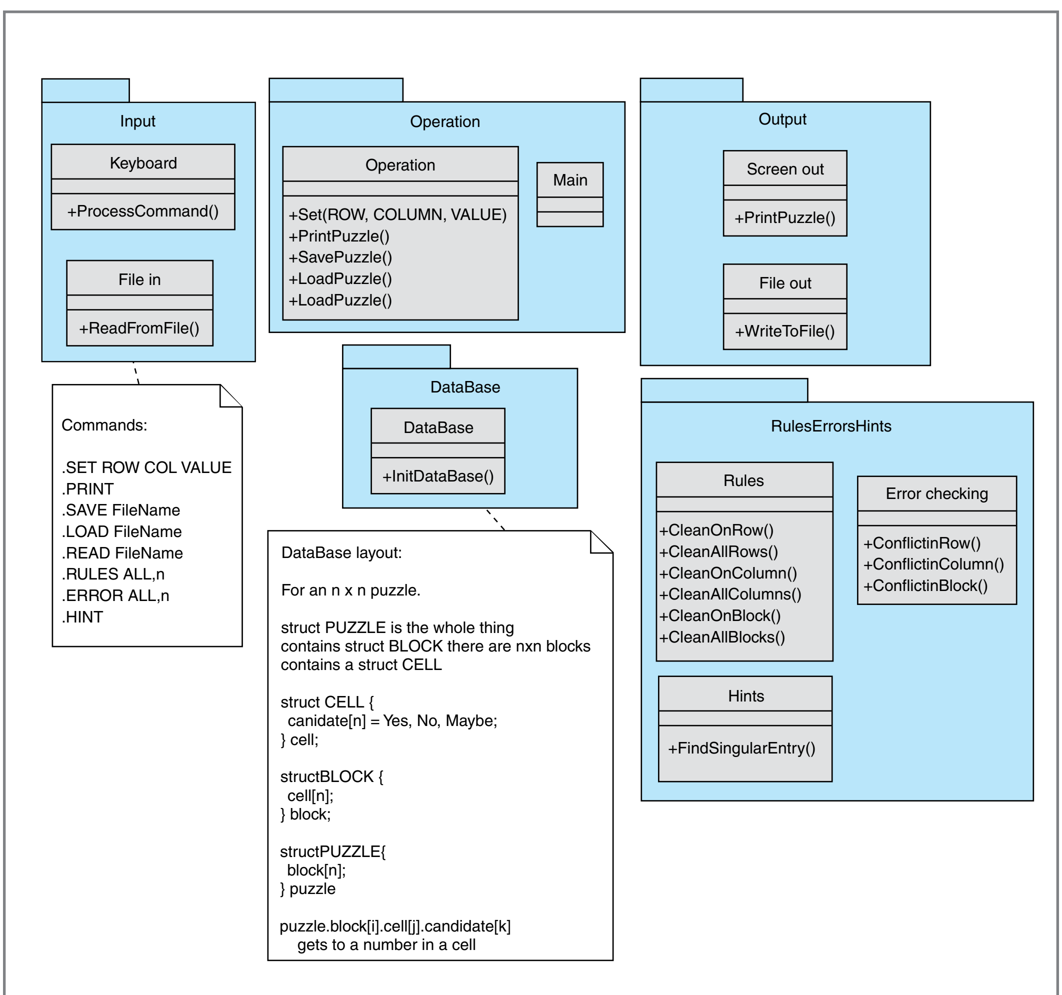


Figure 1—This is the original UML diagram.

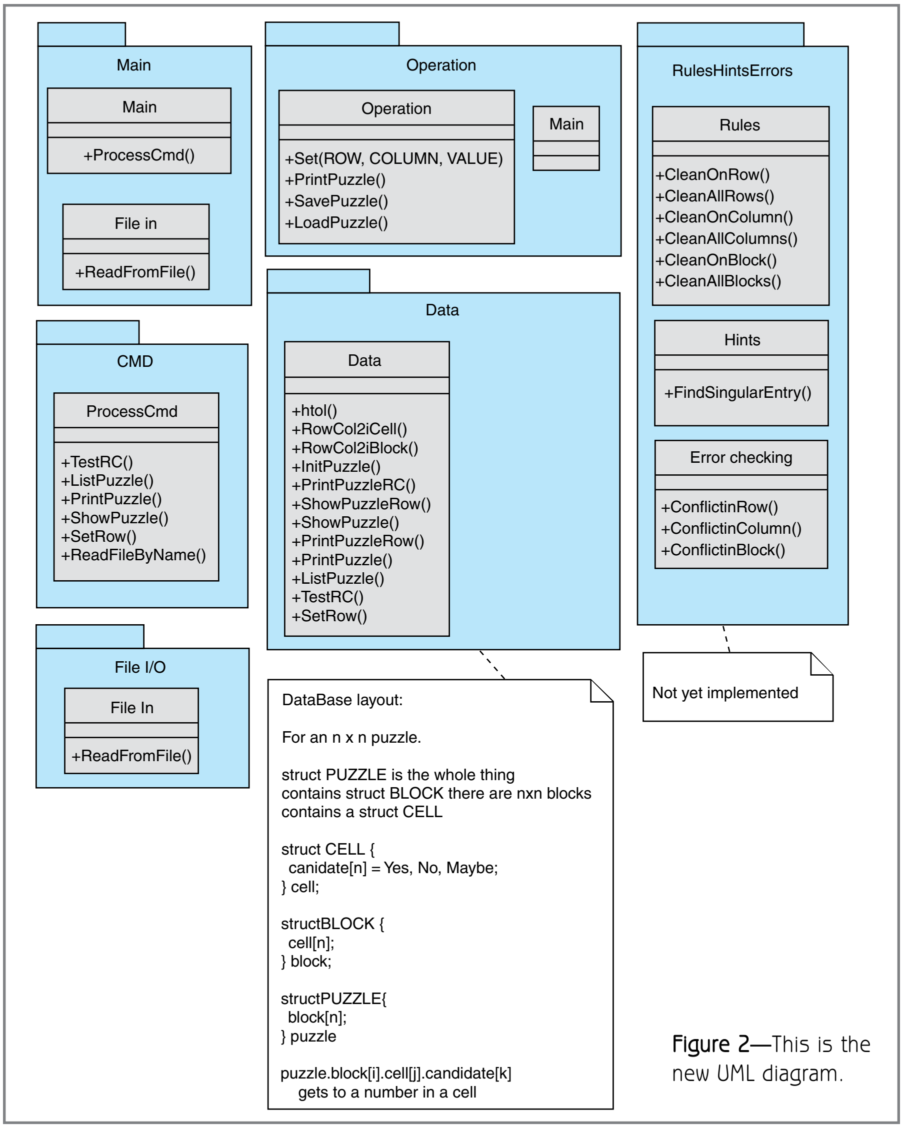


Figure 2—This is the new UML diagram.

open the file for reading. That's done with the command `fp_in = fopen (InFileName,"rb")`. Let's break that instruction into its different components. The variable `fp_in` is a pointer to a type of variable known as `FILE`. If you dig deeper in the C language, you will find that pointer described in more detail. But for now, let's just say it's a pointer to the variable of type `FILE`. If the command is successful, a non-NULL pointer is returned and points somewhere into your memory. If the command fails, a NULL pointer is returned. Thus the test just after we tried to open the file. Two parameters are in the command to open the file. The first is a pointer to a string containing the filename. The second is the mode as to how we are going to open the file. In this case, it's opened for read (r) and binary (b). We could open it for writing by passing the `w` parameter.

Once the file is opened, you read from that file with `Line = fgets(FileInBuf, 100,fp_in)`. The `fgets` command reads a line of data from the file. The variable `FileInBuf` is a place to hold that data. The value `100` tells to routine to read at most 100 characters and the `fp_in` is a pointer to which file will read from. You can have several files opened. I next test for the value of the `Line` variable. If it's a NULL, nothing was read and it's time to exit. If it's an end of file (EOF), then it's also time to exit. If it's neither of these, then we have a line read from the file in the string `FileInBuf` and a pointer to that string in `Line`. And doesn't this pointer to a buffer that holds the command look a lot like how we read characters from the keyboard? The code does not reflect the fact that we could call `ProcessCmd(Line)` and the code would not know if the data came from the keyboard or a file.

This is an important point. You could build a file with a set of commands and read that file into the program (automating the testing process). The program would then execute the commands and you could print out the results. Using DOS, you could pipe that output into a file and read the file at your leisure looking for proper operation. This is very straightforward to accomplish and should make your life easier. If you have different test files, you could run each into the code and save the results. Then with a file compare program, you could look at each change in the code and see if the results were the same. Or, as customers report problems, you could build test cases for each and prove that they were corrected. Being a hero is your job!

PRINT STATEMENT

I said last time the `printf()` was a C command and you should just look it up on the web. While that's still true, let's talk a bit about that function. In its simplest form, `printf("OK");` sends the string `OK` to the standard output device. In our case, it's the screen. And

```
print
|----|----|----|----|
|5.0.|B2..|9...|....|
|..D.|.16.|F.A2|37.B|
|..EB|D.7A|6.4.|2..F|
|A73.|..E4|..B0|9.1.|
|----|----|----|----|
|.A.3|.B.8|...1|4C..|
|10C6|..9.|4..A|B.E5|
|..F.|A5..|.96.|1820|
|E.28|0.16|..D7|FA9.|
|----|----|----|----|
|..8.|C9..|1..E|...7|
|2.41|7..5|.69.|A0..|
|...D|6..1|.B..|.3..|
|8...|.ADE|0..B|..32|
|----|----|----|----|
|..A.|.C.B|.2E6|.F..|
|....|....|....|....|
|.4.E|.8.3|.15.|0...|
|----|----|----|----|
Command OK
```

