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

CIRCUIT CELLAR

THE MAGAZINE FOR COMPUTER APPLICATIONS

#236 March 2010

ROBOTICS

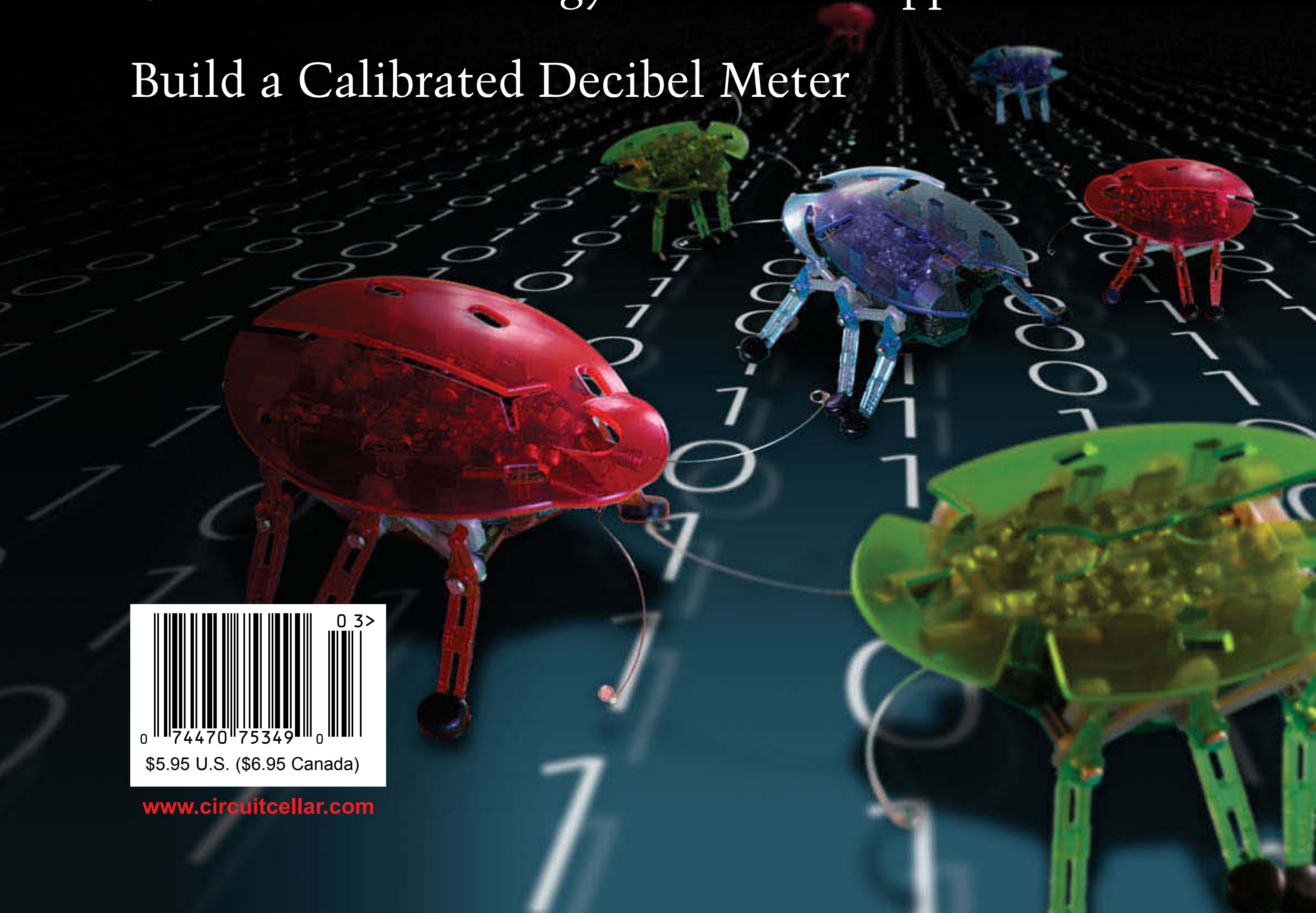
An Environment-Sensing System for Mobile Robots

A Serial Network Hub for an Animatronics System

Directional Light Sensor Design: Phase 2

Use RFID Technology to Control Apps

Build a Calibrated Decibel Meter



$5.95 U.S. ($6.95 Canada)

www.circuitcellar.com

SSH ENCRYPTED SERIAL TO ETHERNET SOLUTIONS

Device P/N: SB70LC-100CR
Kit P/N: NNDK-SB70LC-KIT

$47 *Qty. 1000*

SB70LC

2-port serial-to-Ethernet server

Device P/N: SB700-EX-100CR
Kit P/N: NNDK-SB700EX-KIT

SB700EX

2-port serial-to-Ethernet server
with RS-232 & RS-485/422 support

$129 *Qty. 1000*

Device P/N: CB34-EX-100IR
Kit P/N: NNDK-CB34EX-KIT

$149 *Qty. 1000*

CB34EX

industrial temperature grade
2-port serial-to-Ethernet server
with RS-232 & RS-485/422 support
and terminal block connector

Instantly network-enable any serial device

Works out of the box - no programming is required

Customize to suit any application with low-cost development kit

256-bit encryption protects data from unauthorized monitoring

Features:

10/100 Ethernet

TCP/UDP/SSH/SSL modes

DHCP/Static IP Support

Data rates up to 921.6kbps

Web-based configuration

Need a custom solution?

NetBurner **Serial to Ethernet Development Kits** are available to customize any aspect of operation including web pages, data filtering, or custom network applications. All kits include platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, cables and power supply. The NetBurner Security Suite option includes SSH v1 & v2 support.

NetBurner
Networking in One Day!

Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member

5 Competitive Advantages

Overseas Manufacturing

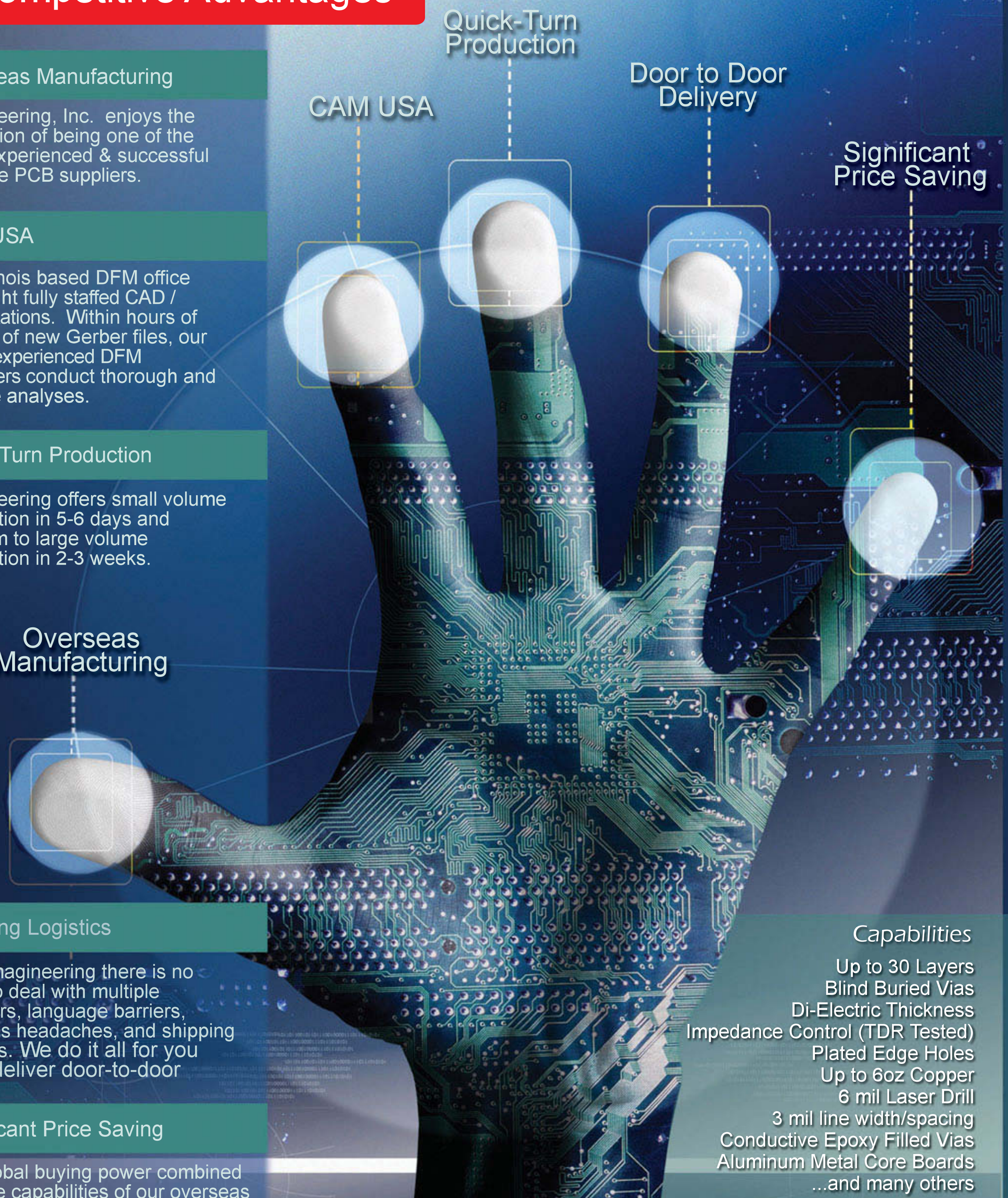
Imagineering, Inc. enjoys the reputation of being one of the most experienced & successful offshore PCB suppliers.

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.

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

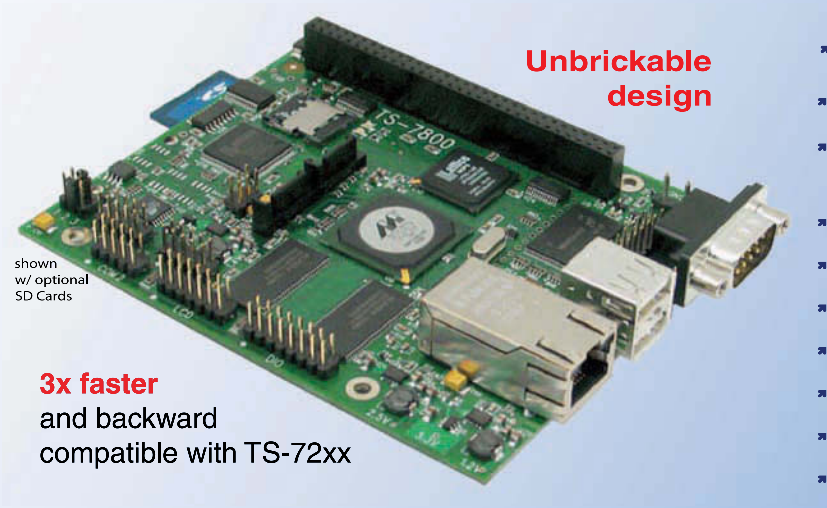
Over the past 5 years, 70,000 prototypes have been successfully delivered from overseas to over 5000 customers

imagineering inc. 847-806-0003 www.PCBnet.com email: sales@PCBnet.com

23 YEARS IN BUSINESS...AND STILL GOING STRONG

Embedded Systems

High-End Performance with Embedded Ruggedness


Unbrickable design
shown w/ optional SD Cards
3x faster and backward compatible with TS-72xx

TS-7800
500 MHz ARM9

$229 qty 100
$269 qty 1

- Low power - 4W@5V
- 128MB DDR RAM
- 512MB high-speed (17MB/sec) onboard Flash
- 12K LUT customizable FPGA
- Internal PCI Bus, PC/104 connector
- 2 host USB 2.0 480 Mbps
- Gigabit ethernet
- 2 SD sockets
- 10 serial ports
- 110 GPIO
- 5 ADC (10-bit)
- 2 SATA ports
- Sleep mode uses 200 microamps
- Boots Linux 2.6 in 0.7 seconds
- Linux 2.6 and Debian by default

TS-TPC-7390
7″ Color Touch Panel Computer

- Low Power, Industrial Quality Design
- Mountable aluminum frame
- 200 MHz ARM9
- 64MB SDRAM (128MB opt)
- 512MB Flash w/ Debian Linux
- 800 x 480 video core
- Dedicated framebuffer- 8MB RAM
- Audio codec with speaker
- Boots Linux 2.6 in less than 2 seconds
- Unbrickable, boots from SD or NAND
- Runs X Windows GUI applications
- Runs Eclipse IDE out-of-the-box

$449 qty 1

More Touch Panel Computers on our website

- Over 25 years in business
- Open Source Vision
- Never discontinued a product
- Engineers on Tech Support
- Custom configurations and designs w/ excellent pricing and turn-around time
- Most products stocked and available for next day shipping

Design your solution with one of our engineers (480) 837-5200

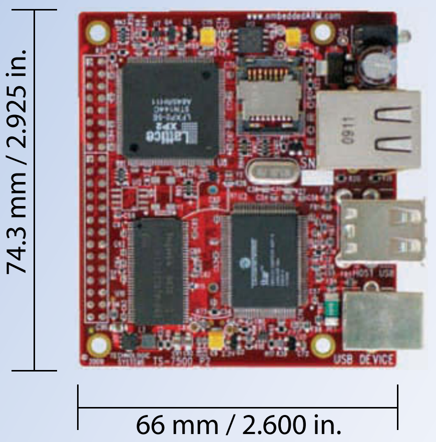
New Products

TS-7500
250 MHz ARM9

NEW!

Our Smallest Computer
at Our Best Price Point

- Low power, fanless, < 2 watts
- 64MB DDR-RAM
- 4MB NOR Flash
- Micro-SD Card slot - SDHC
- USB 2.0 480Mbit/s host (2) slave (1)
- 10/100 Ethernet
- Boots Linux 2.6 in < 3 seconds
- User programmable FPGA - 5K LUT
- Power-over-Ethernet ready
- Optional battery backed RTC
- Watchdog Timer
- 8 TTL UARTs or 3 UARTs & CAN
- 33 DIO, SPI, I^{2}C



$84 qty 100

$99 qty 10

NEW!

modules as low as $89 qty 100

TS-SOCKET Macrocontrollers
Jump Start Your Embedded System Design

Beginning in Q2 2010 Technologic Systems will release a series of Macrocontrollers based on the TS-SOCKET connector standard. These CPU core modules securely connect to your custom base board, enabling drastically reduced design time and complexity. Start your custom embedded system around one of our Macrocontrollers to reduce your overall project risk and accelerate time to market.