Figure 3—Output of the `Print` command

```
show
==== |==== |==== |==== ==== |==== |==== |==== ====   |====    |====   |==== ====  |==== |==== |====
     |0123 |0    |0123|     |  2  |0123 |0123 |      |0123   |0123  |0123 |0123 |0123 |0123 |0123
 5   |4567 |     |4567|     |     |4567 |4567 |      |4567   |4567  |4567 |4567 |4567 |4567 |4567
     |89AB |     |89AB|   B |     |89AB |89AB | 9    |89AB   |89AB  |89AB |89AB |89AB |89AB |89AB
     |CDEF |     |CDEF|     |     |CDEF |CDEF |      |CDEF   |CDEF  |CDEF |CDEF |CDEF |CDEF |CDEF
---- |-----|-----|---- -----|-----|-----|---- -------|-------|------|-----------|-----|-----|----
0123 |0123 |     |    |     |0123 |     |     |      |0123   |      |0123 |  2  |0123 |0123 |
4567 |4567 |     |    |     |4567 |   7 |     |  6   |4567   |4     |4567 |     |4567 |4567 |
89AB |89AB |     |   B|     |89AB |     |  A  |      |89AB   |      |89AB |     |89AB |89AB |
CDEF |CDEF |  E  |    | D   |CDEF |     |     |      |CDEF   |      |CDEF |     |CDEF |CDEF |   F
---- |-----|-----|----- ----|-----|-----|------------|-------|------|----- -----|-----|-----|----
0123 |     |0123 |   3|0123 |     |0123 |     |0123  |0123   |0123  | 1   |     |     |0123 |0123
4567 |     |4567 |    |4567 |     |4567 |     |4567  |4567   |4567  |     |4    |     |4567 |4567
89AB |  A  |89AB |    |89AB |   B |89AB |8    |89AB  |89AB   |89AB  |     |     |     |89AB |89AB
CDEF |     |CDEF |    |CDEF |     |CDEF |     |CDEF  |CDEF   |CDEF  |     |     |C    |CDEF |CDEF
---- |-----|-----|----------|-----|-----|------------|-------|------|---- ------|-----|-----|----
0123 |0123 |     |0123|     |     |0123 |0123 |0123  |       |      |0123 | 1   |     |  2  |0
4567 |4567 |     |4567|     | 5   |4567 |4567 |4567  |       |  6   |4567 |     |     |     |
89AB |89AB |     |89AB|  A  |     |89AB |89AB |89AB  | 9     |      |89AB |     |8    |     |
CDEF |CDEF |   F |CDEF|     |     |CDEF |CDEF |CDEF  |       |      |CDEF |     |     |     |
---- |-----|-----|---- -----|-----|-----|------------|-------|------|-----------|-----|-----|----
0123 |0123 |     |0123|     |     |0123 |0123 | 1    |0123   |0123  |     |0123 |0123 |0123 |
4567 |4567 |     |4567|     |     |4567 |4567 |      |4567   |4567  |     |4567 |4567 |4567 |   7
89AB |89AB |8    |89AB|     | 9   |89AB |89AB |      |89AB   |89AB  |     |89AB |89AB |89AB |
CDEF |CDEF |     |CDEF|C    |     |CDEF |CDEF |      |CDEF   |CDEF  |  E  |CDEF |CDEF |CDEF |
---- |-----|-----|----------|-----|-----|------------|-------|------|-----------|-----|-----|----
0123 |0123 |0123 |    |     |0123 |0123 | 1   |0123  |       |0123  |0123 |0123 |   3 |0123 |0123
4567 |4567 |4567 |    |  6  |4567 |4567 |     |4567  |       |4567  |4567 |4567 |     |4567 |4567
89AB |89AB |89AB |    |     |89AB |89AB |     |89AB  |   B   |89AB  |89AB |89AB |     |89AB |89AB
CDEF |CDEF |CDEF | D  |     |CDEF |CDEF |     |CDEF  |       |CDEF  |CDEF |CDEF |     |CDEF |CDEF
---- |-----|-----|----------|-----|-----|------------|-------|------|-----------|-----|-----|----
0123 |0123 |     |0123|0123 |     |0123 |     |0123  |  2    |      |     |0123 |     |0123 |0123
4567 |4567 |     |4567|4567 |     |4567 |     |4567  |       |      |  6  |4567 |     |4567 |4567
89AB |89AB |  A  |89AB|89AB |     |89AB |   B |89AB  |       |      |     |89AB |     |89AB |89AB
CDEF |CDEF |     |CDEF|CDEF |C    |CDEF |     |CDEF  |       |  E   |     |CDEF |   F |CDEF |CDEF
---- |-----|-----|----------|-----|-----|------------|-------|------|-----------|-----|-----|----
0123 |     |0123 |    |0123 |     |0123 |   3 |0123  | 1     |      |0123 |0    |0123 |0123 |0123
4567 |4    |4567 |    |4567 |     |4567 |     |4567  |       | 5    |4567 |     |4567 |4567 |4567
89AB |     |89AB |    |89AB |8    |89AB |     |89AB  |       |      |89AB |     |89AB |89AB |89AB
CDEF |     |CDEF |  E |CDEF |     |CDEF |     |CDEF  |       |      |CDEF |     |CDEF |CDEF |CDEF
---- |-----|-----|----------|-----|-----|-----------  |-------|------|----------  |-----|-----|----
==== |==== |==== |==== ==== |==== |==== |==== ====   |====    |====   |==== ====  |==== |==== |====
Command OK
```

Figure 4—The output of the `Show` command

`fprintf(fp_out,"OK");` sends the string 'OK' to the file pointed to with `fp_out`. You will need to have previously opened that file. So, it's fairly easy to print text. Characters like `\n` and `\r` represent control characters. In this case, `\n` is new line and `\r` is carriage return.

The next step in printing is to print a variable. The statement `printf(" %1X NO \n\r", i);` prints one space, the variable `i` in hexadecimal format, one space, the text `NO`, one space, a line feed, and a carriage return. Simple enough? The capital `X` means the output is 0..9, A..F. A lowercase `x` would print 0..9, a..f. The number `1` in front of the `X` means just one digit wide. A `%02X` would mean two characters wide and put in a leading zero if required.

There's a lot more to the `printf();` statement, and it is worth looking at a good book on the C library functions or searching the web. I find the more examples I have on the `printf();` statement the easier it is to get it to do what I want it to do.

As you look through the code, you will find other C library functions. One such function is `strncmp(Line, "ROW"', 3);`. If you look it up, you should find that it compares string1 (`Line`) to string2 (`ROW`) for 3 characters and returns the value of the comparison. I'm looking for an exact match, and a return of 0 would indicate that exact match. You should get to know the routines available in the standard C libraries. It's valuable material and included in today's C compilers. You should also learn about the classic pitfalls in using some of them.

I've got several output functions working. One prints the puzzle in its most compact form (the `PRINT` command). One space per cell (see my definition of cell) is used. That space holds a digit if that digit is known or a blank space if the digit is unknown. The next type of printing of the puzzle is to explode each cell into all possible digits. (This is the `SHOW` command.) If a digit is a possible candidate, then it is printed. If it is not a possible candidate, then it is not printed. The last form of printing is to dump the database in mostly text form. This is good for debugging (the `LIST` command).

CODE EXAMPLES

Now for some examples of the code as it stands. Figure 3 is the output of the `PRINT` command after we've initialized the puzzle and entered the clues that were given. Figure 4 is the output of the `SHOW` command on the same database as in Figure 3. Using this command, you can see how the rules might start to be applied. I would start with a row, and every time you find a number that's certain, you would eliminate that number from all other columns in that row. Then use the same rule for columns and blocks. In the files sections, you will find copies of the figures and more.

RUNNING RULES

Next time, I plan to have the rules up and running. The input from a file will be changed to look like the input from the keyboard.

George Martin (gmm50@att.net) began his career in the aerospace industry in 1969. After five years at a real job, he set out on his own and co-founded a design and manufacturing firm (www.embedded-designer.com). His designs typically include servo-motion control, graphical input and output, data acquisition, and remote control systems. George is a charter member of the Ciarcia Design Works Team. He is currently working on a mobile communications system that announces highway info. He is also a nationally ranked revolver shooter.

SOURCES

CC386IDE
LadSoft | http://ladsoft.tripod.com/cc386.htm

MagicDraw Modeling tool (v.9.5)
No Magic, Inc. | www.magicdraw.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*.

For more need-to-know information about topics covered in George Martin's Issue 238 article, the *Circuit Cellar* editorial staff highly recommends the following content:

—

Put C Language to the Test (Part 1)
A Sudoku Puzzle-Solving Program
by George Martin
***Circuit Cellar* 236, 2010**

Ready to write a program in C language? This program is for solving a Sudoku puzzle. Topics: C Language, Program, Sudoku, IDE, UML, database

Go to: www.circuitcellar.com/magazine/236.html

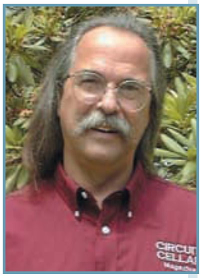
—

Text Adventure Gaming
by Chris Cantrell
***Circuit Cellar* 205, 2007**

Do you like learning new languages? Try using a chip and the Cog Coordination language (CCL) for an interesting gaming application. Topics: Language, Programming, CCL, RISC, SD Card, FAT

Go to: www.circuitcellar.com/magazine/205.html

by Jeff Bachiochi (USA)



Machine Control

Customize and Implement MCU-Based Control Circuitry

You can use your design skills to control virtually any machine. Doing so is just a matter of proper preparation and planning. This article details the process of building an MCU-based tread speed control system. Try these design techniques in your own project.

I don't normally charge willy-nilly into a project without proper background material, but maybe my tale of woe can help others. It all started when a pleasantly extended Fall 2009 ended with a rapid drop in daily temperatures. As Jack Frost began his winter takeover, I was forced to move my outside jogging regimen indoors where I could control the climate. Basically, I dusted off the old treadmill in my basement and started training.

Running indoors kept me out of the cold weather, but I quickly became bored with the unchanging scenery of my cluttered basement. The TV hanging from the ceiling was of no use since the industry's change to a digital transmission standard. I couldn't take my mind off the droning rhythm of my feet pounding to a permanently fixed beat. I was about to call it quits when I got an idea. "Wait, that's it," I said to myself. "When I run in the neighborhood, it's up and down—my gait constantly changes. That's what's missing. I need to be able to control the tread speed by some process other than constantly pushing buttons on the up front control tray." My inner engineer was taking over.

In this article, I'll explain how I built an MCU-based tread speed control system. As you'll see, you can design control circuitry for just about any motorized application imaginable.

MAP

Hidden beneath the electronics shield that covers the motor compartment is a one-sheet treasure map. This map's block diagram clearly shows the direction I must travel to get at the treasure. A single wiring harness connects the motor and power supply in the base to the user panel in the tray.

I needed to hijack signals coming out of the user panel. The harnesses connections were clearly labeled on the map and included power, ground, and an output signal, "Spd Cntrl 0-5 VDC." OK, simple. I just needed to read the analog output, redefine it, and recreate my own version of the analog output. I could hardly wait to peruse micros for one that had both A/D and D/A peripherals.

To make a long story short, I chose a Microchip Technology PIC16F1934. I started out slow. My first plan was to use one channel of an ADC to read the analog signal being requested by the user control panel and just transfer the value to the DAC. I figured this would simply reproduce the original signal. The ADC on the '1934 had 10 bits of resolution. The DAC had only 5 bits, but I thought that would be plenty for this solution.

One of the reasons for choosing the 40-pin version of this device was to make use of the I/O. As you can see in Figure 1, I used 16 outputs as drive to two ports of LEDs. I created two bar graphs so I could visually follow some parameters (see Photo 1). While it does not make much sense to monitor input and output values (when they are identical), it will be helpful to have these operating for future revisions of this project. Most of the important stuff happens via interrupts. The analog conversions are

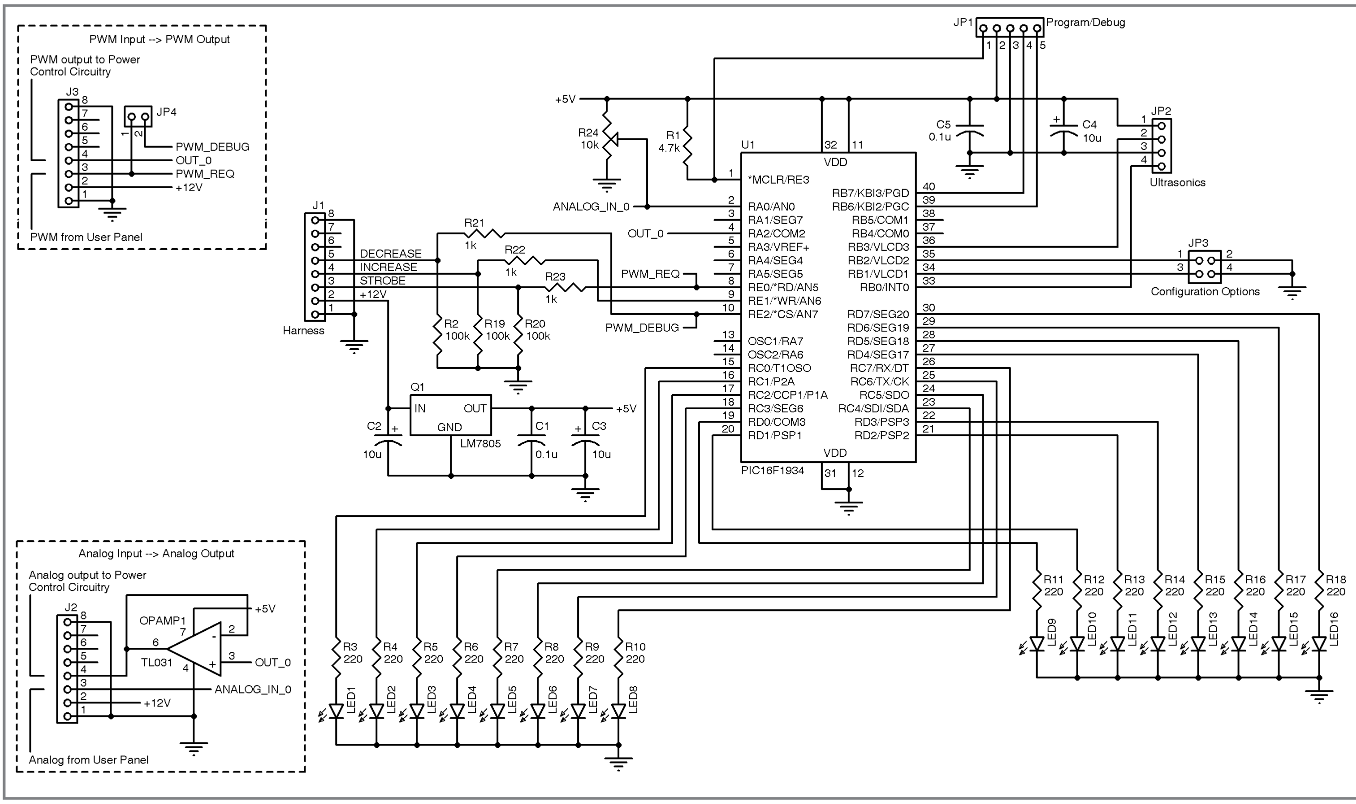


Figure 1—The PIC16F1934 handled the radical changes I asked of it without a hitch. The schematic for this project went through a number of alterations.

self-timed and signal when a conversion is ready. The A/D interrupt saves the current conversion value to a temporary register and raises a user flag. I'll use this flag in the main program loop to indicate when a new value is ready. When using a microcontroller with one level of interrupt priority, it is a good idea to keep all interrupt routines as short as possible to keep them from causing objectionable delays with other time-critical interrupts.

A second interrupt is used to increment the variable `Tick`. It is used to allow the main loop to run at a slower predetermined rate. In this case, the bar graph LEDs can be updated a few times a second. Timer 1 is set up to overflow at its maximum time of 65 ms based on the 4-MHz oscillator timebase. Every time Timer 1 rolls over, the interrupt routine increments the `Tick` register. The main routine can use this variable to determine delay times (in increments of 65 ms).

I like working with bar graphs because it is easy to make them display data in various formats. I define values for each bit in the bar graph—in this case, 8 bits with a value of 0 to 255 (see Table 1). You might want a linear display with bit values equally spaced or some nonlinear display.

You can write the LED display routine to turn on all bits (starting at bit 0) representing those values up to the display value or just turn on the highest bit equaling the display value. This might even show two values (present and peak) by using the first method for displaying the present value and second method for displaying the peak value. You might want to blink the peak value, which helps it to stand out from the present value.

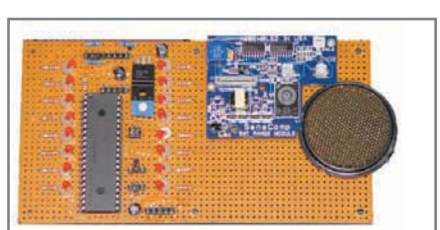


Photo 1—Two vertical rows of LEDs are used as LED bar graphs to give visual feedback of two parameters. The left row of LEDs is used to display distance measurements (body position on the treadmill) in 6" increments. Ultransonics (on the right) are used to measure this distance.

Before using this bar graph routine, you might need to do some computation to adjust the display value. The range of values you want to display might exceed 8 bits. In this case, you may need to do a division to get it into the right range for display. You can also play around with doing other computations that may linearize, shift, or perform some other function in order to make better use of the way you predefine the display routine. I'll give a better example of this in the next section.

POSITION

The Polaroid instant camera that I remember (circa 1980s) probably wasn't on anyone's Christmas list this year. I think Polaroid may have been the first to use ultrasonics to autofocus a camera (see Photo 2). A trip to its website shows new digital cameras in the product line, including one with a built-in printer to carry on the tradition of instant photos. This is a far cry from the original concept of putting a complete darkroom in each camera.

To this day, the Polaroid ultrasonic sensor remains a staple for anyone

Bit number	Linear representation	e^x Representation
0	0–31	0
1	32–63	1
2	64–95	2
3	96–127	3–5
4	128–159	6–13
5	160–191	14–35
6	192–223	36–94
7	224–255	95–255

Table 1—Based on a 0-255 display value, this shows two possible bit definitions for linear and nonlinear bar graph formats.

needing distance measurements up to about 50″. (You can find these at www.senscomp.com.) The basic sensor unit is made up of two parts, the transducer and the transmitter/receiver PCB. It requires 5 VDC and at least one digital input and output from an external source like a microcontroller. When the micro raises the sensor unit's `init` input, its transmitter whacks the transducer with 16 cycles of 40 kHz at a few hundred volts. (It feels like touching an electric fence.) Since this 40 kHz is above my hearing level, I only hear the snap of the transducer coming to life. The transducer emits the burst and the waves travel outward toward any obstacles within its path. Upon hitting an object, some reflections are received by the transducer and registered by the receiver. The receiver reports the echo by raising its (open-collector, don't forget to add a pull-up) output that the micro uses as an "I saw something" signal. By timing the duration between sending and receiving the burst, the micro can determine the time for the burst to travel out and back from an object. At about 1′/ms, if the measured time is 10 ms, the distance to the object is about 5′ (5 ms out and 5 ms back).

By placing the transducer just below the tray on the treadmill, I can measure the distance to an object (a body) anywhere on the tread area. The total tread length is 4.5′ with the transducer placed 1′ in front of the tread. This puts the optimum distance at the tread's halfway point (center of the tread or 2.25′) plus the distance to the transducer (1′) or a optimum measured distance of 3.25′. On an 8-bit bar graph, 6″ increments would be just about right, assuming the value placed in the `DISTANCE` register by the sensor measurement routine is in inches. We need to transform this into values that the LED bar graph routine can display properly.

Refer to Table 2. You can use simple integer math to transform the actual measurement values into values that are more meaningful. The alternative is just to alter the values in the LED bar graph routine to represent the values in inches. When I'm debugging, I use the LED bar graph routine to display other values in real time, so it makes more sense for me to use a standard display routine that can be called by any routine wanting to use the display.

Since I have a timer that is already counting in 1-µs increments (Timer 1), this can also measure the ultrasonic echo. If I read this (free running) timer when I trigger the ultrasonic echo and read it again when the echo returns, I can do some subtraction to find the number of microseconds for the total round-trip 40-kHz burst (see Figure 2). Actually, the distance (in inches) can exceed a byte; however, you are concerned only with values less than 60. An overflow (greater than 255) is treated as 255. This includes an overflow of Timer 0, which is started with the initialization of the ultrasonic burst, and set to overflow 65536-µs later, for timeout purposes (in case the ultrasonic device does not return an echo).

One of the main loop's tasks is to initiate distance measurements. Once enabled, a counter peripheral will be triggered by the echo's rising edge. The counter automatically grabs the Timer 1 count so a distance computation can be performed. The DISTANCE value goes through another computation prior to the LED bar graph display (discussed earlier) to adjust the distance value for the display of distance in 6″ increments.

Photo 2—Polaroid instant cameras have been popular for a long time. This model from the 1980s uses an ultrasonic sensor for auto-focusing.

ADJUSTING THE ANALOG

After preliminary programs to recreate the analog input with the DAC and displaying measurements made by the ultrasonic sensor, it was time to alter the analog value based on the distance measurements. The basis for this project was to allow body position to affect tread speed. I used the distance values to define three zones: a forward zone (increase speed), a steady zone (no adjustment), and a rearward zone (decrease speed). When I am in either the forward or rearward zone, the program will alter the analog signal to raise and lower the tread's speed without requiring me to touch any control.

In my office, I simulate the analog input using a potentiometer wired to the protoboard. One LED bar graph displays the D/A value (and follows the potentiometer adjustment). The other graph displays measurements from the ultrasonic sensor. Using an oscilloscope on the analog output of the DAC, I confirmed that this output was being adjusted whenever the distance measurement left the steady zone.

I brought the prototype down to the basement where my treadmill resided along with a scope to check voltage levels before actually attaching circuitry to the treadmill. I measured the power supply, which was good, and then the analog voltage, which was good too (0 V). So, I started the tread moving manually to see the analog voltage rise—and zing, up it went to 5 V. "But wait," I said. "Shouldn't that be a bit lower? It's pulsing." I then referred to my treasure map, and this time I read output signal, PWM Spd Cntrl 0-5 VDC. Dolt! It wasn't analog! That's what I get for rushing into this without performing proper background work.

REWIND

After I got a prototype built, did I have what I needed to go from analog I/O to PWM I/O? The PWM period was 50 ms while on. To achieve this PWM

Bit #	Distance (actual)	D = Distance – 12 (offset)	V = D x 5 (scalar)	Linear representation
-2	> 0 and < 6	–	–	–
-1	> 5 and < 12	–	–	–
0	> 11 and < 18	> 0 and < 6	> 0 and < 30	0–31
1	> 17 and < 24	> 5 and < 12	> 29 and < 60	32–63
2	> 23 and < 30	> 11 and < 18	> 59 and < 90	64–95
3	> 29 and < 36	> 17 and < 24	> 89 and < 120	96–127
4	> 35 and < 42	> 23 and < 30	> 119 and < 150	128–159
5	> 41 and < 48	> 29 and < 36	> 149 and < 180	160–191
6	> 47 and < 54	> 35 and < 42	> 179 and < 210	192–223
7	> 53 (and < 60)	> 41 (and < 48)	> 209	224–255

Table 2—A sensor value (in inches) is transformed into an LED bar graph value (0-255) by doing an offset adjustment and then scalar multiplication.

period, I would need to use a much slower crystal frequency on the micro as the on-board PWM peripheral's maximum dividers will only bring the available period down to approximately 1,000 Hz at the present oscillator rate. I'd never run an application at such a slow execution speed. Since I'd still need to do somewhat accurate distance measurements, I thought I might run into timing problems if I attempted to use the on-board PWMs. Certainly, I had plenty of power to bit-bang the PWM at the present speed, so I went with the software approach.