- 75mm x 55 mm (credit card sized)
- Dual 100-pin connectors
- Secure connection w/ mounting holes
- Common pin-out interface
- Low profile w/ 6mm spacing

- Simplifies custom embedded systems
- Rapid design process gets products to market faster
- Several COTS base boards for evaluation & development
- Design your own base board or use our design services
- Macrocontrollers are interchangeable for future upgrades

Technologic Systems

We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com

TASK MANAGER

Need-to-Know Info

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

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

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

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

cj@circuitcellar.com [illegible]

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
C. J. Abate

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Jeff Bachiochi
Robert Lacoste
George Martin
Ed Nisley

NEW PRODUCTS EDITOR
John Gorsky

PROJECT EDITORS
Gary Bodley
Ken Davidson
David Tweed

PUBLISHER
Hugo Vanhaecke

ASSOCIATE PUBLISHER
Shannon Barraclough

CUSTOMER SERVICE
Debbie Lavoie

CONTROLLER
Jeff Yanco

ART DIRECTOR
KC Prescott

GRAPHIC DESIGNERS
Grace Chen
Carey Penney

STAFF ENGINEER
John Gorsky

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

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

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

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

PRINTED IN THE UNITED STATES

CONTACTS

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

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

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

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

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

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

Stellaris®

Connected. Versatile. Cost-effective.

Stellaris LM3S9B90 Ethernet Evaluation Kit

featuring USB, Ethernet and CAN

$99

Stellaris LM3S8962 Evaluation Kit

featuring integrated Ethernet and CAN

$89

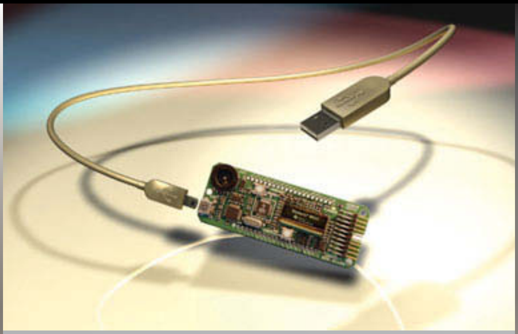
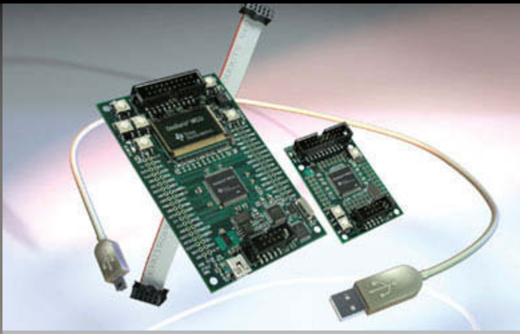
Stellaris LM3S3748 Evaluation Kit

featuring USB Host/Device

$109

Stellaris® Means:

- Deterministic ARM® Cortex™-M3 cores
- Real-time multi-tasking MCUs
- Advanced interrupt-driven software with Stellarisware® software

Stellaris LM3S6965 Ethernet Evaluation Kit

demonstrating an embedded web server out-of-the-box

$69

Stellaris LM3S811 Evaluation Kit

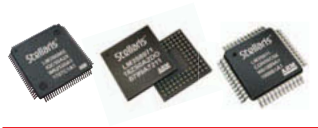
featuring basic embedded control

$49

Stellaris LM3S2965 CAN Evaluation Kit

demonstrating a CAN network out-of-the-box

$79

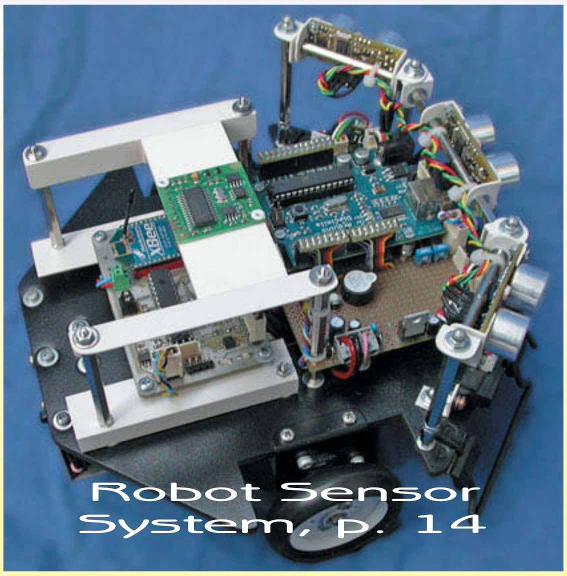


Texas Instruments

INSIDE ISSUE 236

BONUS CONTENT
The Arduino-Based "MiniEric" Robot

March 2010 • Robotics



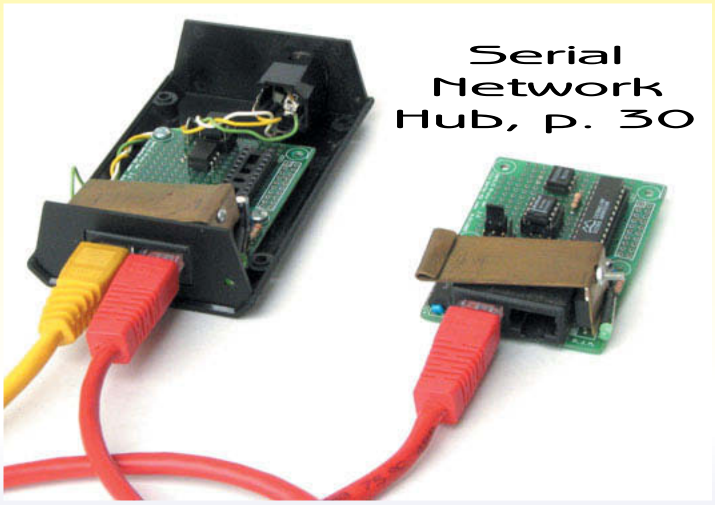
14 **A Sensor System for Robotics Applications**
Guido Ottaviani

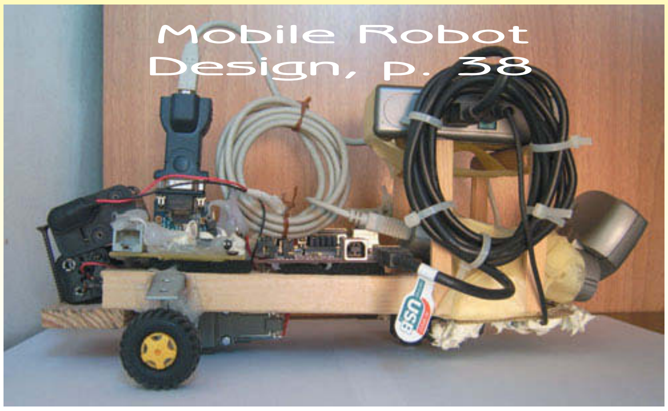
24 **RFID-Based Liquid Control (Part 2)**
Monitoring System Implementation
Brian Millier

30 **Serial Network Hub (Part 1)**
Network Topology and Design Planning
Peter Montgomery

38 **Design and Program a "Minirobot"**
Marco Aiello

46 **Calibrated Decibel Meter Design**
Larry Cicchinelli





TASK MANAGER 4
Need-to-Know Info
C. J. Abate

NEW PRODUCT NEWS 8
edited by *John Gorsky*

TEST YOUR EQ 13

CROSSWORD 74

INDEX OF ADVERTISERS 79
April Preview

PRIORITY INTERRUPT 80
A Matter of History
Steve Ciarcia

54 **FROM THE BENCH**
Sun Tracker (Part 2)
Power Up and Start Tracking
Jeff Bachiochi

62 **LESSONS FROM THE TRENCHES**
Putting C Language to the Test (Part 1)
A Sudoku Puzzle-Solving Program
George Martin

68 **SILICON UPDATE**
Tales from the Crypto
A Look at Embedded Design Security
Tom Cantrell

DESIGN Stellaris® 2010

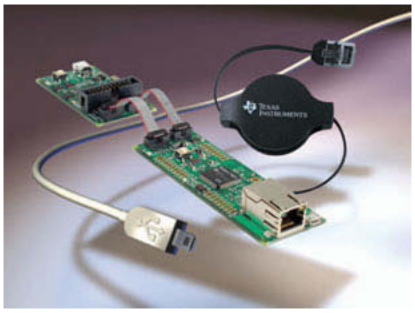
TEXAS INSTRUMENTS

Challenge yourself against other top embedded engineers around the world in DesignStellaris 2010, sponsored by Texas Instruments.

Use the Stellaris® LM3S9B96 microcontroller from Texas Instruments with Keil's RealView® Microcontroller Development Kit (RVMDK) and SafeRTOS™ from Wittenstein to create your design contest entry, and see how far your design will take you!

Stellaris EKK-LM3S9B96 Evaluation Kit free with your contest entry while supplies last!

The EKK-LM3S9B96 Evaluation Kit includes: an evaluation board with an 80 MHz LM3S9B96 MCU featuring Ethernet MAC+PHY, CAN, USB OTG, and SafeRTOS in ROM; a time-limited copy of the Keil RealView Microcontroller Development Kit, cables, documentation, and StellarisWare® software.



- **No purchase necessary to enter.**
- **$10,000 in cash prizes!**
- **Entry deadline is June 23, 2010.**
- **Winners will be announced at the Embedded Systems Conference Boston 2010.**
- **Submit your design today!**

ARM Cortex KEIL WITTENSTEIN CIRCUIT CELLAR

For Complete Details, Visit: www.ti.com/designstellaris2010

TEXAS INSTRUMENTS

CUSTOMIZED LEAD SCREWS AND NUTS FOR PRECISION CONTROL

Haydon Kerk now offers precision lead screws and customized nut designs for motion-control applications where ball screws may have been originally considered. Kerk **lead screws and custom nut designs** create exceptional value for designers and end users in a variety of industries.

For vertical applications, Kerk lead screws can be designed to self-lock and prevent back-driving, unlike traditional ball screws. Utilizing a sliding motion between the nut and the screw to convert rotary input to linear output motion, the Kerk non-ball lead screws offer accuracy comparable to ground ball screws. Kerk lead screws are manufactured with a choice of proprietary coatings, depending on the application requirements. Kerkote and Black Ice dry lubrication coatings eliminate the need for external lubrication and maintenance. Value can also be added to an application by designing and molding customized application-specific nut features. As with all Kerk brand anti-backlash and freewheeling nuts, customized nuts can be molded using Kerkite high-performance polymers or various customer-specified specialty materials.

Manufactured from 303 stainless steel and produced with an exclusive precision rolling process, Kerk lead screws are available in standard diameters from 1/8" (3.2 mm) to 15/16" (23 mm), with standard leads from 0.02" to 3" (0.5 mm to 76 mm). Custom materials, sizes, and leads are available upon request. Please contact Haydon Kerk Motion Solutions directly for pricing.

Haydon Kerk Motion Solutions
www.haydonkerk.com

LOW-COST DEVELOPMENT SYSTEMS FOR MOTOR DESIGNS

The **dsPICDEM MCHV** and **dsPICDEM MCSM** are two new low-cost motor control development systems. The former is for the control of high-voltage motors. The latter is for stepper motors. Along with related application notes and free source code software, these development tools enable rapid designs using dsPIC digital signal controllers.

The dsPICDEM MCHV development system is the industry's only development tool for the rapid evaluation and design of a wide variety of high-voltage, closed-loop motor control applications using AC induction motors, brushless DC motors, or permanent magnet synchronous motors. The board includes in-circuit debugging circuitry, eliminating the need for a separate debugger for development with the dsPIC33 motor control DSC families. Additionally, this tool combines a proven motor control system and power factor correction for regulatory requirements.

The dsPICDEM MCSM Development Board is the industry's most cost-effective tool for creating unipolar and bipolar stepper motor applications. This board enables the rapid development of both open-loop and current-closed-loop microstepping routines using the dsPIC33 motor control families. This development tool also provides engineers with a control GUI, which allows them to focus on integrating the other application features and fine-tuning the motor's operation.

The dsPICDEM MCHV Development System (part # DM330023) is priced at **$650**. The dsPICDEM MCSM Development Board (part # DM330022) costs **$129.99**. It is also available as the dsPICDEM MCSM Development Kit (part # DV330021) for **$269.99**, which includes a stepper motor and 24-V power supply.

Microchip Technology, Inc.
www.microchip.com

BRUSHED DC MOTOR CONTROL MODULE WITH CAN

The **MDL-BDC24** is a brushed DC motor control module with CAN that improves on the highly successful MDL-BDC design by providing variable-speed control for both 12- and 24-V brushed DC motors at up to 40-A continuous current, while adding a new RS-232 serial control input that also functions as a serial-to-CAN bridge. The MDL-BDC24 includes a rich set of sensor interfaces, connectivity, and control options, including analog and quadrature encoder interfaces. The module features highly optimized software and a powerful 32-bit Stellaris microcontroller to implement open-loop voltage control as well as closed-loop control of speed, position, or motor current. The H-bridge on the device is run at high frequency through the MCU's integrated PWM interface and enables DC motors to run smoothly and quietly over a broad speed range.

The MDL-BDC24 is powered by the ARM Cortex-M3-based Stellaris LM3S2616 microcontroller that provides efficient and deterministic performance while integrating CAN, UART, and advanced motion control capabilities. The MDL-BDC24 design also incorporates high-quality analog components, including the SN65HVD1050 CAN transceiver, MAX3221 RS-232 line driver/receiver, TPS54040 swift DC/DC converter, TPS73633 voltage regulator, and INA193 current shunt monitor. The MDL-BDC24 is suitable for a wide variety of consumer and industrial applications, including factory automation devices and systems, mobile robots, household appliances, pumping and ventilation systems, and electric wheelchairs and mobility devices.

The MDL-BDC24 costs about **$102** in single-piece quantities.

Texas Instruments
www.ti.com

TWO-CHANNEL, 10-BIT, 25-MSPS OSCILLOSCOPE

The **PropScope** is a two-channel oscilloscope that is capable of reading 25 million samples per second with 10 bits of resolution over 1-, 2-, 10-, or 20-V peak-to-peak waveforms. Power is provided through the USB port requiring only a single cable to connect the PropScope to any laptop or desktop PC.

A built-in expansion port allows additional capabilities and upgrades by simply plugging in an expansion card. A PropScope DAC card is even included, providing an analog trigger, a 4-bit digital trigger, an 8-bit digital to analog converter, and a 4-bit NTSC/PAL output. Other cards will be available to add even more useful features.

The included software provides a traditional scope interface along with auto measurements and the ability to store and export waveforms. The software also provides features not normally available in a stand-alone oscilloscope, including a function generator, a logic analyzer, a spectrum analyzer, a vector-scope, and more.

The PropScope costs **$249.99**.

Parallax, Inc.
www.parallax.com

STARTER KIT FOR ARM CORTEX-M0

The **IAR KickStart kit for NXP LPC1114** is believed to be the world's first commercial starter kit for ARM Cortex-M0-based microcontrollers. The kit contains all the necessary hardware and software for engineers to quickly design, develop, integrate, and test Cortex-M0 applications.