I was a little gun shy. Instead of measuring the input on time and recreating an output PWM with similar timing, I took a slightly different approach. Although I couldn't see a signal on the harness that carried any kind of synchronization between the user panel and the power circuitry, I wanted to avoid this as a possible issue. So, I synchronized my output PWM with the input PWM for the user panel.

Every PWM output pulse rises when the PWM input rises. The measurement of the input pulse duration is handled by a second counter similar to the ultrasonic measurement. This counter transfers the present Timer 1 count (microseconds) on alternating edges, the time between the rising edge, and the falling edge will be the duration. When the scope signals are on, the duration is between 10 and 40 ms. These values are used for verifying that the output "on" time does not stray from these minimum and maximum parameters. Initially, this isn't be a problem for recreating the input PWM signal for testing purposes.

On the first pass, I used a potentiometer to simulate analog input. I then created a PWM signal for simulation. I could still read the pot as before, but the value (0 to 255) wouldn't be used to run a DAC. Instead, the value was used as the on time of a separate PWM output (not to be confused with the output PWM created from measuring the input PWM). Got it? So here's the run down. I performed an analog read of the potentiometer position and created a PWM with a 50-ms repetition rate whose on-time duration was a function of the pot. I then treated the simulation signal as if it was a PWM from the user panel, measured it, and reproduced the signal as a PWM output (through the harness to the power circuitry).

Once the simulation test was successful, I could add the code to use the ultrasonic zone information to lengthen or shorten the "on" duration being requested by the user panel. This was handled by an offset variable that is adjusted by the zone information. It was the combination of this offset variable and the input PWM's on time that impacted the output PWM's duration fed to the power circuitry, and this was where the maximum and minimum checking becomes necessary. I didn't want to create an output PWM signal that went beyond the boundaries of what was expected.

```
40-kHz burst's r(ound-)t(rip) time (us) = RTTimeH:L
(RTTimeH:L)/2 = O(ne)W(ay)TimeH:L (us)
(OWTimeH:L)/1000 = OWTime (ms) or (feet)
(OWTime)*12 = DISTANCE (inches)
or
DISTANCE = (RTTimeH:L)*12/(2*1000) = (RTTimeH:L)/167
```

Figure 2—The speed of sound is used to calculate distance. This is roughly 1 ms/ft. Since the sound burst travels out and is reflected back, these sound waves travel twice the actual distance to the object.

RUNNING

OK, here we go, back to the basement. I gave a quick check to make sure things hadn't changed while I was away. Everything seemed to be in order. I built up a short piece of harness that allowed me to obtain power and also break the PWM connection, routing these to my prototype. I positioned the ultrasonics and started the treadmill. While positioned within the steady zone, the tread began to move and increase in speed up to a predetermined level. I moved forward and the ultrasonics sensed that I'd moved into the forward zone. A request was made for an increase in speed. The output PWM had obviously increased its duration because the tread began to speed up while I continued to stay within the forward zone. "Look, Ma. No hands."

And then, the treadmill shut down. An error code was flashed on the user panel as it all came to a stop. I sat down on the tread with my hands on my face mumbling something that I probably shouldn't print here. I pulled all the plugs, turned out the lights, and went back upstairs to think.

SAFETY FIRST

I shook off the urge to eat by having a large dish of *Forbidden Chocolate* ice cream while investigating error codes. There, on the backside of my treasure map, was a list of error codes with their causes. Let's see, E22: "*Overspeed sense*—the treadmill is running faster than the set speed for too long." It seemed that there were a number of checks and balances based on feedback. If I artificially requested more speed, the tachometer showed the tread was moving faster. The user panel's request for speed was expecting a certain number of tachometer pulses, but it was seeing an increased number of pulses. There must be something wrong, so shut everything down.

I'm not here to debate the definition of "robot," but only to suggest that Isaac Asimov's laws of robotics are important to all good designs. If you substitute the word "treadmill" for "robot" in these rules, you can see what I mean. User safety is the number one priority. Any algorithm used in interpreting a user's request must not override the safety issue. Any algorithm used to prevent system failure must not create a safety issue or ignore a user request. Introduced in his 1942 short story titled "Runaround," Asimov's three rules of robotics are important for all designers who wish to keep their creations friendly: a robot may not injure a human being or, through inaction, allow a human being to come to harm; a robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law; and a robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

I obviously created a safety issue for the closed-loop treadmill control. Asimov would smile at this. I felt defeated.

REWIND AGAIN

I could've tried to outthink the safety algorithm by adding or subtracting tachometer pulses or replace the user control panel altogether. The user panel had so many good features that I decided to take a new approach. I investigated the user panel more closely and found that the membrane keypad's switches were used to ground pulled-up inputs on a parallel latch. The control circuitry reads this latch to determine if the user is requesting input. I used an ohmmeter to associate each bit with a corresponding key on the front panel. I could simulate a key press with an open-collector transistor placed across each contact that I want to control. Using this method, the system's safety issues should not be compromised, as it wouldn't know that I didn't physically push the increase or decrease speed button.

Refer to Photo 3. I added an NPN transistor across the 2 bits I labeled as increase and decrease, along with some bias resistors to keep the transistor off (high resistance) just in case the microcontroller's output pins became tristated (or disconnected).

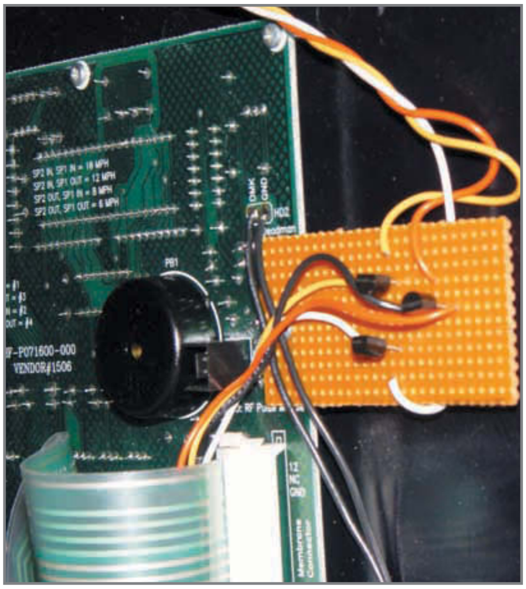


Photo 3—I added NPN transistors across each keypad input that I wanted external control over. Since each input already has pull-up resistors, the open-collector transistor shorts this to ground when its base has a positive voltage applied.

This all seemed pretty straightforward, so I rewired the microcontroller again and changed the code to put a high pulse on the appropriate transistors to simulate an actual increase or decrease button push. Then it was back to the basement.

I checked the voltages, attached the circuitry, and started the treadmill. I figured I could do all this in my sleep. But I wasn't sleeping, was I? I moved into forward zone and smiled as I saw the LED flash, which meant the increment pulse was being applied to the parallel port bit. But wait, there was no beep for the increment of the speed display on the user panel. I physically pressed the increment key, the system reacted with a beep, and then there was an increase in both display speed and tread velocity. Why wasn't my circuitry doing the same thing?

STALEMATE

My goose was about cooked. It was Christmas. I had avoided most of the season's craziness by inventing my own fiasco. It was time to set this all aside and do family stuff. I don't know when it hit me. Perhaps it was during midnight mass on Christmas eve, or while I was cooking Christmas dinner Friday morning, or amidst the frenzy of spending the day with my kids and grandkids. Or maybe it came to me Christmas night while I tossed and turned from over-indulging on good food and drink. Anyhow, what seemed obvious was obviously not. I must have interpreted the keypad incorrectly. I couldn't de-laminate the keypad without ruining something, so I went back to looking at the parallel latch inputs one more time. It turned out it wasn't a simple one-to-one relationship, but many of the keys were actually DPST switches—that is, each button had two switch contacts. One contact was used as a key identifier, like increase or decrease, with a second contact used as a strobe. This way the increase or decrease contacts could be spread across multiple functions with separate strobes used for each function. My increment and decrement inputs needed a third input—the speed function strobe.

Another bit was added to the circuitry and only minor modifications were necessary to the code to add the speed function's control. Another test, and finally success. What started out with complicated interfacing techniques ended up as simple non-invasive automatic control. As I increased my running speed and moved into the forward zone on the treadmill, the project's ultrasonic measuring circuitry indicated this movement to the control algorithm. This in turn requested the treadmill to increase its speed by faking presses of the increase speed button. Once the treadmill's increasing speed overcame my running speed, I returned to the steady zone. When I tired and my speed decreased, I fell back into the rearward zone. This time the algorithm faked a push of the decrease speed button and the treadmill's speed was reduced until my speed was greater than that of the tread and I moved back into the steady zone.

DESIGN COMPLETION

I don't have to tell you about the feeling of euphoria I finally experienced after overcoming this barrage of unexpected disasters. It took only a matter of weeks for me to grow a donkey's tail. So, I apologize for taking you along with me. But you know, one of my idols, Thomas Edison, said, "Genius is 1% inspiration and 99% perspiration." While I don't think his perspiration had anything to do with running on a treadmill, he was not one to give up. Before finding a suitable substance to use as the filament for the light bulb, he allegedly said something along the lines of: "Results! Why, man, I have gotten a lot of results. I know several thousand things that won't work."

I won't attempt to blame this whole mess on the season. Let me just say I'm glad it's now 2010. I want to wish you all the best as we begin a new decade together. Peace out.

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for Circuit Cellar *since 1988. His background includes product design and manufacturing. You can reach him at jeff.bachiochi@imaginethatnow.com or at www.imaginethatnow.com.*

SOURCES

PIC16F1934 Microcontroller
Microchip Technology, Inc. | www.microchip.com

6500 Ranging module
SensComp, Inc. | www.senscomp.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*.

For more need-to-know information about topics covered in Jeff Bachiochi's Issue 238 article, the *Circuit Cellar* editorial staff highly recommends the following content:

—

Pump Control
by Richard Wotiz
***Circuit Cellar* 215, 2008**

This variable-speed drive for an AC induction motor is an excellent device for pump operation. It includes active PFC and inrush current limitings. Topics: Motor, Speed Control, AC Induction, PFC

Go to: www.circuitcellar.com/magazine/215.html

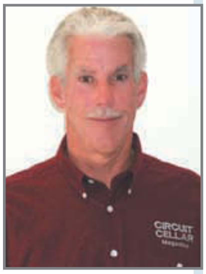
—

Build a Three-Axis CNC Mill Machine
by Gordon Dick
***Circuit Cellar* 201, 2006**

Build a computer-controlled mill machine. The system includes an X-ray machine, motion controller, and a laptop. Topics: Machine Control, Power, Cabling

Go to: www.circuitcellar.com/magazine/201.html

by Tom Cantrell (USA)



IC Therefore IR

You don't have to look farther than your favorite gadget to realize that "touch sensing" is all the rage. Whether it's a cell phone, MP3 player, or even a touchscreen PC, it's easier than ever to let your fingers to do the walking. But now Silicon Laboratories has come up with some clever chips that let your design get up close and personal.

A variety of technologies (inductive, resistive, capacitive, and more) are capable of touch sensing. And today, virtually every major IC supplier offers chips that make it easy to add that feature to your application.

Silicon Laboratories (SiLabs for short) is no exception. Building upon their line of high-performance, mixed-signal '51 MCUs (with roots going back to their acquisition of Cygnal Integrated Products), they've added capacitive touch-sensing to the mix with a combination of chip features and software tools. Good stuff, but not really a new concept since every major MCU supplier has a similar story to tell. However, dig a little deeper into the SiLabs product catalog, and you'll find an interesting alternative—namely, infrared (aka "IR").

SHINE ON

Of course, IR isn't a new idea either. Indeed, I seem to recall a touchscreen CRT solution from way back when that consisted of a plastic frame around the screen with IR LEDs and photodiodes arranged as an XY grid. Put your finger on the screen and you'd block the light, with position being determined by the "shadow" cast on the photodiodes. And let's not forget Steve Ciarcia's own IR forays. Long-time readers may recall

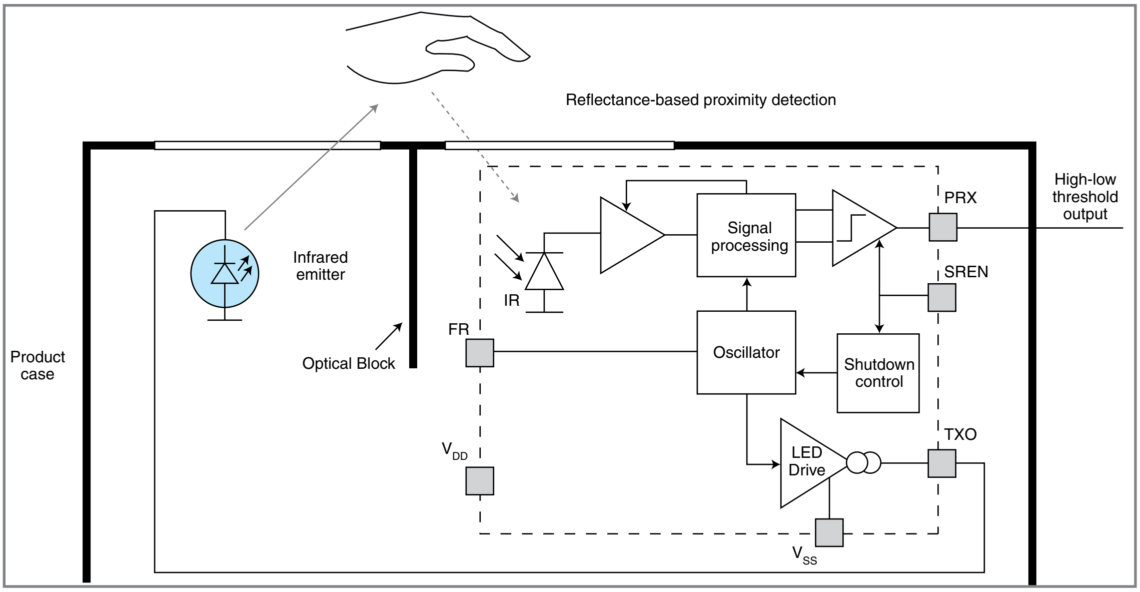


Figure 1—Although based on the same IR premise, the Si1102 (shown here) and Si1120 (see Figure 2) target quite different applications.

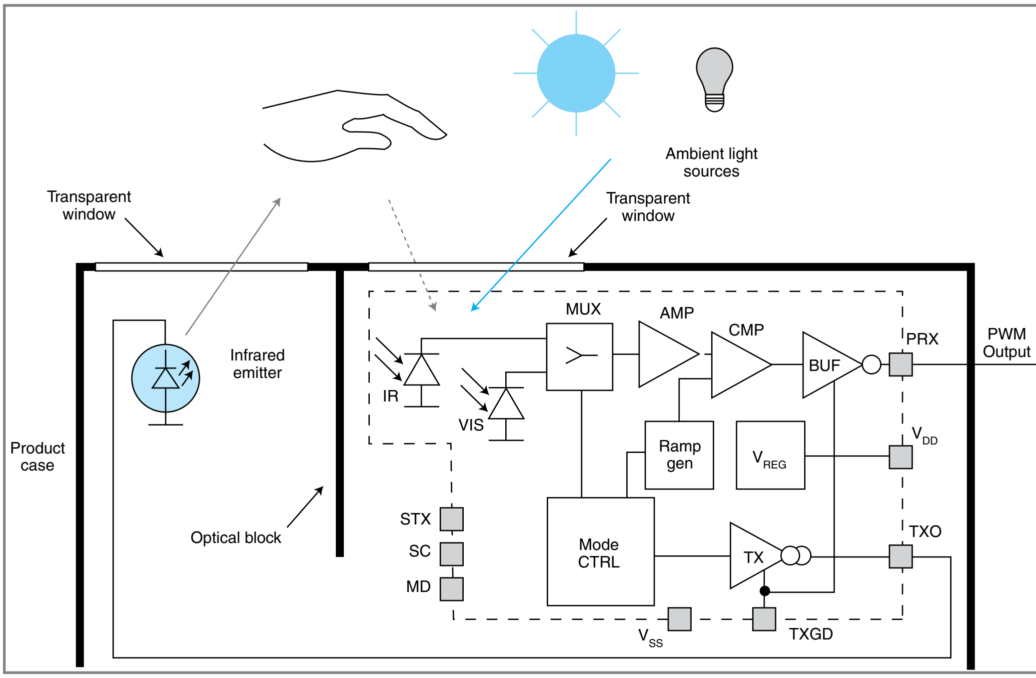


Figure 2—The '1120, in conjunction with an MCU and clever software, is way more versatile than the Si1102.

when he homebrewed an IR scanner comprising a photodiode mounted in a solar cigarette lighter that served as a lens to focus the rays.

While IR isn't new, SiLabs has come up with a clever way of using it to extend the sensory envelope from "touch" to "proximity." The difference may seem small and, in fact, may be just a matter of inches. Nevertheless, "proximity" adds a completely new dimension, literally, that clever designers will surely find creative ways to exploit.

SiLabs offers two parts for your consideration, the Si1102 and Si1120 (see Figure 1 and Figure 2). Both utilize the same strategy (i.e., turn on an IR LED and then measure the IR that is reflected from any nearby objects). The parts look nearly identical encapsulated in their tiny (3 × 3 mm) transparent packages with just eight pins. But beyond these obvious similarities, the '1102 and '1120 are actually quite different under the hood and target rather different application scenarios—although, naturally, there's some overlap. Let's start with the simpler '1102 ($0.90 at 10,000) to get a grasp of the basics and then move on to the fancier '1120 ($1.05 at 10,000).

Photo 1—Lighting your way to new applications is the Si1102 Optical Proximity Detector. The evaluation board combines the '1102 (in the center) with an IR LED (top), a handful of discretes, and a battery (bottom side of board).

The Si1102EB ($39.99) makes it easy to get started since it includes the key components that comprise a minimal system (i.e., an Si1102, IR LED, and power supply). As for the IR LED itself, 850 nm is ideal, but other wavelengths, even visible (red), will work in a pinch, although range and sensitivity will be derated.

Looking at the Si1102EB (see Photo 1), it may appear that the '1102 chip has leads, but it's just an illusion because of the clear package (i.e., the pads are actually underneath the package). Fortunately, the board provides a header for easy access to the otherwise nearly impossible-to-probe chip.

A quick glance at the schematic (see Figure 3) is the best way to understand how the '1102 works. Let's go through the pin connections one by one, starting with power (VDD) and ground (VSS). More on the subject of the power supply in a moment; but for now, let's just say the chip itself runs on anything from 2 to 5.25 V and consumes very little average power. As usual, higher voltages offer some performance advantage (e.g., faster sample rate), but at the cost of higher current consumption.

The IR LED "flashlight" that illuminates the scene (via TXO and TXGD) is the real story when it comes to overall power consumption. According to the datasheet, we're talking about up to 400 mA, which is a really big number that immediately had me questioning the prospects for the Si1102EB battery. But it turns out the LED SiLabs uses is only 45 mW with current limited accordingly.

Furthermore, even with a high-power LED, average power consumption is relatively low because the LED is only illuminated briefly. Furthermore, the LED on time decreases as the reflectance increases beyond the proximity threshold. For example, at a sample rate of 20 Hz with 10% detection ratio, the average power consumed by 400 mA LED would be less than 1 mA.

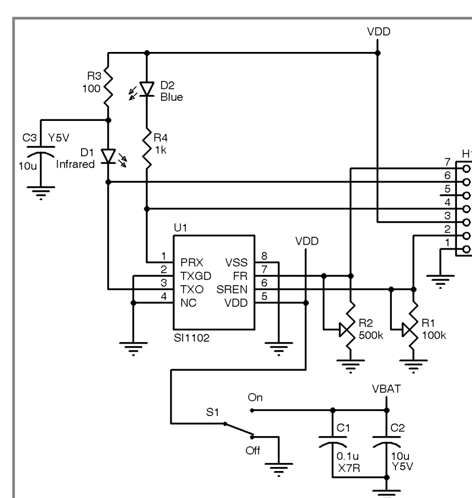


Figure 3—With the '1102, a basic "proximity detector" is as simple as can be. Just add an IR LED and a few discretes.

Average power consumption may not be an issue, but for anything less than a car battery (or big regulator), instantaneously switching a 400-mA load is. Even though the chip limits the TXO slew rate, the current surge will cause a voltage drop on VDD that could glitch the '1102 or other chips connected to the same power supply. For high-power LED applications, a better solution is to power the LED from its own supply (less than 7.2 V) and add a regulator to provide a clean supply to the '1102 and other logic. To that end, the chip is designed such that VDD and TXO can be connected to separate supplies with no power-sequencing or latch-up issues.

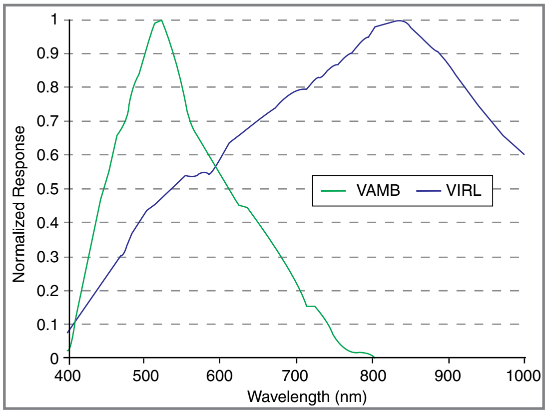


Figure 4—The '1120's dual (IR and visible) photodiodes enable the determination of the "Color Ratio," and thus characterize the source (e.g., incandescent, fluorescent, sunlight) of ambient light.

The sample rate is determined by a resistor (a trimpot in the Si1102EB's case) connected to the aptly named "Frequency Resistor" (FR) pin and ranges between 2 Hz (FR left open) and 250 Hz (FR shorted to ground). Do recall that the upper sample frequency limit is reduced for lower supply voltages—for example, to about 75 Hz at 3.3 V. Finally, the "Sensitivity Resistor/Enable" (SREN) input establishes the threshold for proximity detection with a smaller resistor increasing sensitivity. Although the '1102 is designed for standalone operation, as demonstrated by the EB, an MCU can be used to switch in and out different resistors to dynamically adjust the sample rate and sensitivity.