IAR KickStart Kit includes a development board fitted with the LPC1114 microcontroller, an 8-KB KickStart edition of Embedded Workbench for ARM, and a 20-state evaluation edition of IAR visualSTATE. The board also provides debug support through IAR J-Link-OB, a standard JTAG connector, or a small SWD connector. The board is powered via the USB interface, removing the need for any external power supply. It also provides some user-configurable devices, such as a small LCD, buttons and LEDs, an analog trim wheel, a buzzer, and a prototyping area. In addition, the UART pins are routed to a DE9 connector.

IAR J-Link-OB is a small board mounted JTAG/SWD debug interface that connects via USB to the PC host running Windows. It integrates into IAR Embedded Workbench and is fully plug-and-play compatible.

IAR Embedded Workbench is an IDE with a complete and easy-to-use set of C/C++ cross compiler and debugger tools for professional embedded applications.

IAR visualSTATE is a UML-compliant graphical design environment for reactive systems, with advanced formal verification and validation tools as well as a very powerful code generator.

IAR KickStart Kit for the LPC1114 costs **$139**.

IAR Systems
www.iar.com

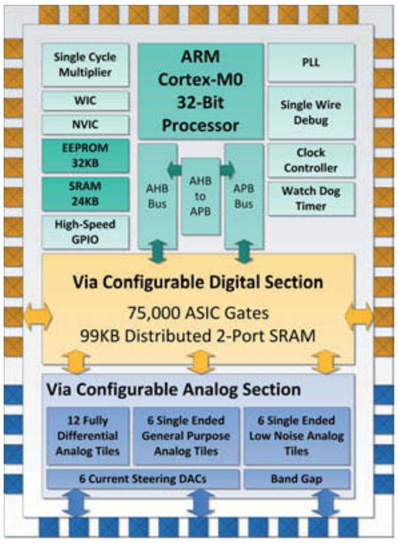
CONFIGURABLE ARM CORTEX-M0 MIXED-SIGNAL ASIC

The **Mocha-1** is the first SoC to integrate the world-class ARM CortexT-M0 with the via-configurable analog and digital functions needed to rapidly and inexpensively deliver embedded mixed-signal solutions. This combination allows electronic system designers to customize analog and digital features with lower power consumption and greater system cost savings than last-generation ASIC solutions. Developers are able to integrate entire existing discrete PC boar d-level designs into the Mocha-1 platform.

Triad's Via-Configurable Analog (VCA) technology combines silicon-pro ven analog, digital, memory, and microprocessor resources on a single ASIC die. These resources are o verlaid with a global routing fabric. Unique to this approach, all of the resources on a VCA are configured by placing vias within the global routing fabric. Vias are placed on the VCA using T riad's advanced, analog-aware, via-only automatic place-and-route software, which configures and interconnects the entire VCA without the need for full-custom manual layout. This approach shortens time to prototypes, shortens time to market, reduces development and tooling costs, and accelerates the design process by enabling the reuse of pro ven mixed-signal IP blocks.

The Mocha-1 integrates a Cortex-M0 processor running at 25 MHz with 32 KB of EEPROM memory, 24 KB of SRAM, 75,000 gates of user configurable logic, and a wide selection of analog resources including op-amps, DACs, ADCs, resistors, capacitors, transistors, and switches. Combining the Cortex-M0 with these configurable analog and digital resources allows Mocha-1 to realize a wide range of single-chip, mixed-signal SoC solutions for industrial, medical, sensor, Bluetooth low energy, and military/aerospace applications.

Mocha-1 provides configurable ASIC solutions unique to each customer. Price is a function of package and production v olume, ranging from **$7** to **$15** per ASIC, with customer-optimized solutions available for under $2 in high-v olume applications. Triad is now accepting customer designs for the Mocha-1 array, with silicon being delivered in Q1.


Single Cycle Multiplier
WIC
NVIC
EEPROM 32KB
SRAM 24KB
High-Speed GPIO
ARM Cortex-M0 32-Bit Processor
AHB Bus
AHB to APB
APB Bus
PLL
Single Wire Debug
Clock Controller
Watch Dog Timer
Via Configurable Digital Section
75,000 ASIC Gates
99KB Distributed 2-Port SRAM
Via Configurable Analog Section
12 Fully Differential Analog Tiles
6 Single Ended General Purpose Analog Tiles
6 Single Ended Low Noise Analog Tiles
6 Current Steering DACs
Band Gap

Triad Semiconductor, Inc.
www.triadsemi.com

Low Cost Panel PC

PPC-E7

- Fanless ARM9 200MHz CPU
- 3 Serial Ports & SPI
- Open Frame Design
- 3 USB 2.0 Host Ports
- 10/100 BaseT Ethernet
- SSC-I2S Audio Interface
- SD/MMC Flash Card Interface
- Battery Backed Real Time Clock
- Up to 64 MB Flash & 128 MB RAM
- Linux with Eclipse IDE or WinCE 6.0
- JTAG for Debuging with Real-Time Trace
- WVGA (800 x 480) Resolution with 2D Accelerated Video
- Four 12-Bit A/Ds, Two 16-Bit & One 32-Bit Timer/Counters

Microsoft Windows CE

2.6 KERNEL

Setting up a Panel PC can be a puzzling experience. However, the PPC-E7 Compact Panel PC comes ready to run with the Operating System installed on Flash Disk. Apply power and watch either the Linux X Windows or the Windows CE User Interface appear on the vivid color LCD. Interact with the PPC-E7 using the responsive integrated touch-screen. Everything works out of the box, allowing you to concentrate on your application, rather than building and configuring device drivers. Just Write-It and Run-It. Starting at $495.

For more info visit: www.emacinc.com/panel_pc/ppc_e7.htm

Since 1985
OVER
24
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • www.emacinc.com

THE ORIGINAL SINCE 1994

PCB-POOL®

Beta LAYOUT

Servicing your complete PCB prototype needs:

- **Low Cost - High Quality** PCB Prototypes
- **Easy Online Ordering**
- **Full DRC included**
- **Lead-times from 8hrs** NEW !
- **FREE laser SMT stencil** NEW !

FREE LASER STENCIL WITH ALL PROTOTYPE PCB-ORDERS

Beta LAYOUT

Email: sales@pcb-pool.com
Toll Free USA: 1877 390 8541
www.pcb-pool.com

Simply send your files & order ONLINE

PIEZOELECTRIC ENERGY-HARVESTING POWER SUPPLY

The **LTC3588-1** is a complete energy-harvesting solution optimized for low energy sources including piezoelectric transducers. The LTC3588-1 integrates a low-loss, full-wave bridge rectifier with a high-efficiency buck converter to harvest ambient vibrational energy via piezoelectric transducers and then convert it to a well-regulated output to power application microcontrollers, sensors, data converters, and wireless transmission components. The device operates from an input voltage range of 2.7 to 20 V, making it ideal for a wide array of piezoelectric transducers, as well other high-output impedance energy sources. Its high-efficiency buck DC/DC converter delivers up to 100 mA of continuous output current or even higher pulsed loads. Its output can be programmed to one of four (1.8, 2.5, 3.3 or 3.6 V) fixed voltages to power a wireless transmitter or sensor.

An ultra-low quiescent current (450-nA) under-voltage lockout mode with a wide hysteresis window enables charge to accumulate on the storage capacitor until the buck converter can efficiently transfer a portion of the stored charge to the output. In its no-load sleep state, the LTC3588-1 regulates the output voltage, consuming only 950 nA of quiescent current while continuously charging the storage capacitor. The LTC3588-1 requires minimal external components and is packaged in either a 3 mm × 3 mm DFN or a thermally enhanced MSOP-10 package, providing a compact solution footprint for a broad range of energy-harvesting applications.

Pricing starts at **$2.95** each for 1,000-piece quantities. Industrial temperature-grade versions, the LTC3588IDD-1 and LTC3588IMSE-1, are also available. Pricing starts at **$3.47** each for 1,000-piece quantities.

Linear Technology Corp.
www.linear.com

NPN

Distributor of Electronic Components
RF Modules
From Part 15 to Part 90 Compliant
Narrow Band FM, UHF Multi-Channel
Data Loggers
Stand Alone and
Wireless Mesh Networking Logger
GSM/GPRS
M2M Solutions
GSM/GPRS modules and modem series
ZigBee Pro
OEM Modules and USB ZigBee Sticks,
Mesh Networks
GPS
OEM Modules and USB ZigBee Sticks,
Mesh Networks
Industrial Bluetooth
OEM, Modules, Wireless Device Servers, RS-232
Long range options, low cost
Wi-Fi
RS232 / 422 / 485 to Wi-Fi Adapter
Connect Data Acquisition Equipment through Serial Port to Wi-Fi network
LEMOS INTERNATIONAL
1.866.345.3667
sales@lemosint.com
www.lemosint.com

DEVELOPMENT SOLUTION FOR ENHANCED MID-RANGE PICs

The **PIC16F1937 Development Kit** combines a powerful optimizing C compiler and ICD-U64 programmer/debugger with a new prototyping board expressly designed to support the development of segmented LCD applications running on Microchip Technology's PIC12F1*xxx*/PIC16F1*xxx* devices.

Segmented LCDs are used in a wide variety of cost-sensitive industrial and consumer products, such as digital thermometers and blood sugar monitors, instrumentation, and measuring devices. The C compiler speeds up the development of that software and contains built-in functions that convert symbols directly into decoded LCD segments. This greatly simplifies the task of displaying text on the screen.

The prototyping board serves as a known-good hardware platform and features a PIC16F1937 MCU with integrated LCD driver and an LCD screen, an RS-232 serial port, push buttons, potentiometers and LEDs, and an expansion I/O port for connection to sensors and other custom circuitry. It's accompanied by complete documentation, including an exercise book with LCD drive examples.

The compiler supports up to 56,000 instructions and 4-KB RAM for these new 14-bit opcode devices. The Enhanced Mid-Range core's new 16-level hardware stack is fully supported by the compiler's call stack, reducing the number of in-lined system functions that must be coded. Built-in system functions are provided for all of the enhanced peripherals, including ADCs and comparators, SPI and I^{2}C serial buses, PWM, mTouch, nonvolatile memory, and LCD controllers.

The PIC16F1937 Development Kit is immediately available in product configurations costing from **$50** to **$459**.

CCS, Inc.
www.ccsinfo.com

NPN

FRONT PANELS & ENCLOSURES

Customized front panels can be easily designed with our free software Front Panel Designer

- Cost-effective prototypes and production runs
- Wide range of materials or customization of provided material
- Automatic price calculation
- Fabrication in 1, 3 or 7 days

Sample price:
$43.78 plus S&H

FRONT PANEL EXPRESS

www.frontpanelexpress.com
(206) 768-0602

AP CIRCUITS
PCB Fabrication Since 1984

As low as...
$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB

Test Your EQ

Edited by David Tweed

Problem 1—*What is the difference between symmetric and asymmetric encryption?*

Problem 2—*Besides sending secret messages, what are some of the other uses for public-key cryptography?*

Problem 3—*Besides a strong cryptographic algorithm, what else is required in order to have a secure communication system?*

Problem 4—*What is a session key?*

Contributed by David Tweed

What's your EQ?—The answers are posted at www.circuitcellar.com/eq/
You may contact the quizmasters at eq@circuitcellar.com

PCBCORE

China PCB Supplier
(Prototype thru Production)

✓ 1-layer up to 30-layer
✓ Cost and quality
✓ On time delivery
✓ Dedicated service
✓ Instant Online Quote & Order
..... Day and Night

No minimum quantity - 1 piece is welcome
Check our low price and save big $$$...

86(571)86795686 sales@pcbcore.com

www.pcbcore.com

Easy Embedded Linux
OmniFlash

We brought you the world's easiest to use DOS controllers and now we've done it again with Linux. The **OmniFlash** controller comes preloaded with Linux and our development kit includes all tools you need to get your project up and running fast.

Out-of-the-box kernel support for USB mass storage and 802.11b wireless, along with a fully integrated Clock/Calendar puts the **OmniFlash** ahead of the competition.

Call 530-297-6073 Email sales@jkmicro.com
www.jkmicro.com

JK microsystems, Inc.
International Orders Welcome
MADE IN USA

FEATURE ARTICLE

by Guido Ottaviani

A Sensor System for Robotics Applications

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

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

ARDU-RINO

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

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

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

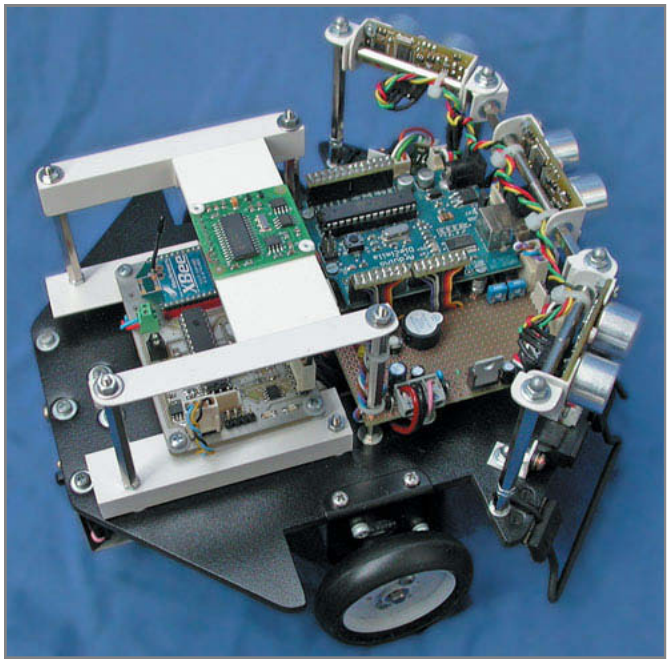


Photo 1—My upgraded robotics platform features a new sensor system that enables obstacle avoidance and location recognition.

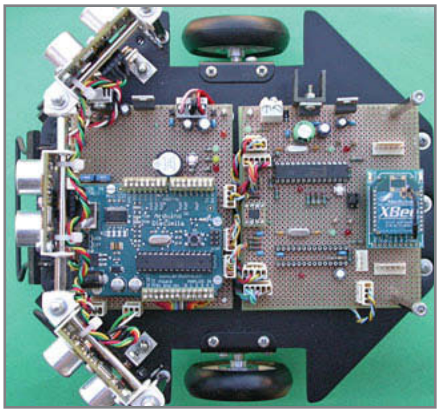


Photo 2—This is my robot's sensor board with most of the necessary devices installed.