It all comes together at the "Proximity" (PRX) output, which goes low when the reflected IR exceeds the threshold set with the SREN pin. The PRX output remains latched between samples, but otherwise there is no hysteresis built-in. That means you'll get a lot of jitter on PRX in the detect/no detect transition zone so some external glitch suppression logic might be called for if you're controlling a device directly. On the other hand, an attached MCU could take advantage of the unfiltered PRX output to fine-tune proximity decisions at the margin by averaging the detect/no detect ratio over a number of samples.

As you might imagine, with a fixed threshold (i.e., set by resistor on the SREN pin), the '1102 is best suited to applications where the ambient IR level is somewhat low and doesn't change a lot. Many applications will be well served using a trimpot (instead of a fixed resistor) for SREN to allow fine-tuning for a specific installation.

Indeed, the most troubling potential source of interference is reflections associated with the packaging of the '1102 and LED pair. Unlike the EB board, most real applications don't have the luxury of leaving the components hanging in the breeze. Put them together behind a reflective window, and the '1102 can shoot itself in the foot with optical coupling from the nearby LED swamping the much smaller reflection from a more distant object. Considering the '1102 can detect as little as 1 $\mu W/cm^2$ while the LED next door puts out up to 100 mW/cm^2, you can understand why the pair should be packaged using a so-called "dual-port" configuration that optically isolates them as much as possible. If the '1102 and LED share a single window, it's better to place the devices as wide apart and as close to the window as possible to minimize internal reflection. While you're at it, consider using a window material that filters visible light (sunglasses if you will) since the photodiode sensitivity, though centered on IR, extends a bit into the visible range.

Do some casual experimentation with the EB, as I did, and you'll see it works pretty well in practice. After turning down the sensitivity (i.e., SREN) pot until the blue proximity detect LED went off (i.e., setting the detection threshold just above ambient), I found the '1102 could detect the presence of my hand from about 15 cm. I experimented with various objects to confirm the expected optical response (i.e., range increases for big, bright-colored and shiny objects, decreases for small dark and dull ones). Certain applications may be able to take advantage of this by affixing a highly reflective marker (e.g., shiny tape) to objects of interest.

Mode	Description
PRX400	Proximity detection, 400-mA LED current
OFC	Offset calibration for high sensitivity
PRX50	Proximity detection, 50-mA LED current
PRX50H	Proximity detection, 50-mA LED current, high reflectance
VIRL	Visible and infrared ambient, low range
VAMB	Visible ambient
VIRH	Visible and infrared ambient, high range

Table 1—Seven different '1120 modes of operation give an attached MCU the software hooks to handle a variety of applications.

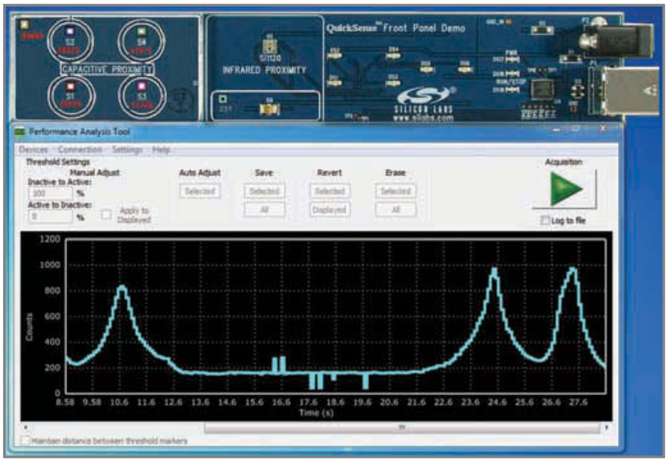


Photo 2—The Front Panel demo board combines capacitive and optical sensing in one design. Here you see the response as I move my hand at the range threshold. Note as well the interference from an IR remote, which is fortunately minor (a few hundred counts compared to full scale of over 12,000).

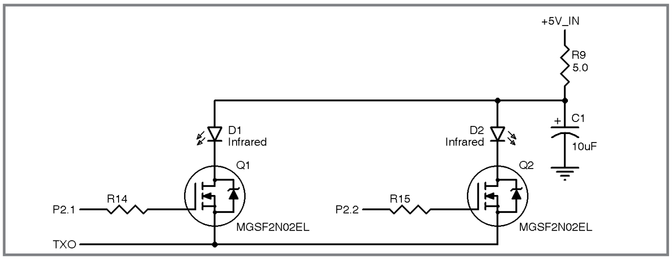


Figure 5—Adding a second LED is a simple matter of sharing the '1120 using a pair of transistors and MCU software.

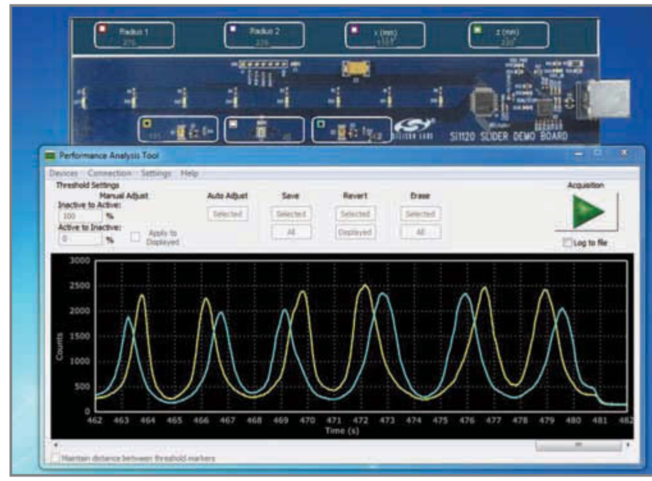


Photo 3—Using two LEDs, as demonstrated with the IR Slider board, enables unique 2-D and 3-D position-sensing applications.

Like using a flashlight during the day, it's no surprise performance is better the lower the surrounding light level. However, the parts are specified to work up to 100 kLux ambient, which is nearly full sunlight and far higher than typical indoor lighting. Just remember that the amount of IR in ambient lighting depends on the source, with "hotter" lights (e.g., incandescent, halogen) emitting more IR than "cooler" (e.g., fluorescent) ones. I had to dial back the '1102 sensitivity (and thus range) to achieve reliable operation in all lighting conditions.

Recalling Steve's IR lash-up, when it comes to boosting range, a lens is the way to go. It does make detection more directional, but that may actually be a good thing for certain applications in order to narrow the field of view to an area of particular interest. Lenses can be used for the LED (i.e., narrow the angle), the '1102 (focus the reflection), or both (for the most gain). An example in SiLabs's "Si1102 and Si1120 Designer's Guide" (a must read) shows how the combination of a small (7-mm) lens for the '1102 and a narrow angle (10°) LED can boost range by a factor of six.

GETTING SMART-IR

Now that you've mastered the basics, the easiest way to explain the '1120 is in terms of how it differs from the '1102. The biggest change is that the '1120 is meant to be combined with an MCU, the pair capable of delivering a smarter solution that goes way beyond the simple "proximity detection" of a standalone '1102. Yes, it might be possible to run the '1120 nearly standalone, but that wouldn't fully utilize its features and at least a little bit of external logic would be required. And these days, a "little bit of logic" usually means an MCU anyway.

The '1120's MCU orientation is reflected in its pin functions. The power, ground, and LED connections (VSS, VDD, TXGD, and TXO) are the same, but the PRX output is now the pulse-width-encoded output from the photodiode rather than the binary (i.e., detect/no detect) output of the '1102. Make that "photodiodes" plural, since the other big difference from the '1102 is that the '1120 supplements the IR sensor with an additional visible light sensor (see Figure 4).

The hardwired (i.e., resistor) control pins of the '1102 are replaced with MCU interface signals on the '1120. Shutdown/Clock (SC) is an Enable signal that both wakes up the part and clocks in a mode selection (MD). Once enabled and with mode set, the strobe input (STX) initiates a light measurement cycle which generates a low-going pulse on the PRX output with the pulse duration (0 to 2 ms) reflecting the light level.

Photo 4—Dual LEDs (IR and visible) enable software to interpret gestures, such as the direction and speed of a "swiping" motion. In addition, they allow an application to automatically adapt to changing ambient lighting conditions.

Each time you wake up the part (i.e., assert SC), it goes through a mode selection cycle (even if the mode you wish to use hasn't changed) and incurs a 500-µs power-up delay. That shouldn't be a problem for "human-interface" response times. However, once the mode is set, as long SC remains asserted, you can take multiple measurements by repeatedly toggling the STX pin. This gives you the opportunity for oversampling (i.e. averaging multiple samples) to improve the signal-to-noise ratio. However, oversampling (unlike using a lens) does increase latency and power consumption.

It's the seven different modes (see Table 1) that set the '1120 apart. The PRX400 mode (i.e., proximity detection, 400-mA LED current) is the one that comes closest to matching the '1102. As you might guess, PRX50 mode works the same, except the LED current is reduced to 50 mA, useful for apps that care more about power consumption than range or sensitivity. The PRX50H mode is a variant designed to allow "single-port" packaging of the '1120 and LED. By reducing the LED power and photodiode sensitivity, PRX50H mode helps prevent internal reflections or leakage from swamping the return. A "single-port" may be the only solution in a retrofit application that doesn't have packaging flexibility. Just remember, you'll always get better performance from a "two-port" (i.e., optically isolated) configuration and should use it if you possibly can. The OFC offset correction mode works just like the other proximity modes, except the IR LED isn't lit. OFC mode can be used to periodically correct for

any power supply or temperature drift, most notably the fact the LED drive current is rather dependent on temperature (0.47% per degree Celsius).

The "V" modes (VIRL, VIRH, and VAMB) are key to the '1120's advanced capabilities, taking advantage of both the visible and IR photodiodes to finely characterize the ambient lighting situation. Indeed, the ability to determine the so-called "color ratio" of IR to visible light even enables you to determined which type of lighting is present (e.g., fluorescent, incandescent, and sunlight).

SiLabs offers a number of options for experimenting with the '1120, starting with the QuickSense Front Panel Demo ($39.99). Combining the '1120 with one of the company's touch-enabled MCUs allows you to compare and contrast IR and capacitive techniques. The kit works with QuickSense Studio, a PC-based tool that facilitates all aspects of software development, combining an extensive API with utilities for configuration, calibration, and analysis.

As I write this, SiLabs has just begun adding '1120 support to QuickSense Studio, starting with the ability to monitor and capture '1120 output on your PC screen via USB. It's not much, but enough to get started and try some experiments. For example, in Photo 2, you can see a fine-looking response as I moved my hand right at the detectable range threshold, a good 1' away (300-plus mm). That's the difference between "touch" and "proximity."

Just for kicks I grabbed the TV remote, aimed it at the '1120, and started punching buttons. The interference was noticeable, but just barely at perhaps a percent or two of full scale. I won't say there isn't any application where interference from IR remotes (or other intentional emitters) might be an issue, but it shouldn't be a concern for most.

There's one other noteworthy difference between the '1120 and '1102. As I mentioned earlier, the LED on-time for the '1102 is inversely related to reflectance (i.e., proximity) once an object is within range. For the '1120, the situation is just the opposite. Due to an internal servo mechanism related to the PRX output, the LED on-time increases with reflectance. If you're simply interested in a "change" or "threshold crossing," software should terminate the cycle (i.e., de-assert STX) as soon as a conclusion is reachable to avoid leaving the LED on longer than necessary.

SEEING DOUBLE

If you really want to have some fun with the '1120, I recommend the IR Slider Demo Kit. It's only a few bucks more ($49.99) and takes the concept to the next level by combining the '1120 with not one, but two, IR LEDs. Adding a second IR LED is a simple hardware upgrade (see Figure 5) that opens the door for way more sophisticated applications. Now we're talking about the ability to establish "position" rather than just "proximity."

Photo 3 shows the two-LED setup in action as I wave my hand left and right. Better yet, notice how the software is able to decode the speed and direction of a "swiping motion" and, taking advantage of the '1120's dual photodiodes, calculate the "color ratio" to identify the type of ambient lighting (see Photo 4).

With the sample rate set to the maximum the software allowed (100 Hz), I was impressed with the responsiveness and ability of the two-LED scheme to accurately detect relatively quick motions of my hand. Whatever overhead (e.g., communication and signal processing) is involved (i.e.,'51 software and PC software), it wasn't so bloaty as to bog things down.

It's interesting to contemplate extending the concept with even more LEDs. Of course, at some point, the task of "visualizing" the environment might be better served with a camera and fancy processor running image-processing software. Anyway, I think even a simple setup (i.e., one or two LEDs) has more than enough intriguing possibilities to inspire the imagination of creative designers.

How about you? Do you see the light?

Tom Cantrell has been working on chip, board, and systems design and marketing for several years. You may reach him by e-mail at tom.cantrell@circuitcellar.com.

REFERENCE

[1] "AN442 - Si1102 and Si1120 Designer's Guide," AN442, 2009, www.silabs.com/Support%20Documents/TechnicalDocs/AN442.pdf.

SOURCE

Si1102 and Si1120 IR Proximity sensors and evaluation kits
Silicon Laboratories, Inc. | www.silabs.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*.

For more need-to-know information about topics covered in Tom Cantrell's Issue 238 article, the *Circuit Cellar* editorial staff highly recommends the following content:

—

Capacitive Sensing Technology
by Jeff Bachiochi
***Circuit Cellar* 182, 2005**
Many exciting products include circular cursor controllers. Jeff details advances in capacitive sensing technology. Topics: Capacitive Sensing, Touch Sensor, Charge Transfer, Electrodes

Go to: www.circuitcellar.com/magazine/182toc.htm

—

ATir Keyboard Interface
by Steven Savage
***Circuit Cellar* 201, 2007**
Steven's ATir design is a cross-platform solution to interface an IR remote control to type keyboard macros to a PC. It includes an IR receiver/demodulator and a few discrete components. Topics: IR, Interface, Remote Control, Communications

Go to: www.circuitcellar.com/magazine/201.html

3 PORT INTERFACE
RS-485 to Ethernet Converter

Only
$170

RS-485 to Ethernet Converter

Powerful feature

- Protocol converter RS485 between Ethernet
- Offer TCP/IP Communication to Devices with RS485 I/F

Specification

Network	: TCP, UDP, DHCP, ICMP, IPv4, ARP, IGMP, PPPoE, Ethernet, Auto MDI/MDIX , 10/100 Base-TX Auto negotiation (Full/half Duplex)
Serial	: RS485 3 Ports, 1,200~115,200 bps, Terminal block I/F Type
Control program	: IP Address & port setting, serial condition configuration, Data transmit Monitoring
Accessory	: Power adapter 9V 1500mA, LAN cable
Etc	: - DIP Switch(485 Baud Rate setting) - LED: Power, Network, 485 Port transmission signal

www.icbank.com

CROSSWORD

Across

3. Coal, petro [two words]
5. An input applied for system/device control
8. Jack Kilby invented the integrated circuit in 1958, but he won the Nobel Prize in 2000 for ___
9. Moves cursor, replaces mouse
10. C/d
13. Antonym: impedance
14. AGP = Accelerated Graphics ___
15. Measures fluid flow velocity; measures airspeed [two words]
16. Slash, /
17. Simulated load
20. Mailing list management program
21. A petabyte = 1 ___ bytes

Down

1. WS_FTP is for?
2. Discovered the piezoelectric effect in 1883
4. The "S" in SATCOM
5. Conductor without insulation
6. 1 billion watt-hours
7. A system or code "fix"
11. Increased power
12. I^2C, SPI, 1-Wire
14. Where's a "track"?
18. Founded the ARRL in 1914
19. 10^{-21}

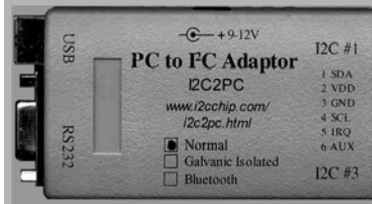
The answers will be available in the next issue and at www.circuitcellar.com/crossword.

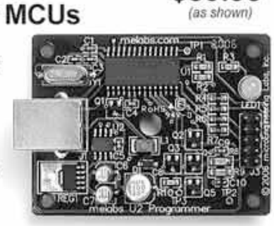
IDEA BOX

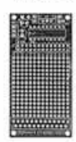
THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/advertise). ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2" x 3" FORMAT. E-mail adcopy@circuitcellar.com with your file or send it to IDEA BOX, Circuit Cellar, PO Box 180, Vernon, CT 06066. For current rates, deadlines, and more information contact Peter Wostrel at 800.454.3741, 978.281.7708 or peter@smmarketing.us.

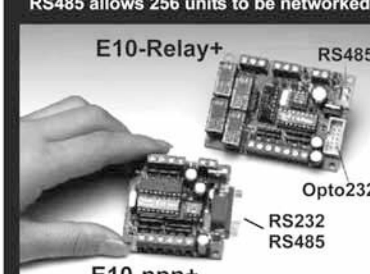
The Vendor Directory at www.circuitcellar.com/vendor/
is your guide to a variety of engineering products and services.


PC to I²C Adaptor
I2C2PC
www.i2cchip.com/i2c2pc.html



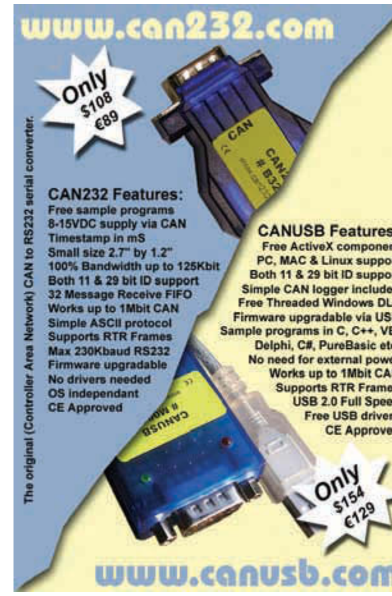


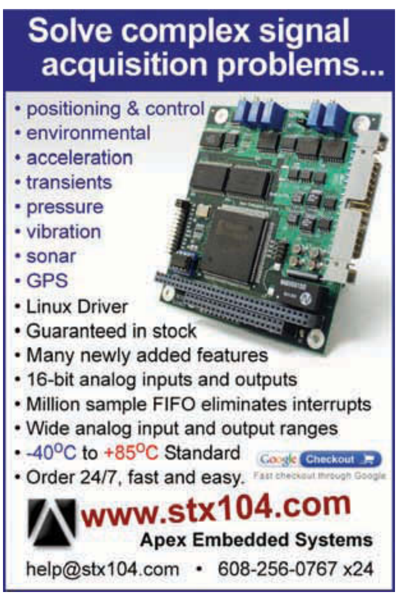

V1.1 melabs.com
N9600 I2C=$50


E10-Relay+
RS485
Opto232
RS232
RS485
E10-npn+


SHORTEN TIME-TO-MARKET
REDUCE DEVELOPMENT COSTS
AVOID DESIGN RISK
ARM9: i.MX27, LPC3250, LPC3180
ARM11: i.MX35, i.MX31
ARM7: LPC2294
XScale: PXA270
Blackfin: ADSP-BF537
Coldfire: MCF5485
x86: Z510, Z520, Z530 (Atom®)
Cortex A8: OMAP3530, AM3517
PowerPC: MPC5554,MPC5567,MPC5200B,MPC565,MPC555
PHYTEC embedded System On Module technology is your production-ready hardware and software solution. Combined with our design services, your new ideas arrive to market in the most timely and cost-efficient manner.
PHYTEC
Visit us online or call today
www.phytec.com | 800.278.9913


KETEREX®
Turn your PC into a SPI, I2C, SMBus, or GPIO Controller
Affordable Test and Measurement Tools
www.keterex.com

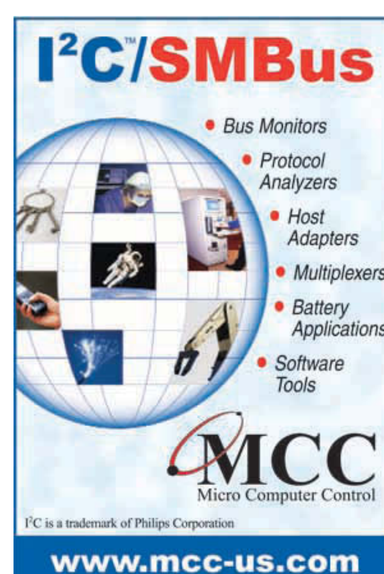

www.can232.com
Only $108 €89
The original (Controller Area Network) CAN to RS232 serial converter.
CAN232 Features:
Free sample programs
8-15VDC supply via CAN
Timestamp in mS
Small size 2.7" by 1.2"
100% Bandwidth up to 125Kbit
Both 11 & 29 bit ID support
32 Message Receive FIFO
Works up to 1Mbit CAN
Simple ASCII protocol
Supports RTR Frames
Max 230Kbaud RS232
Firmware upgradable
No drivers needed
OS independant
CE Approved
CANUSB Features:
Free ActiveX component
PC, MAC & Linux support
Both 11 & 29 bit ID support
Simple CAN logger included
Free Threaded Windows DLL
Firmware upgradable via USB
Sample programs in C, C++, VB, Delphi, C#, PureBasic etc.
No need for external power
Works up to 1Mbit CAN
Supports RTR Frames
USB 2.0 Full Speed
Free USB drivers
CE Approved
Only $154 €129
www.canusb.com


Solve complex signal acquisition problems...
• positioning & control
• environmental
• acceleration
• transients
• pressure
• vibration
• sonar
• GPS
• Linux Driver
• Guaranteed in stock
• Many newly added features
• 16-bit analog inputs and outputs
• Million sample FIFO eliminates interrupts
• Wide analog input and output ranges
• -40°C to +85°C Standard
• Order 24/7, fast and easy.
Google Checkout
www.stx104.com
Apex Embedded Systems
help@stx104.com • 608-256-0767 x24


Disk On Chip
Ready for Delivery
From 16MB to 128MB Available!
Call 530-297-6073 Sales@jkmicro.com
www.jkmicro.com
JK microsystems, Inc.
International Orders Welcome