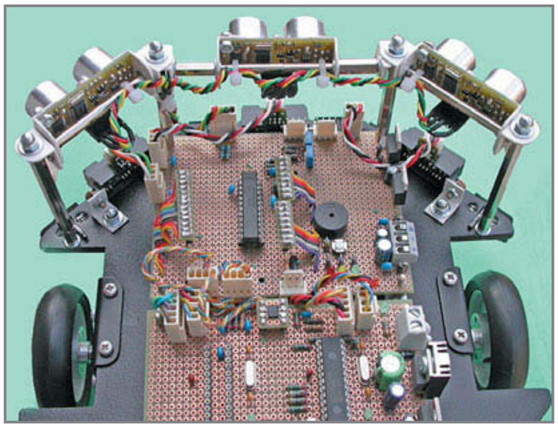


Photo 3—I have my own version of a shield for the Arduino.

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

SMELL

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

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

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

SIGHT

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

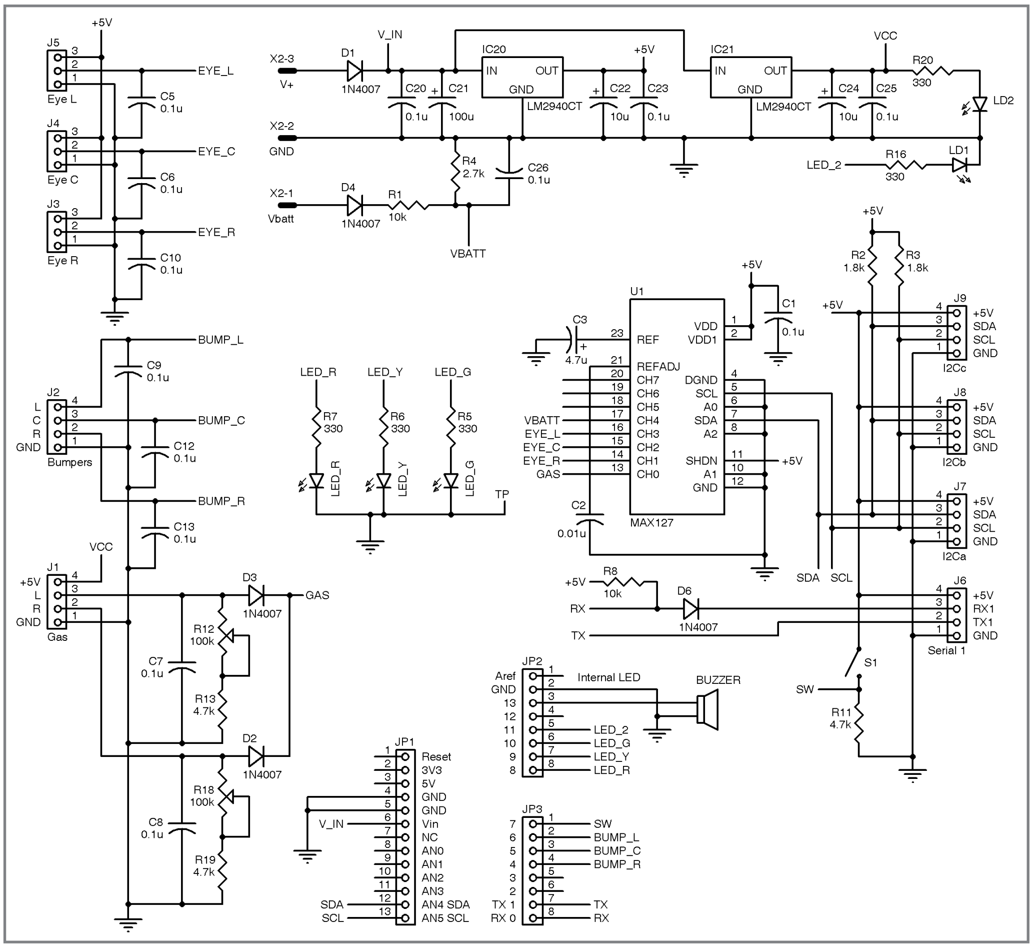


Figure 1—The board is based on an Arduino. Thus, most of the circuits serve to connect and adapt various sensors to the Arduino's ports. The only active component is the MAX127 A/D-to-I ^{2}C expander.

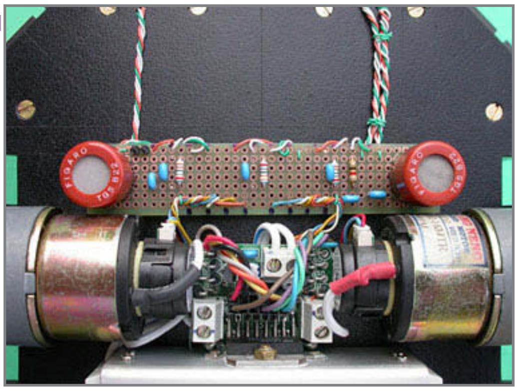


Photo 4—Figaro TGS822 gas sensors are installed on the bottom of the robot.

For infrared sight, I used three Sharp GP2D120 analog distance-measuring sensors (see Photo 6b). This kind of sensor uses an array of receivers to compute the angle of the infrared beam after the reflection on the object. The internal circuit returns a voltage proportional to the distance with a transfer function specified in the datasheet. They are cheaper and smaller than ultrasonic sensors, but the measuring range is limited. For this reason, there are different models for different ranges. The GP2D120 is the best option for this application because it starts from 4 cm, which enables the robot to detect nearby objects. This is fine for mapping the objects with precision, but there is a risk of missing thin obstacles. It works well with mat surfaces. But shiny or glossy objects can be problematic because the IR beam can be reflected away from the sensor. For instance, I have a cylindrical umbrella stand in my house that's similar to a large metallic can. The robot can't avoid it with only the IR sensors. Furthermore, IR sensors can go blind or return false measurements if an intense light (even in visible range) hits them directly. The analog output of the sensors is read through three ports on the MAX127 ADC to I²C chip. I can use the same bus used for SRF08 sensors.

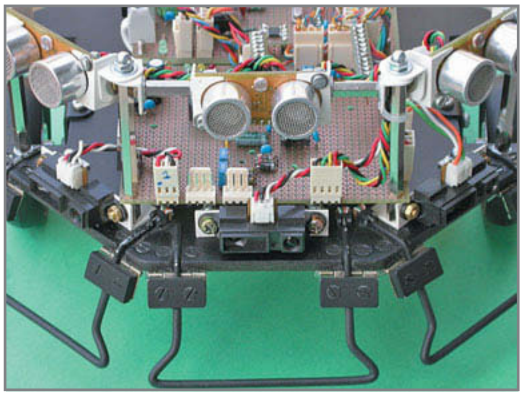


Photo 5—Most of the sensors—range finders, IR distance measuring sensors, mechanical bumpers, and CDS photoresistors—are located in the front. All of them are replicated on three sides with an angle of 45° to expand the system's range of sight.

Ultrasonic sight is what bats use to see. It involves waiting for the echo of ultrasonic waves reflected by objects. My design features three Devantech SRF08 ultrasonic range finders that are driven through the I²C bus (see Photo 6a). With a protocol similar to the one used to read and write a 24CL*xxx* EEPROM, they can be programmed and read.

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

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

Note that a CDS photoresistor is already installed on the SRF08 module. It can measure the ambient light intensity and put the value in a register that's readable through the I²C bus in the same fashion as distance values.

TOUCH

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

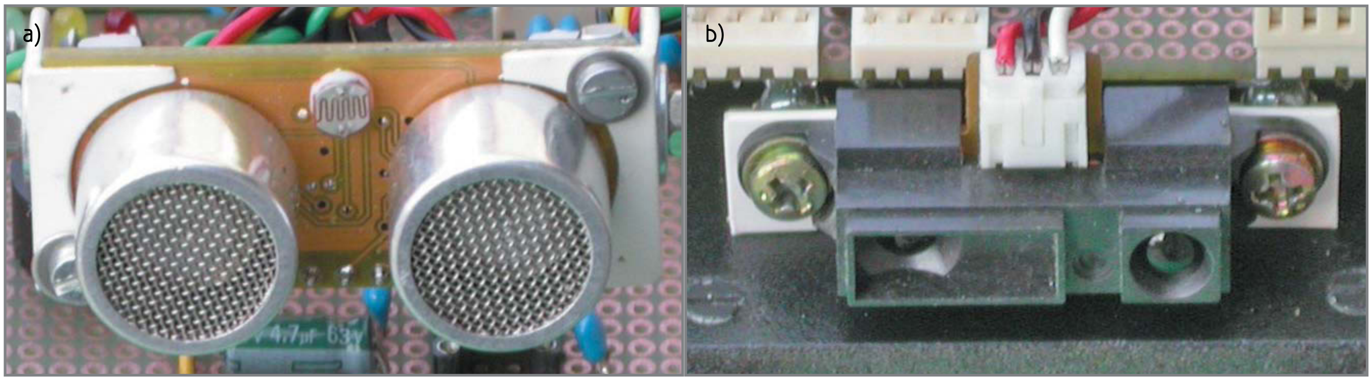


Photo 6a—The SRF08 module contains a complete circuit that handles the procedures for measuring distances between 3-cm and 6-m with ultrasonic waves. b—The GP2D120 infrared module can accurately measure objects from 4 to 30 cm.

Introducing the new, improved CUWIN

CE FC

CUWIN4300A

800 x 480 resolution, 260K colors
RS232 x 2 / RS485 x 1 or RS232 x 3
Mono Speaker and Stereo jack
Real time clock (Battery backup)
USB I/F (ActiveSync)
Keyboard or Mouse support
ARM9 32bit 266MHz processor
Windows CE 5.0
64MB FLASH, 64MB SDRAM

$499 / Qty. 1

CUWIN3200A

800 x 480 resolution, 260K colors
RS232 x 2 / RS485 x 1 or RS232 x 3
Mono Speaker and Stereo jack
Real time clock (Battery backup)
USB I/F (ActiveSync)
Keyboard or Mouse support
ARM9 32bit 266MHz processor
Windows CE 5.0
64MB FLASH, 64MB SDRAM

$399 / Qty. 1

CUWIN3500A

800 x 480 resolution, 260K colors
RS232 x 2 / RS485 x 1 or RS232 x 3
Mono Speaker and Stereo jack
Real time clock (Battery backup)
USB I/F (ActiveSync)
Keyboard or Mouse support
ARM9 32bit 266MHz processor
Windows CE 5.0
64MB FLASH, 64MB SDRAM

$429 / Qty. 1

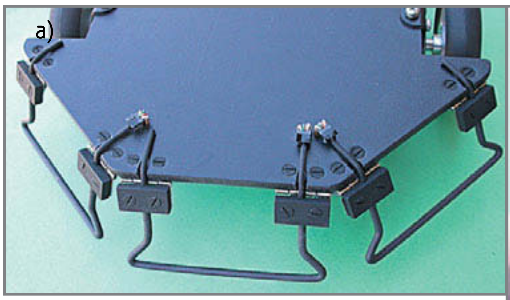
* Waterproof Front Panel

- ✓ Touch Panel
- ✓ Color LCD Display
- ✓ Ethernet / RS232 / RS485
- ✓ Works with MS Visual Studio

COMFILE TECHNOLOGY

1175 Chess Dr., Suite F, FOSTER CITY, CA 94404
call : 1-888-9CUBLOC (toll free)
1-888-928-2562
email : sales@comfiletech.com

www.comfiletech.com



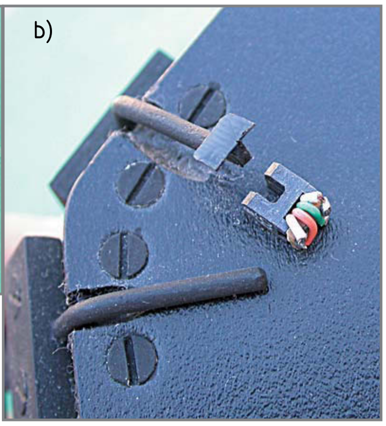


Photo 7a—The bumpers on the front of the robot can detect small and low objects that the other sensors can't pick up. b—This fork-shaped photo-interrupter can detect the bumper's movement.

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

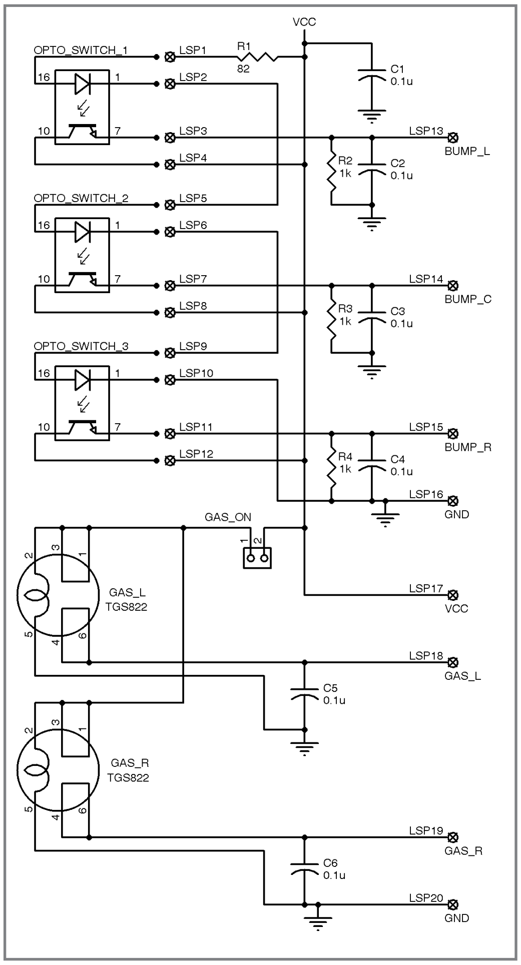


Figure 2—The lower board includes the optical switches for the bumpers and some components for signal conditioning.

SOUND

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

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

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

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

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

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

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

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

WIZnet 2010 iMCU DESIGN CONTEST

CIRCUIT CELLAR® THE MAGAZINE FOR COMPUTER APPLICATIONS

Awaken the Wonderful Wizard in YOU!

NOV. 2009 ~ JUN. 2010

Your creativity and design project idea could win you a share of $15,000 in Cash Prizes and Recognition in Circuit Cellar magazine.

For details, visit www.WIZwiki.net

FUTURE ELECTRONICS

iMCU7100EVB Contest Special Price: 49.00 USD

Co-sponsor Official Sample Purchase: www.FutureElectronics.com

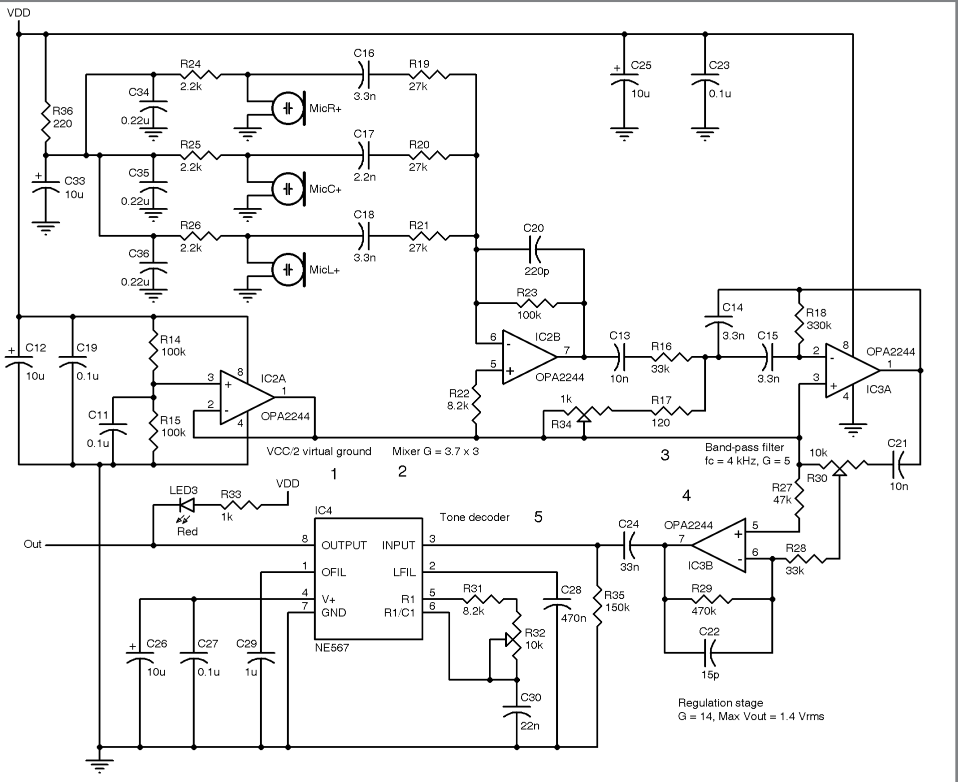


Figure 3—This three-chip board, used to detect the 4-kHz tone generated by the sound target, could be logically divided into five sections, each one with its precise purpose in the decoding process (as noted by the numbers in this figure and detailed in the text). To increase this circuit's performance,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

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

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

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

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

First low-cost mixed signal oscilloscope!

- Oscilloscope, Spectrum Analyzer, Recorder
- Logic Analyzer, Pattern Generator
- 2 + 5 Analog Channels (12 bit, 1 MS/s),
- 16 Digital Channels, Square and PWM Generator
- Up to 128 Simultaneous DAQ Devices
- Free Software Upgrades

PoScope mega1

www.poscope.com

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

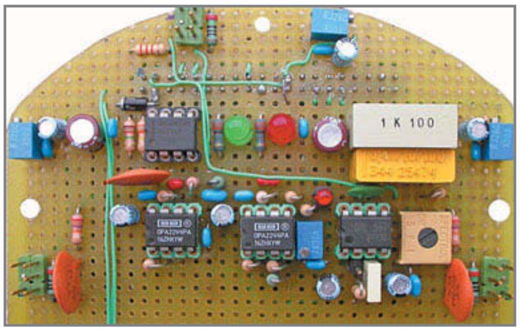


Photo 8—This is a prototype of the sound board on perfboard.

DIRECTION

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

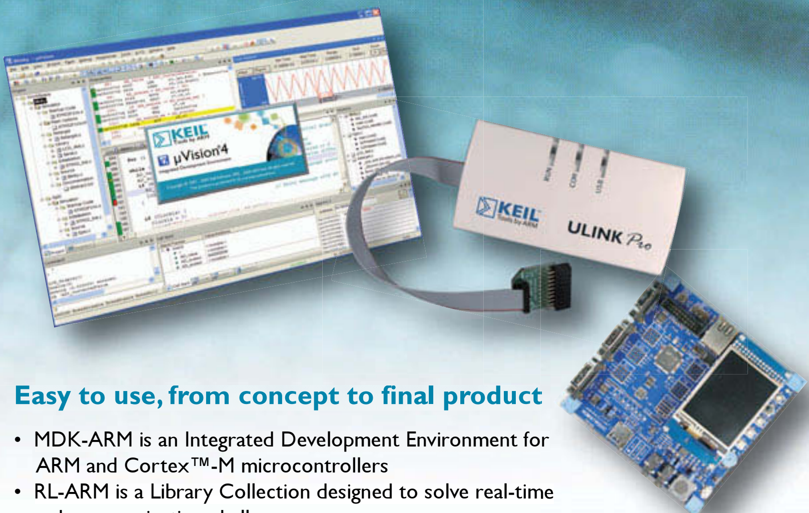
To increase dead-reckoning precision, many kinds of "sensor fusion" methods are experimented with. Gyrodometry is a sophisticated method.[3] This uses a combination of sensors that requires some computation capabilities for both the microcontroller and the programmer who writes the code.

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

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

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

Leading Embedded Development Tools...



Easy to use, from concept to final product

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

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

KEIL™ Tools by ARM

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

SOFTWARE & SETTINGS

The software runs on an Arduino Diecimila board. All the sensors except the bumpers are interfaced on the I²C bus directly or through a MAX127 converter. The battery voltage is monitored with an A/D port to determine when a power shortage is coming. Thanks to the Wire libraries, the software is very simple, and it's just a matter of cyclically polling each sensor with the right timing.

Because the SRF08 sensors come with the same address, the first setting must assign a different I²C address to each one. It is needed just once to change the default address E0. Thus, you must have only one sonar on the bus at a time. When powering up the SRF08 without sending any commands, it will flash its address on the LED—one long flash followed by a number of shorter flashes, indicating its address from 0 to 15. The flashing is terminated immediately by sending a command.

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

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

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

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

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

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

CYCLING

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

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

READING & TRANSMISSION

The SRF08 sensors can return the distance already converted in centimeters or inches. While reading the distance value, the light values from SRF08 modules are also read. This can be done within a single I²C reading cycle on registers 1 and 2. Sharp IR sensors instead return an analog value inversely proportional to the distance of the object. According to the documentation, the response curve can be approximated with a 1/x trend. A simple conversion can be obtained with the following formula:

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

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

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

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

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

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

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

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

DESIGN SUCCESS

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

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

PROJECT FILES

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

REFERENCES

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

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

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

RESOURCES

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

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

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

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

SOURCES

Diecimila board
Arduino | www.arduino.cc

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

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

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

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

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

by Brian Millier

RFID-Based Liquid Control (Part 2)

Monitoring System Implementation

This article series details how to build an RFID-based controller for monitoring dispensed liquid nitrogen from a tank in a laboratory setting. Here you learn about the circuitry, command structure, and system operation.

In the first part of this article series, I described my RFID-based transaction system design. I work in a science department at Dalhousie University in Halifax, Canada, so this was a custom design (as are most of my designs). Basically, I chose inexpensive off-the-shelf components from the small quantities at my disposal. In addition to designing the system, my goal was to take the data collected by the system and transfer it to the accounting staff in a convenient way: as a file on a USB flash drive.

I'll begin this article by explaining how I worked with an FTDI Vinculum VDRIVE2 module's command structure. I'll then provide circuitry details and describe the system from the operator's point of view (see Photo 1). Thanks for "tuning in" again.

VDRIVE2 TO MCU INTERFACE

The Atmel ATmega32 MCU that I chose for this project contains only one UART port, and both the RFID module and VDRIVE2 require a UART port. At first, it seemed I was coming up a little short. The BASCOM-AVR Basic compiler that I used supported bit-banged virtual serial ports, but I was afraid they might not be up to the task in this case. Since the RFID reader and the VDRIVE2 modules are never used simultaneously, I used a simple CMOS multiplex chip to share the ATmega32's single UART port with both devices. I had ATmega32s on hand for this project. But if I were starting from scratch, I'd choose the newer Atmel ATmega324 that has two UART ports.

The VDRIVE2 can also communicate with a microcontroller via a SPI port. To do this, you have to change a jumper on the back of the module, as described in the module's datasheet. For some reason, FTDI chose to implement a nonstandard SPI protocol. By this I don't mean the four different SPI modes that arise from the various clock phase/level conventions used by the various manufacturers of SPI devices. In this case, FTDI chose to implement a SPI that might best be described as a pseudo-I^{2}C model. It's a 13-bit protocol with start, R/*W, address, and status bits in addition to the 8 data bits. I decided that it would be

Photo 1—This is the liquid nitrogen generator in its own small room. W e don't dispense the liquid nitrogen directly from the blue tank, but rather transfer it to other dewars for actual dispensing outside of the room. The gray panel on the wall to the right is a unit I built that weighs these dewars in conjunction with a load cell and chain hoist visible at the left of the picture.

a hassle to write custom driver routines to suit this protocol, so I stuck with the standard UART interface.

While the VNC1L chip contains two DMA controllers and 4 KB of SRAM for incoming and outgoing USB packet buffering, this still provides only a finite amount of buffering for the USB datastream. For that reason, when communicating with the VNC1L via the UART port, you must monitor the *RTS handshake output before sending data to the device and drop the *CTS input low to allow the VNC1L to transmit data. I used a buffered, interrupt-driven, serial input routine. But the largest buffer that I can implement is 256 bytes, so I must use the *CTS line to throttle back the incoming VDRIVE2 data when I'm reading in the RFID information file. This data then must be stored in EEPROM memory, and that takes some time.

The VDRIVE2 module reports its presence *very shortly* after power is applied by sending a sign-on message. The VDRIVE2 module itself has no accessible Reset line, so you can't hold it in Reset until your MCU is ready to accept serial input. Therefore, I initialize the interrupt-driven serial I/O routines right at the start of my firmware, so that the ATmega32 is ready to receive serial data well before the VDRIVE2 reports its presence. It is just as easy to leave the *CTS line high at start-up and only assert it when the ATmega32 was ready to accept serial input.

In this project, I send the VDRIVE2's sign-on message out to the controller's LCD for diagnostic purposes. If no sign-on message is received, it means the VDRIVE2 module is bad. In this case, the LCD leaves a "Testing Flash Drive Interface" message on the screen indefinitely.

After the VDRIVE2 sends its sign-on message (its firmware revision number), it checks its USB port for the presence of a flash drive. If it finds one, it reports the following:

```
Device Detected P1
No Upgrade
D:\>
```

The final line is the old DOS prompt, which is familiar to us "old-timers" who grew up with PC DOS.

The "No Upgrade" message bears explanation. The VDRIVE2 module is capable of updating its own firmware automatically via an update file located on the flash drive. This file, which would normally be supplied by FTDI, has to be loaded onto the flash drive using the filename "FTRFB.FTD." Generally, such a file would not be present and the "No Upgrade" message would be reported (as I've shown).

In this project, I use the presence of a flash drive, as reported by the aforementioned prompt, to switch the program into one of two modes. One is the default mode: No Flash Drive Present-Liquid Nitrogen Dispensing Mode. The other is Flash Drive Present-Maintenance mode. This mode enables you to set the date/time, download the RFID tag file, upload the monthly figures, and zero the totals and erase transactions.

Up to this point, we have not sent any commands to the VDRIVE2. Before proceeding, it is important to note that the VDRIVE2 can interact with its host microcontroller in two modes: Extended and Short. The former uses short mnemonic commands, most of which are two or three characters long. These generally mimic the old DOS commands that they emulate (e.g., DIR for a directory list). The latter mode uses a single-character mnemonic for each command. The Extended mode is handy for troubleshooting when using terminal emulation software on the PC, while the Short mode is faster and more efficient when using a microcontroller.

Frankly, I found the Vinculum VDRIVE2 firmware documentation (V 1.06)—which was available when I first developed the project—to be rather cryptic in places. It also contained a few critical errors that slowed me down somewhat. I spent quite a while figuring out how the VDRIVE2 worked by using a terminal emulator program on my PC. When I subsequently wrote the microcontroller firmware, I decided to continue using the same Extended command set, which I had used with the terminal emulator, even though it was a little bit less efficient.

Later, while using a VDRIVE2 in a subsequent project, I rechecked the Vinculum website to see if the VDRIVE2 firmware documentation had been updated. I found that the documentation was indeed much more detailed, but it also appeared that the firmware present in the newer VDRIVE2 modules was a bit different from what was present in the module I used when building the project. Rather than pointing out problems I had with the older documentation, I'll concentrate on the latest firmware as described in FTDI's "Vinculum Firmware User Manual" (V2.3).

In addition to the aforementioned two command modes, the VDRIVE2 firmware also handles two different data modes: ASCII and Binary. This affects the way numeric parameters are passed to and returned from the VDRIVE2. At power-up, the VDRIVE2 defaults to Binary mode in which all numbers are expressed using a fixed number of bytes, depending on the range of numbers needed by that particular command. What must be noted (and what wasn't clear in the original documentation) is that numbers should be sent to the VDRIVE2 MSB first, whereas numbers being returned are sent LSB first. In ASCII mode, numbers can be entered as either decimal, or as hex numbers preceded by either of the following: "$" or "0x." All numbers are returned as hexadecimal numbers and contain a space prior to the trailing <CR> terminator. Table 1 contains the four commands that are used to switch between the various command and numeric modes I described.

In the case of the two Command Set commands, either the short or the extended versions are recognized,

Extended command Set	Short command set (hex codes)	Function
SCS <CR>	10 0D	Switch to Short command set
ECS <CR>	11 0D	Switch to Extended command set
IPA <CR>	90 0D	Express parameters in ASCII
IPH	91 0D	Express parameters in binary

Table 1—These are the commands needed to switch amongst the various VDRIVE2 Command/Numeric Modes.

regardless of the mode in which the device is currently running. I find it strange that Vinculum decided to have the VDRIVE2 boot up with both the Extended Command set active (good for a terminal emulator) and the Binary numeric mode active (awkward for a terminal emulator).

With the aforesaid mode selections out of the way, you can start issuing commands to the VDRIVE2. One of the first commands needed is the DIR command, which lists all the files present on the flash drive in the current directory. More useful, for this project, is the "DIR filename" command, as it returns an indication of the presence of that file on the flash drive, as well as its length in bytes. When reading a file, it's important to know the file's length in advance. That is the only way you'll know when you've reached the end of the file (no EOF indication is given by the VDRIVE2 firmware).

While the older documentation claimed that the size was returned as "size in hex(4 bytes)," the size is returned as follows in reality:

ASCII mode:

```
Filename_$xx _$xx_ $xx_ $xx_<CR>
```

Binary mode:

```
Filename_cccc <CR>
```

The _ symbol represents a space. `cccc` is a 4-byte binary number (long integer) representing the file length. `$xx` is a two-character hexadecimal number

Once you determine the file size, it's relatively easy to read it by using the `RD filename` command, followed by a serial input loop set up to read the correct number of bytes contained in the file. Although you don't have to close a file opened for Read (using the `CLF` command), FTDI recommends that you do so for the sake of future compatibility.