XL- MaxSonar
Ultrasonic Ranging is EZ
XL-MaxSonar Products
• High acoustic power • Low cost
• Low power, 3V-5.5V, (< 4mA avg.)
• 1 cm resolution • Serial, pulse width, & analog voltage outputs
• Real-time auto calibration with noise rejection • No dead zone
XL-MaxSonar-EZ
• Choice of beam patterns
• Tiny size (<1 cubic inch)
• Light weight (<6 grams)
XL-MaxSonar-WR (IP67)
• Industrial packaging
• Weather resistant
• Standard ¾" fitting
• Quality narrow beam
www.maxbotix.com


Display with Touch Screen
• Embedded, Standalone, C/C++ Programmable...
• Low Cost User Interface with Ethernet, USB, CF, RS232, ADC, DAC, I/Os, Relays...
• U-Drive™ 5.7" QVGA TFT, CAN, Ethernet, CF, ADC, DAC, ...
start at $379 OEM
60+ Low Cost Controllers with TFT, ADC, DAC, UARTs, 300 I/Os, solenoid, relays, Compact-Flash, LCD, Ethernet, USB, motion control. Custom board design. Save time and money.
TERN INC.
1950 5th Street, Davis, CA 95616 USA
Tel: 530-758-0180 • Fax: 530-758-0181 www.tern.com • sales@tern.com


I²C™/SMBus
• Bus Monitors
• Protocol Analyzers
• Host Adapters
• Multiplexers
• Battery Applications
• Software Tools
MCC
Micro Computer Control
I²C is a trademark of Philips Corporation
www.mcc-us.com

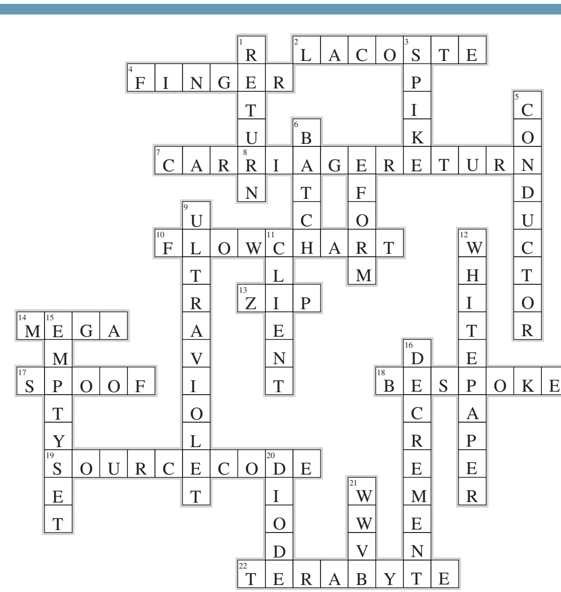
CROSSWORD ANSWERS from Issue 237

Across

2. **LACOSTE**—He built a picoammeter for #237
4. **FINGER**—Used in to find user info in Unix
7. **CARRIAGERETURN**—\r [two words]
10. **FLOWCHART**—Depicts decisions, particularly software-related decisions [two words]
13. **ZIP**—Phil Katz
14. **MEGA**—1 million
17. **SPOOF**—Trick a system
18. **BESPOKE**—Customized
19. **SOURCECODE**—Translated by a compiler into machine language [two words]
22. **TERABYTE**—1,000 GB

Down

1. **RETURN**—A common's path
3. **SPIKE**—Quick surge of volts
5. **CONDUCTOR**—Cu, Ag, Au
6. **BATCH**—Extension .BAT
8. **EFORM**—For placing orders online
9. **ULTRAVIOLET**—100 to 400 nm
11. **CLIENT**—Needs a server's services
12. **WHITEPAPER**—Product info document [two words]
15. **EMPTYSET**—Set of size 0 [two words]
16. **DECREMENT**—Reduce by 1
20. **DIODE**—Conducts one way
21. **WWVB**—NIST radio station



INDEX OF ADVERTISERS

The Index of Advertisers with links to their web sites is located at www.circuitcellar.com under the current issue.

PREVIEW of June Issue 239

Theme: **Communications**

OAE Probe Amp and Intercom (Part 2): Otoacoustic Experiments

Home Automation for an Energy-Efficient House (Part 2): Embedded CANOpen Node Hardware & Software

Custom Interpreter Development: An Innovative LED Movie Project

THE DARKER SIDE **DC/DC Converter Basics**

ABOVE THE GROUND PLANE **Totally Featureless WWVB Clock (Part 3):** Signal Processing

FROM THE BENCH **Application Communication with USB (Part 1):** The Enumeration Process Explained

SILICON UPDATE **Ditch the Switch?**

PRIORITY INTERRUPT

by Steve Ciarcia, Founder and Editorial Director

Electronic Alzheimer's

It certainly isn't a beach in the Bahamas down here at the cottage, but short sleeve shirts and tiki bars forever eradicate unpleasant memories of freezing in Connecticut. I only mention this because one of the other things I did down here this winter was go to an open-air Moody Blues concert. OK, perhaps there were only three people under the age of 50 in the 4,500-plus audience, but it definitely got me thinking about my collection of hundreds of CDs and videotapes and their ages too.

At the post-concert party, I wanted to play one of my Moody Blues video favorites—*A Night at Red Rocks with the Colorado Symphony Orchestra* (Polydor Records, 1993)—until I remembered that the only copies I had of it were on Laser Disk and VHS tape. The Laser Disk player bit the dust a decade ago, and I no longer had a VHS player at the cottage. There's nothing worse than a high-tech guy being subverted by technical progress.

Rather than laud the fact that paper seems to last a thousand years and photographic negatives more than a hundred, perhaps we need to ask ourselves more about how we survive the fact that with each evolution in current technology we seem to have less and less long-term data integrity. ;-) Bits don't have expiration dates, but data memory can only endure forever if the media and file formats remain coherent and discernable. The hardware that breaks down or goes obsolete, the software formats that go extinct, and the online presence (think "cloud") that suddenly vanishes one afternoon is hardly confidence-inspiring.

Presuming you are willing to periodically refresh or convert everything you own to the prevailing format (analog to MP3, HD to Blue-ray, MPEG2 to MPEG4, floppy to flash, etc, etc) and store it on a prevailing compatible medium, you might have a prayer of saving stuff for the long haul. Of course, deciding which medium is a whole 'nother problem.

Opinions vary on the longevity of optical CDs and DVDs. Basically, the bargain optical disks you burn yourself may not last because of the crappy quality of the dye and glue used between the layers. (You find out whether you burned good ones [10 years] or bad ones [two years] the hard way.) Commercial CDs and DVDs are said to last 10 to 25 years, but they may have quality-control issues too.

An alternative to optical storage these days seems to be flash memory. The dollar-to-MB ratio is still very steep, but they do have advantages. Unfortunately, getting reliable statistics or informed opinions about long-term data retention is difficult. I always thought that unpowered flash could sit on the shelf for 10 years and still work, but some suggest it is considerably less. The confusing part of finding the truth is that most flash lifespan statistics are about using smart wear-leveling and powered HDD-like applications, not load-and-unplug-it data storage.

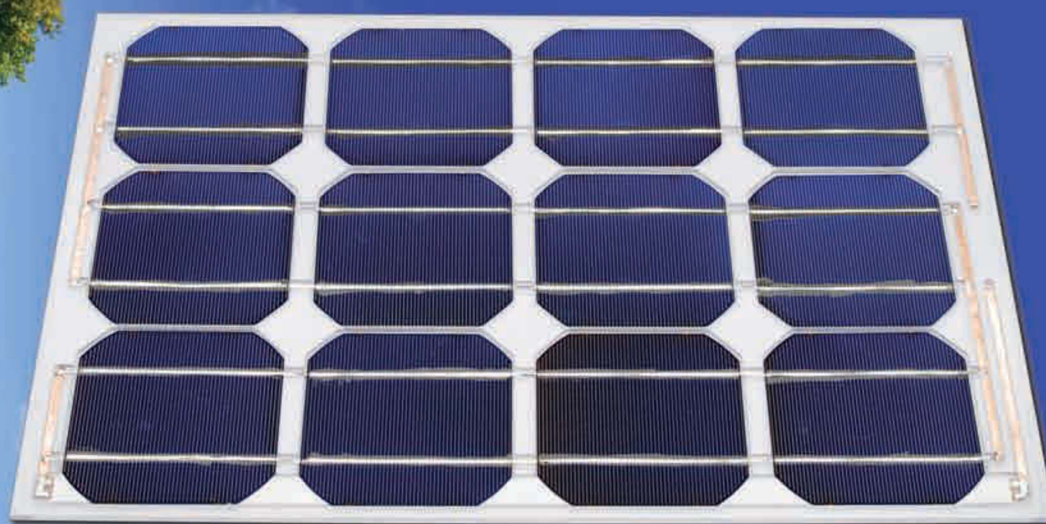
Professionals who know better say that magnetic tape is still the best long-term storage medium, but few of us have tape drives hanging around at home. At $100/TB, the most cost-effective, high-volume magnetic storage medium is still the HDD. Keeping a HDD offline might retain data considerably longer, but powered and spinning is the determining factor for its real lifetime. Unfortunately, even on this statistic, it is hard to get a consistent, let alone straight, answer about lifespan. I read that Google suggested four years, but others have said it is two or less. Compare that with your own experience. I've had hard drives running for 10 years that are still going strong and others that barely lasted one. It's all about the bearings apparently.

So, even if you convert all your files and formats to lessen hardware obsolesce, it's still a Catch-22 on where you save the new data files. Certainly, the promise of the cloud beckons some trusting souls. After all, who doesn't want someone with deeper pockets replacing HDDs every three to four years so we always have reliable data access? The jeopardy here is control. What happens if Google pulls the plug on YouTube or Yahoo shuts off Flickr? Got copies of your videos and photos?

While there are things like 1,000-year carbon nanotube and diamond memories someplace in the development stream, we all have to make relevant data storage decisions today using current technology. Odds are we'll all do it differently because there are so many variables; but, whatever the technique, success will be assured if we "LOC step" together—**L**ots **O**f **C**opies.

steve.ciarcia@circuitcellar.com

30 Watt Solar Panel Kit



Ready to get off the grid? Parallax's 30 Watt Solar Panel Kit is a do-it-yourself system that can produce up to a maximum of 30 watts of clean, green electrical energy. Not a solar demonstration toy! It is a true kit, once assembled you will have a solar panel that produces a substantial amount (6 volts) of electrical power. In fact, the energy produced per square foot is comparable to many commercially available solar panels. Once you permanently seal the panel, it can withstand the outside elements as well. Daisy-chain multiple units together for higher voltage/current/power output! Dimensions: 23.8 x 15.8 x 0.375 in (60.45 x 40.12 x 0.95 cm). *Assemble your panel with confidence! Online demonstration videos show you how to properly handle and solder the delicate cells and their wiring interconnections.*

30 Watt Solar Panel Kit
(#33000; $149.99)

Order the **30 Watt Solar Panel Kit** at www.parallax.com or call our Sales Department toll-free: 888-512-1024 (Mon- Fri, 7am-5pm, PDT).

Friendly microcontrollers, legendary resources.™

PARALLAX INC
www.parallax.com

"ParallaxInc" on Twitter, Facebook, and YouTube