Writing a file to the VDRIVE2 is also pretty easy, but you must know the size of the file you're writing in advance. In this project, only two files that are written to the flash drive: a Totals file and a Transactions file. In

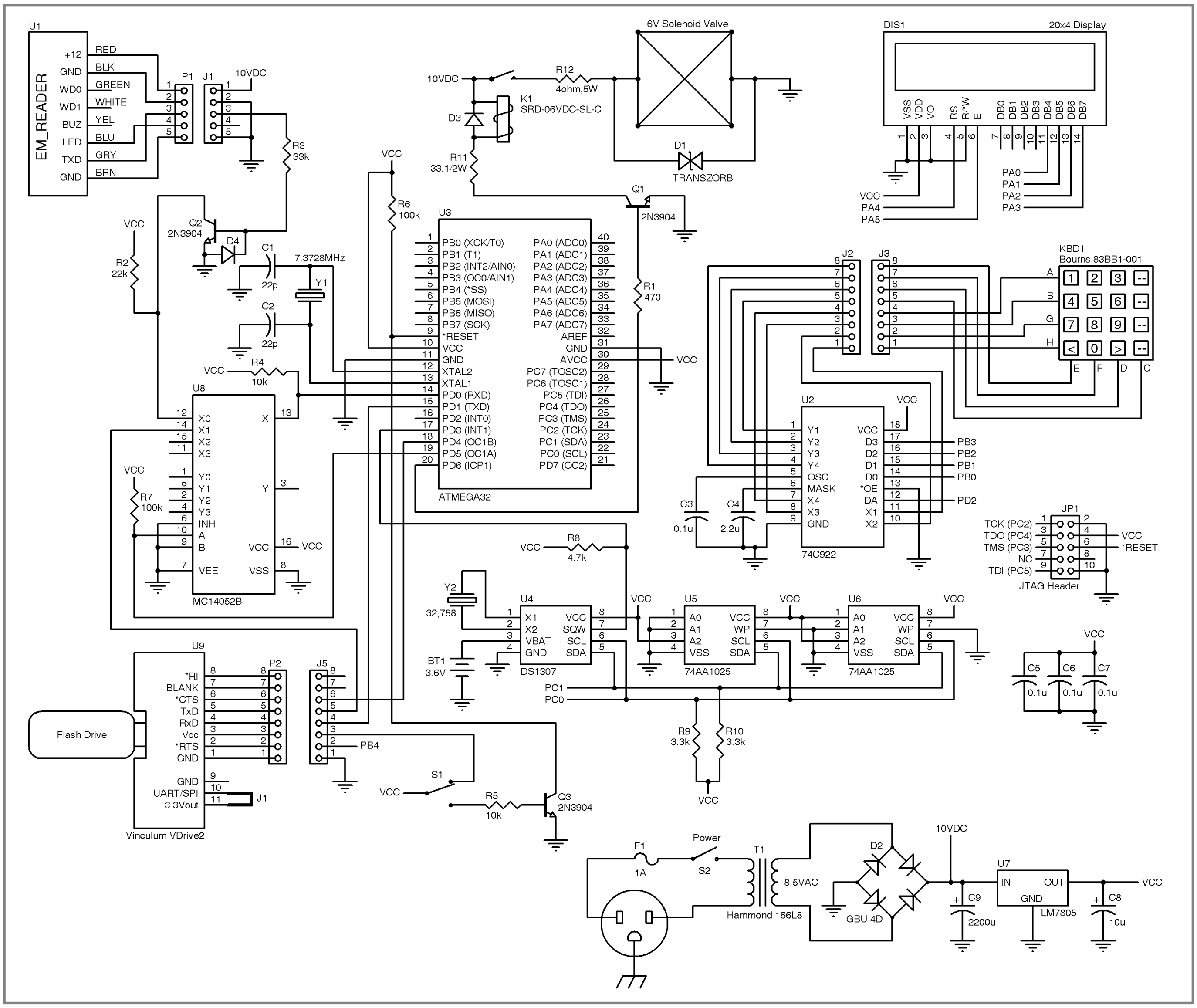


Figure 1—Take a look at the circuitry. A better match for the MCU would be an ATmega324, with its two UART ports. It would eliminate the MC14052 MUX chip.

both cases, these are tables with records having fixed length fields. Therefore, you can easily calculate the file size in advance. Writing these files involves sending an `OPW filename` command to open the file, followed by a `WRF command` containing the file size expressed as a 4-byte-long integer. Then the data is sent out 1 byte at a time, all the while checking the *RTS line to ensure that the VDRIVE2 module is ready to accept the data. After all the data has been sent, a `CLF filename` command is sent to close this file. I issued the `DLF filename` command prior to writing a file to delete any existing file of the same name.

The VDRIVE2 firmware also contains commands to create and navigate up and down a folder hierarchy—much like DOS. It also contains utility commands `FS` and `FSE` to report the amount of space remaining on the flash drive. These commands can take up to a minute to execute on a large, heavily populated flash drive. The VDRIVE2 module I purchased had early firmware that did a free sector count whenever a flash drive was inserted. Thus, there was a bit of a delay when a drive was inserted, which would take an extended amount of time on a heavily populated flash drive. I believe this step has been eliminated in the newer firmware: now you can invoke the `FS` command (or `FSE` for flash drives greater than 4 GB) when necessary before writing new files. For this project, only three files are ever on the 64-MB flash drive I use, so such checks are quite short.

CIRCUITRY

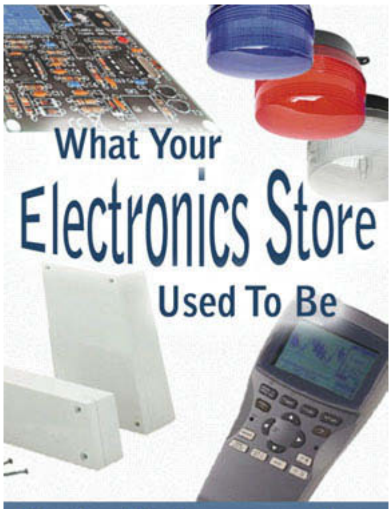
As you can see in Figure 1, a transformer/bridge rectifier supplies about 10 V unregulated to the unit. The raw 10 V is used to directly power the RFID module, which is specified to run on 9 to 12 VDC. Similarly, the valve—which requires 6 V at about 1 A—is also supplied by the raw 10-V supply through a dropping resistor. A 7805 regulator supplies the 5 V needed for everything else.

The RFID module is designed to be wall-mounted with a cable connecting it to a controller located somewhere else. It transmits an RS-232-level signal, and I convert this to the TTL-level signal needed for the ATmega32, using Q2 and associated passive components. An MC14052 CMOS multiplexer chip switches the ATmega32's RxD input line between the RFID module's output and the VDRIVE2 module's output.

The VDRIVE2 module requires an input signal and is directly connected to the ATmega32's TxD line. The VDRIVE2 *RTS status output is connected to the ATmega32's PortB4, and its *CTS line is driven by PortD4.

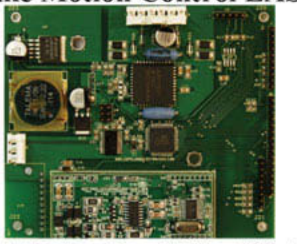
The ATmega32's hardware I^{2}C port is used to interface both the Maxim Integrated Products (Dallas) DS1307 real-time clock chip and the two Microchip Technology 74AA1025 64-KB flash memory devices. Like most I^{2}C devices, the 74AA1025 memory chips contain three address lines, which, in most devices, allows up to eight of the same devices to be daisy-chained together on one bus. I thought I was enough of an I^{2}C "expert" to know how to connect these lines without actually checking the datasheet closely, and I grounded the A2 lines on both chips. However, in the case of the 74AA1025 devices, you must tie the A2 line high for the device to work properly. Strangely enough, the first half of the memory space of each device worked fine, even with my wiring error, but I was only able to get the second half to work after I examined the datasheet closely and discovered my mistake! I'm sure Microchip has a good reason for going this route, but it seems to fly in the face of the standard I^{2}C addressing scheme. The DS1307 RTC circuit uses a standard 32.767-kHz watch crystal and a CR2032 3-V lithium battery for backup when the main power is shut off.

When I was building the unit, I wasn't sure what the program code was going to look like, so I decided to play it safe and interface the keypad to the ATmega32 using a 74C922 keypad encoder chip. This chip performs the actual keyboard scan and raises its DA line when a key is pressed. The DA line is fed to an interrupt line on the ATmega32. Keyboard input is handled by an interrupt service routine (ISR).



DesignNotes.com, Inc.
Receive a FREE Gift with orders over $100.00!
1-800-957-6867 www.DesignNotes.com

Make Motion Control EASY



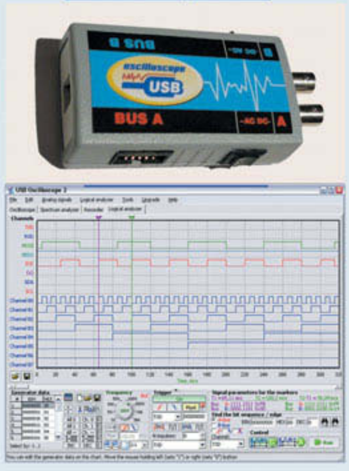
- Unipolar/bipolar Stepper Driver
- Single 7.0V to 45VDC Input
- Supports up to 2.5A per phase
- FreeRun mode for flexible control with a joystick
- Full, 1/2, 1/4 and 1/8 step modes supported
- Home and limit switch inputs
- User programmable IO pins
- RS-232 interface (USB avail)
- Extensive command set
- USB, Ethernet, WiFi and other interfaces available via CSI interface.
- Small Footprint 4.3" x 4.7"
- Optional LCD/Keypad

ALSO AVAILABLE
- Single board computers
- CNC design and manufacturing
- Contract assembly and MORE...

www.copelandelectronics.com

Copeland Electronics, Inc.
(614) 475-1690

USB Oscilloscope for $169.50
Logic and Spectrum Analyzers, Generator.
www.HobbyLab.us



Function	Description
Download totals	Transfer the total dispensed for each RFID tag to USB flash drive
Zero all totals	Clear the totals for each RFID tag in internal flash memory
Upload ID tag file	Transfer a table of valid RFID tags/account names to internal flash memory from the USB drive
Set time/sate	Load the internal RTC with the correct time/date
Valve timing	Enter the two parameters related to valve timing

Table 2—These functions are available in Maintenance mode.

This makes the program firmware simpler, and makes it easier to implement timeouts for various user input routines. It was, admittedly, a bit of laziness on my part. The ATmega32 has enough free I/O lines to handle the keyboard's 4 × 4 matrix, as well as plenty of free program memory to hold a keyboard scan routine.

I wanted the unit be user-friendly, so I used a 4 × 20 LCD to enable the complete spelling of messages. No cryptic messages here. The LCD is interfaced to the ATmega32 with the classic 4-bit data bus configuration, along with E and RS control signals. All six LCD signals connect to the ATmega32's PortA.

I used a small 6-V PCB relay to control power to the solenoid valve. The solenoid valve draws about 1 A, and even though I used a 6-V surge suppressor across it, I still felt better isolating this inductive load from the microcontroller circuitry with a relay. A 4-Ω, 5-W resistor drops the raw 10-V power supply down to the 6 V necessary to activate the valve. Similarly, the 6-V relay coil is powered by the raw 10-V power supply through a resistor.

During development, it was useful to have a manual reset switch to aid in testing. To make this useful, I also wanted to cycle the power to the VDRIVE2 module because the VDRIVE2 does not have its own reset input. Therefore, the reset circuit consists of switch (S1). One position of S1 supplies power to the VDRIVE2. The other position provides, in conjunction with Q3, an active-low RESET signal to the ATmega32.

The unit's speed is predominantly controlled by the data transfer speed of both the VDRIVE2 and the 74AA1025 flash memory chips. So, I decided to operate the ATmega32 at the relatively low clock rate of 7.3728 MHz—a clock rate that also produced a perfectly timed data rate of 9,600 bps.

The ATmega32 contains JTAG circuitry, which allows for active debugging using Atmel's AVR Studio software. Therefore, I included the standard Atmel JTAG socket in this project, which came in handy while debugging the firmware. The JTAG interface uses four PORTC lines. Although it isn't shown in the diagram, I also included the standard Atmel SPI programming interface in case I would have to reprogram it later with a conventional SPI AVR programmer.

OPERATION

Let's look at how the unit operates from the user's point of view. You see the controller operating in Dispensing mode. In this mode, the current time/date is displayed on the LCD, as well as the first user prompt. When an RFID tag is brought within a few inches of the RFID reader, the module sends the tag ID to the microcontroller's RxD line via the MC14052 multiplexer. The microcontroller then scans its table of valid ID code entries to find a match. If a match is found, it displays your account name on the LCD. It prompts you for the amount to dispense, which you enter on the keypad. When you confirm that the amount is correct, the microcontroller updates two tables in its onboard E^{2}PROM flash memory. One table simply keeps running totals of the amount dispensed for each RFID tag. Another table stores a transaction record consisting of an ID number, date, time, and amount dispensed for auditing purposes.

The valve is then opened for a timed interval to dispense the requested amount of liquid nitrogen. There is not an exact mathematical relationship between the amount of dispensed liquid nitrogen and the time that the valve is open. Therefore, the controller opens the valve for an amount of time greater than needed in worse-case conditions. I depend on the customer (and the honor system) to control the amount dispensed. In other words, a customer could take somewhat more than what he or she enters into the controller. However, the chemistry department was depending entirely on the honor system before when customers would simply write down what they took on a sign-up sheet. In that scenario, they could take nitrogen and not sign up at all if they were dishonest.

The other mode of operation is Maintenance mode. The controller enters this mode when it is powered and finds a USB flash drive inserted into the VDRIVE socket. When it does, it checks for the presence of a password file (pwd.txt). If it finds this file on the flash drive, it compares the value of the string found in it to a string constant called "pwd," which is embedded in the program code. This is the first constant defined in the source code listing. If a match is found, the program enters Maintenance mode; otherwise, it just displays a password error message and goes into an infinite loop. Maintenance mode includes the five options shown in Table 2. Switching between the two modes is done only at power-up, at which time the system senses the presence or absence of a USB flash drive in the VDRIVE2 socket and jumps to the appropriate routine for that mode.

WRAP UP

The "ready-to-run" RFID module I obtained from Futurlec sped up the design process for this project. Due to its low price, it didn't make sense for me to try and roll my own using an RFID ASIC chip, antenna, and other parts. I was also assured that the low-cost tags would work with that reader, which might not have been the case had I designed my own using one of the RFID ASIC chips made by several domestic IC manufacturers. Similarly, the time I spent learning how to use the Vinculum VDRIVE2 module was modest.

I'll likely incorporate this module in future projects. There is no doubt that low-cost USB flash drives provide a convenient way to transfer any amount of data, large or small, between any project containing an embedded controller and a PC. ■

Author's Note: I'd like to express kudos to Mark Albert who wrote the excellent Bascom-AVR Basic compiler, which makes Atmel's AVR family of microcontrollers such a pleasure to use.

Brian Millier (brian.millier@dal.ca) is an instrumentation engineer in the Department of Chemistry at Dalhousie University in Halifax, Canada. He also runs Computer Interface Consultants.

PROJECT FILES

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

RESOURCE

FTDI, "Vinculum Firmware User Manual," Ver. 2.3, 2007.

SOURCES

ATmega32 Microcontroller
Atmel Corp. | www.atmel.com

VDRIVE2 Interface module
Future Technology Devices International | www.ftdichip.com

DS1307 Real-time clock
Maxim Integrated Products, Inc. | www.maxim-ic.com

Bascom AVR Basic compiler
MCS Electronics | www.mcselec.com

74AA1025 64-KB Flash memory device
Microchip Technology, Inc. | www.microchip.com

MC14052 Multiplexer chip
ON Semiconductor | www.onsemi.com

EM RFID Card reader
Shanghai Huayuan Electronic Co. Futurlec (distributor) | www.futurlec.com

SEALEVEL®

Computing/HMI | Serial | I/O

The right connection creates incredible power.

Connectivity and control. Making your interface as reliable as the tides—and just as strong. Sealevel creates hardware and software solutions for both digital and serial interface requirements.
We Listen. Think. And Create.

Experience exceptional computing at low power with the SBC-R9, an application-ready platform for your next product design.

sealevel.com > sales@sealevel.com > 864.843.4343

by Peter Montgomery

Serial Network Hub (Part 1)

Network Topology and Design Planning

If you're building a custom multipart animatronics or robotics system, consider adding a serial network hub to run the show. Here you learn why building a six-port RS-485 hub for a custom network packet system can simplify node wiring and more.

What can I say? I really like Halloween. I also enjoy designing hardware and writing software. What's a geek to do? Combine both interests into a fully automated Halloween display system each year, of course!

This is the third article I've written about the technology behind my ever-evolving Halloween display (www.socalhalloween.com). What began in 1995 as a couple of Styrofoam tombstones, a pumpkin, and a sheet shaped like a ghost has mutated into a large-scale yard display featuring custom animatronics, smoke machines, lights, six channels of digital audio, and a dedicated RS-485 control network to run it all. And the latest addition? An RS-485 hub designed to handle my custom RS-485 serial network packets (see Photo 1).

The basic design philosophy behind my control system is to have programs running on a laptop inside my house that talk to and control hardware nodes located outside. An example of a hardware node is shown in Photo 2. My generic I/O node can control or poll anything that can be interfaced to its I/O bits. I created a data packet format for my system to allow me to create software and hardware that can communicate without writing custom protocols for each new device. My Halloween network uses standard CAT-5 Ethernet cables to carry both full-duplex RS-485 communications as well as 12 V of DC power. This lets me plug hardware nodes onto cables and provide both data and power with a single cable. I chose CAT-5 cables because they are affordable, readily available, and provide four twisted pairs of conductors in them.

The RS-485 network turned out to be a solid design choice. Through the use of conventional serial communications and an inexpensive level converter IC, I have a robust network that works flawlessly. All the dedicated hardware can interact, and the laptop controls everything.

RS-485 DETAILS

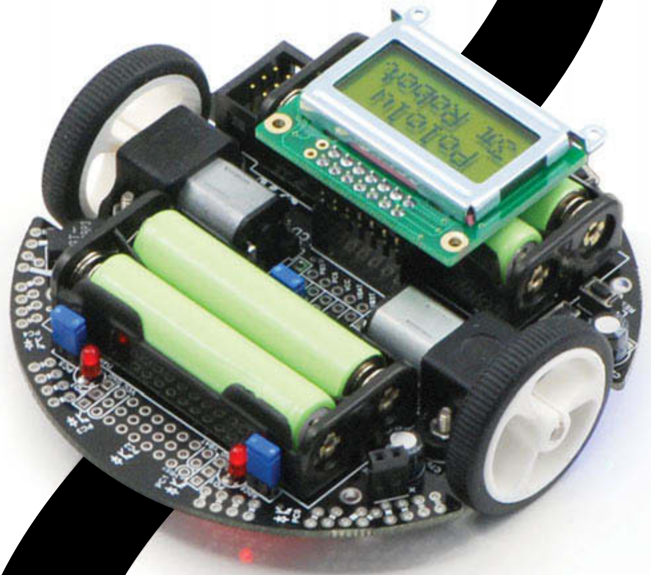
If you're familiar with RS-485, you're probably thinking, "Why does he need a hub for RS-485?" If you're unfamiliar, let me explain a bit about why everyone else is perplexed.

RS-485 is an electrical spec that sends data using a differential balanced line over twisted pair wiring. Simply put, the data is sent via two transmitters simultaneously, but

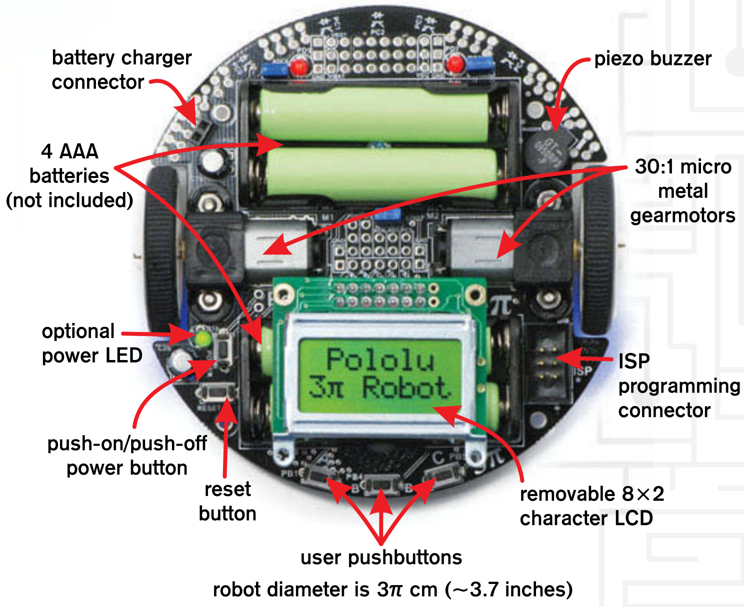
Photo 1—This is the finished network hub design.

3pi Robot

The Pololu 3pi robot is a complete, high-performance mobile platform featuring two micro metal gearmotors, five reflectance sensors, an 8×2 character LCD, a buzzer, and three user pushbuttons, all connected to a C-programmable ATmega328P microcontroller. Capable of speeds exceeding 3 feet per second, 3pi is a great first robot for ambitious beginners and a perfect second robot for those looking to move up from non-programmable or slower beginner robots.

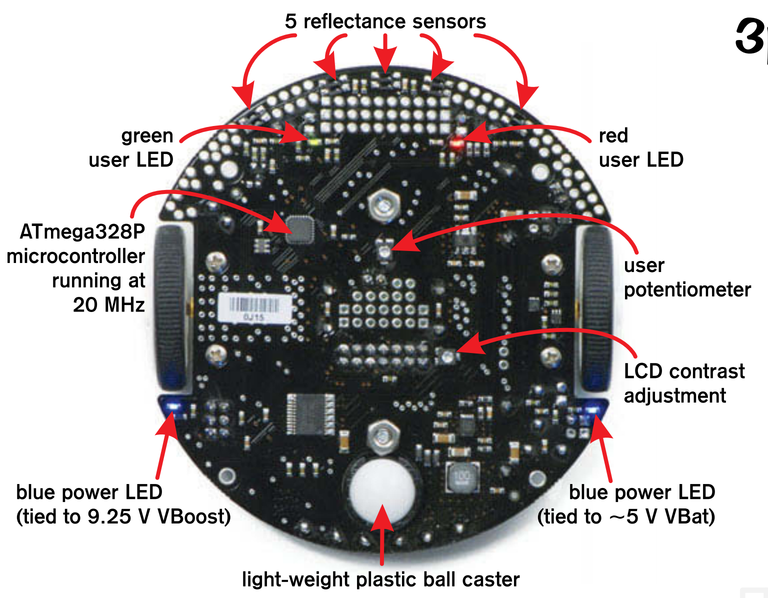


Item #975: $99.95
Available at www.pololu.com/3pi


battery charger connector
piezo buzzer
4 AAA batteries (not included)
30:1 micro metal gearmotors
optional power LED
ISP programming connector
push-on/push-off power button
reset button
removable 8×2 character LCD
user pushbuttons
robot diameter is 3π cm (~3.7 inches)

3pi Line-Following and Maze-Solving

The 3pi robot is designed to excel in line-following and maze-solving competitions. It has a small size (3.7" diameter, 2.9 oz without batteries) and takes just four AAA cells (not included), while a unique power system runs the motors at a constant 9.25 V independent of the battery charge level. The regulated voltage allows the 3pi to reach speeds up to 100 cm/second while making precise turns and spins that don't vary with the battery voltage.


5 reflectance sensors
green user LED
red user LED
ATmega328P microcontroller running at 20 MHz
user potentiometer
LCD contrast adjustment
blue power LED (tied to 9.25 V VBoost)
blue power LED (tied to ~5 V VBat)
light-weight plastic ball caster

3pi Features

- Two metal gearmotors
- 8×2 character LCD
- Five reflectance sensors
- Buzzer, LEDs, three user pushbuttons
- High-traction silicone tires
- Speeds exceeding 3 ft/sec using innovative constant-voltage motor supply
- On-board C/C++-programmable Atmel AVR ATmega328P microcontroller
- Works great with free GNU C/C++ compiler and free AVR Studio development environment.
- Extensive C/C++ libraries and example programs make it a snap to use all hardware features.

Pololu Find out more at www.pololu.com/3pi or by calling 1-877-7-POLOLU the signal is inverted on one of the transmitters. On the receiving side is a pair of receivers, one of which takes the inverted signal and inverts it again, thus restoring the polarity back to normal. This reinverted signal is summed with the signal that was sent unchanged, and the result is the final output signal. Figure 1 shows the basic concept. There is a reason for going through all this trouble. If any electrical noise is induced on the line during transmission, it will be induced the same amount on both wires. However, since the receiver inverts one line and then sums them, the noise will now be inverted on one of the lines. When the two signals are added together, the noise cancels itself out, leaving just the pristine signal. Unlike RS-232, which requires the use of a negative voltage supply, RS-485 is single ended, which simplifies power issues. The RS-485 spec can reliably send data up to 1,200 m.

There are two standard network topologies used with RS-485: half duplex and full duplex. In a half-duplex system, as shown in Figure 2, two wires are used to connect all nodes. Each node's transmitter pair and receiver pair are connected to these two wires, and any node can communicate with any other node. The advantage is that you only need two wires and a common ground to enable communication—thus

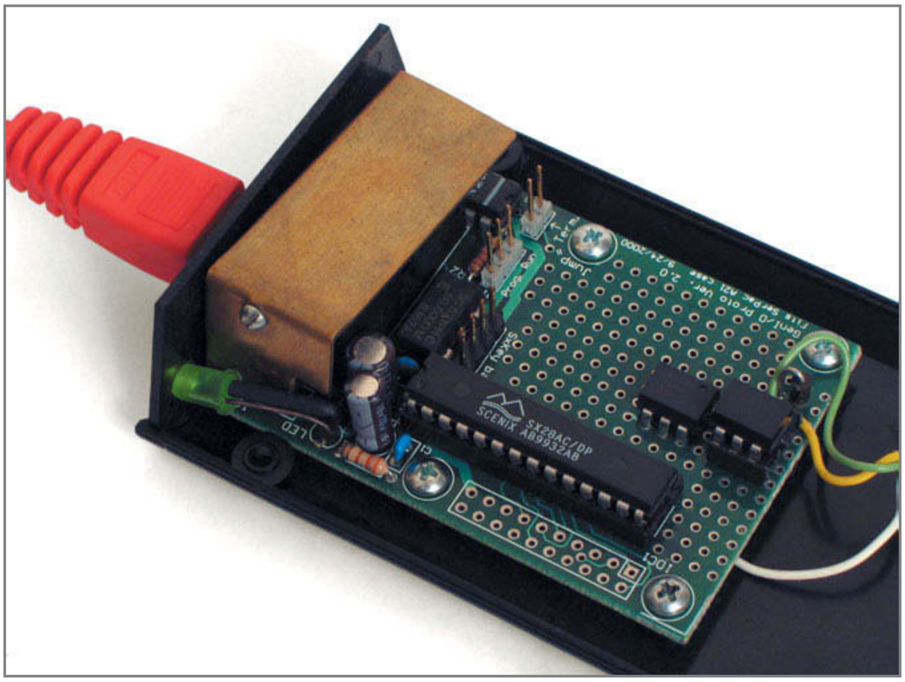


Photo 2—This is a generic I/O node from my RS-485 networked show system.

Easy Embedded Linux

OMNI - EP

- 200 Mhz ARM9 CPU
- 10/100 Mb Ethernet
- 32 MB RAM
- 16 MB Flash
- 16 Digital I/O Lines
- 2 Ports of USB 2.0
- SPI Bus
- AC97 Amplified Audio
- Battery Backed Clock
- 2 Serial Ports
- Low Power Consumption
- RoHS Compliant

Our newest ARM9 Linux controller, the OmniEP doesn't cost an arm and a leg. It delivers removable storage, amplified audio, ethernet and serial RS232 communication ports in a rugged and attractive enclosure. Models without enclosure and LCD available.

The OmniEP comes preloaded with Linux to jumpstart your development process, with LCD and pushbutton drivers supplied. Large capacity USB drives can be easily mounted in the USB port.

Call **530-297-6073** Email **sales@jkmicro.com**
www.jkmicro.com

JK microsystems, Inc.

International Orders Welcome MADE IN U.S.A

FlashPro430
FlashPro-CC
FlashPro2000
GangPro430
GangPro-CC

USB Flash Programmers for Texas Instruments' MCUs
MSP430, Chipcon CCxx, C2000 DSPs

**Reliable and the fastest programmer on the market.
Perfect for production usage.**

* can assign unique serial number
* up to eight programmers can be connected to one PC and program target devices simultaneously

Elprotronic Incorporated

www.elprotronic.com

simplifying wiring. The disadvantage is that it complicates the actual communication since only one node can transmit on the wire pair at a time. If two or more nodes try and transmit simultaneously, the resulting collision will simply create digital noise and no data will get through.

The full-duplex system uses two wire pairs and requires the use of a master/slave system. The node designated as the master can communicate with every slave, and vice versa, but slaves can't talk to each other. The reason is that one wire pair connects the transmitters of the master to all the slaves, and the other wire pair connects all the transmitters of the slaves to the master receiver pair (see Figure 3).

AN RS-485 HUB

If a single pair of wires can have multiple receivers or transmitters, why make a hub? The reason is that even with a full-duplex setup with multiple slave transmitters on a wire pair, only one of them can talk at a time. The issue of multiple devices trying to use a shared resource like a network is at the heart of all network design. Ethernet enables multiple transmitters and receivers to hang on the wires, but the Ethernet spec also dictates a collision detection and retransmission scheme. After checking that the line is not in use, an Ethernet device transmits while monitoring the line to make sure the data on the wire is the same as what it is sending. If it isn't, the device assumes some other device is transmitting simultaneously. All Ethernet devices do this, and when they detect collisions, they both stop and wait a randomized amount of time before retransmitting. In theory, the randomization means one will start transmitting first and the other will wait for the line to be clear before it transmits.

This works quite well in practice. Witness the billions of Ethernet devices operating worldwide right now. However, the RS-485 spec is only an electrical spec, meaning that issues such as collision detection are left up to the individual designer to take care of. As a result, most folks

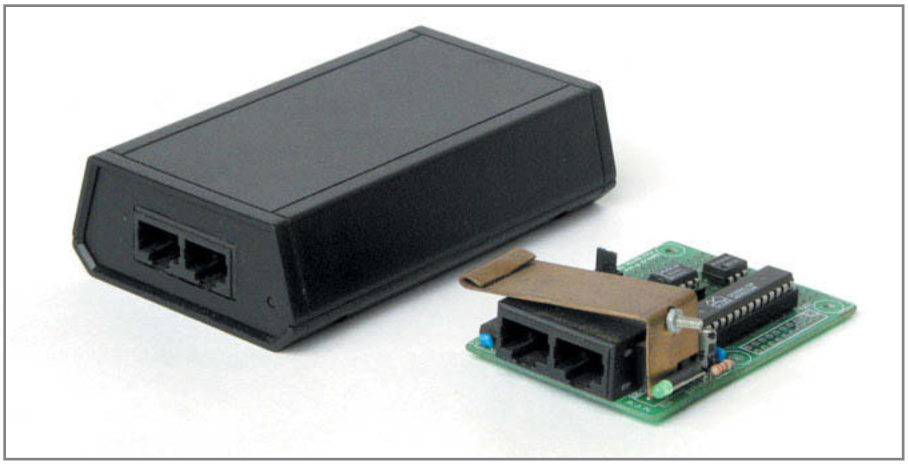
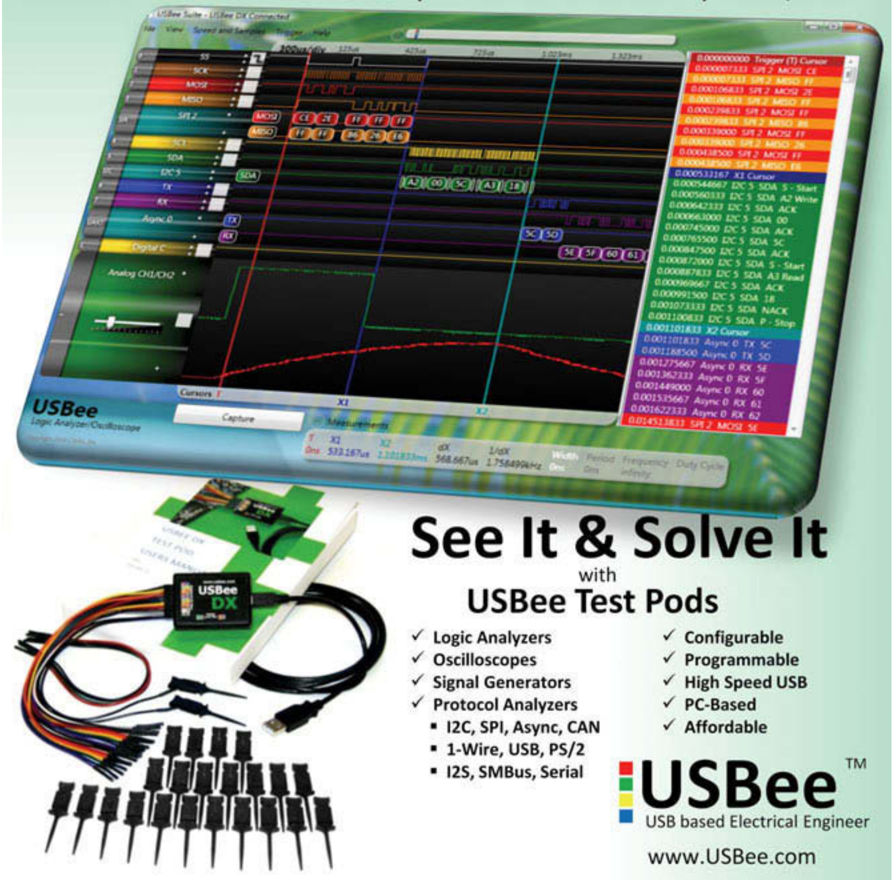


Photo 3—This hardware node has RS-485 networking. The boar d on the right is the same as what's in the case on the left.

bypass collision detection and use the much simpler approach of letting only one transmitter talk at a time.

There are numerous common approaches to handling this rule. My system uses the full-duplex master/slave concept with polling. In this approach, the master is the only node that can initiate communication. Slaves never send a packet without the master node first requesting information from that slave node. Further simplifying communication, slave nodes do not send any packets back to acknowledge having received a packet from the master node. The master assumes that the packets it sends are being received. Having each slave acknowledge packets would not only complicate the network software, but

Visit us in Booth 1142 at the Embedded Systems Conference in San Jose April 27-29, 2010



See It & Solve It
with
USBee Test Pods

- ✓ Logic Analyzers
- ✓ Oscilloscopes
- ✓ Signal Generators
- ✓ Protocol Analyzers
  - I2C, SPI, Async, CAN
  - 1-Wire, USB, PS/2
  - I2S, SMBus, Serial

- ✓ Configurable
- ✓ Programmable
- ✓ High Speed USB
- ✓ PC-Based
- ✓ Affordable

USBee™
USB based Electrical Engineer
www.USBee.com

A new era, a new ESC

New tracks. New courses. New focus.

ESC Silicon Valley brings together systems architects, design engineers, suppliers, analysts, and media from across the globe. With cutting edge product demonstrations, visionary keynotes, and hundreds of essential training classes, ESC is the ideal conference for the embedded design community to learn, collaborate, and recognize excellence.

ESC Silicon Valley 2010 features:

- Two "Build Your Own Embedded System" programs featuring the new Intel Multicore Atom and the Freescale Tower development system
- Multicore Expo and Conference co-located with ESC Silicon Valley
- ARM Partner Pavilion
- Display and Connected Devices Emerging Technology Zones
- Keynotes from Dr. Michio Kaku, theoretical physicist and bestselling author and Richard Templeton, CEO & President, Texas Instruments

Start your own personal development at ESC 2010. You can't afford to miss it.

Silicon Valley
McEnery Convention Center, San Jose, April 26 - 29, 2010
Conference: April 26 - 29, 2010
Expo: April 27 – 29, 2010

Register Today.
www. embedded.com/sv

Learn today. Design tomorrow.
ESC

it would greatly slow down the system. Imagine that the master wants to send packets to nodes 1 and 2. If the system had each slave acknowledge every packet sent, the master could not send a packet to node 2 until it received an acknowledgement from node 1. Otherwise, there is the chance that both nodes 1 and 2 could overlap their reply packets, thus destroying the replies from both.

Requiring acknowledgement of packets received also requires a timeout period to prevent the master node from waiting forever for a slave acknowledgement packet. Timeout periods are trickier than they seem. Too long, and the system response is slowed to a crawl when a slave node has a problem. Too short, and the slave could send the acknowledgement after the timeout period. This new packet would also require an acknowledgment, and the problem could keep repeating. Suppose we create a timeout period of 0.25 s. While this certainly sounds like a nice, short duration, the potential for network slowdown due to a timeout is massive. The minimum packet size in my Halloween Network is 4 bytes and the network runs at 9,600 bps. Thus, in 0.25 s, the system could theoretically send 60 packets. If a node was not responding for some reason, it would send only one packet in that same 0.25 s, thus making the network run 60 times slower. Okay, let's cut the timeout in half and have a 0.125-s timeout. It sounds good until you realize that a timed out node will still cause the network to run 30 times slower than it should. You can see the problem is not easily solved simply by shrinking the timeout period. This is why I do not have slave nodes send acknowledge packets.

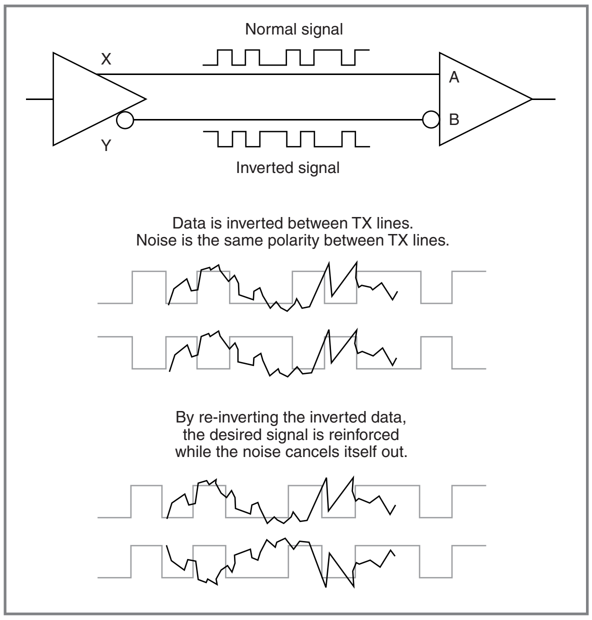

Normal signal
X
A
B
Y
Inverted signal
Data is inverted between TX lines.
Noise is the same polarity between TX lines.
By re-inverting the inverted data, the desired signal is reinforced while the noise cancels itself out.

Figure 1—RS-485 uses differential signaling to eliminate noise. Any noise induced on the twisted pair transmission lines will be canceled out when the "B" channel is inverted at the receiver and summed with the "A" channel.

I have a laptop, which is the master, and all the hardware nodes that do the real work of running the show are the slaves. Most of the nodes are output-only devices and simply need to be told what to do. For example, the smoke machine node simply triggers an output bit connected to an opto-relay that then triggers the actual smoke machine. If the smoke machine node misses a packet, the worst-case scenario is that a puff of smoke doesn't get created. However, it will be getting another packet in a few seconds to generate another smoke puff, and the end result doesn't impact the show. Input nodes are a different case. For example, I have a node that is connected to an infrared beam system that can tell when a person has passed by. This is used to trigger events in the show. That node clearly has to tell the laptop when the beam has been broken. The preferred approach would allow the node to transmit whenever the beam got broken; but instead, I have a program on the PC polling the node four times a second to check the status. Like most polling schemes, it is an inefficient use of network bandwidth and processor overhead. I would love to be able to have that node simply send a message whenever the event happens without risking packet collisions.

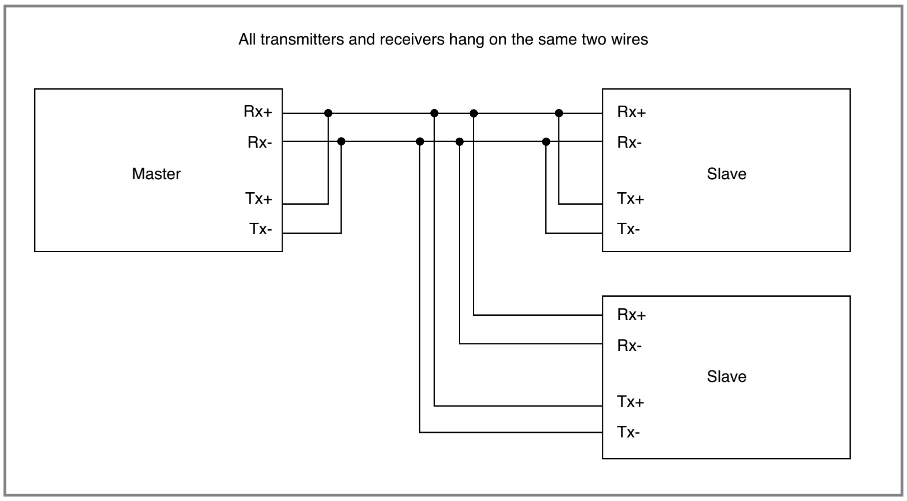


Figure 2—Many RS-485 systems use a simple half-duplex topology where all receivers and transmitters hang on the same wire pair. Software prevents more than one transmitter from operating at the same time.

NETWORK TOPOLOGY

To help allow this, I redesigned my hardware nodes to use a "store and forward" scheme, where the node receives messages from downstream nodes on one port, and then forwards those upstream to the laptop using another physically separate port. If the node has to send a message at the same time, it will send its own message first and then send the stored message. This is a modified full-duplex topology, which is shown in Figure 4. The physical implementation of a hardware node is shown in Photo 3. Note the use of a two-port RJ-45 jack. One jack is designated as the upstream (closer to the master) port, and the other the downstream (farther from the master) port. Photo 4 shows a pair of nodes daisy-chained together. Each hardware node has two receiver UARTs and RS-485 chips and one transmitter UART and RS-485 chip. By having in essence a dedicated link between each node heading back upstream to the master, any slave can transmit at any time without fear of collision since it is the only node on that dedicated link.

While it's a good system, there are numerous drawbacks. First, each node needs an extra RS-485 receiver chip—one for receiving upstream messages from the master, and one for receiving downstream messages from the node below it. This adds an extra chip to the design. It also requires more complicated software on each node, because each node must handle storing and forwarding messages from other nodes, along with interleaving messages from itself as needed. There is also a time delay introduced by each and every node in the system because a node has to completely receive a packet before sending it on. Finally, most small embedded processors are incredibly light on RAM, meaning that there isn't much space to buffer messages.

The time delay issue is subtle. It doesn't sound like it would be such a big deal until you do the math. With the Halloween network running at 9,600 bps, and the smallest network packet being 4 bytes long, receiving a packet takes about 4.2 ms. If I had 16 nodes daisy-chained together, sending a message from the farthest node to the laptop would take 67 ms. While that may not sound like much, it means that the farthest node can only send a maximum of 15 messages per second instead of 240. It also means that the delay time varies depending on where a node is located in the chain.

Up until last year, I'd been able to work around this issue satisfactorily. However, that changed when I created the Halloween Remote ("Dynamic Animatronic Remote," Parts 1 and 2, *Circuit Cellar* 218 and 219, 2008). Unlike most of the nodes in my system, the remote needs to sends lots of packets back to the host since it is controlling programs on the laptop. Further, the messages aren't all tiny 4-byte messages. They can get much bigger. All of this puts a strain on the limited buffering space of the nodes. In addition, the remote itself doesn't store and forward since that would require dragging two cables (one upstream and one downstream) around as I used it. Since the remote has only a single cable, I would be forced to plug the remote in as the system's last node. This is the node farthest downstream from the master and

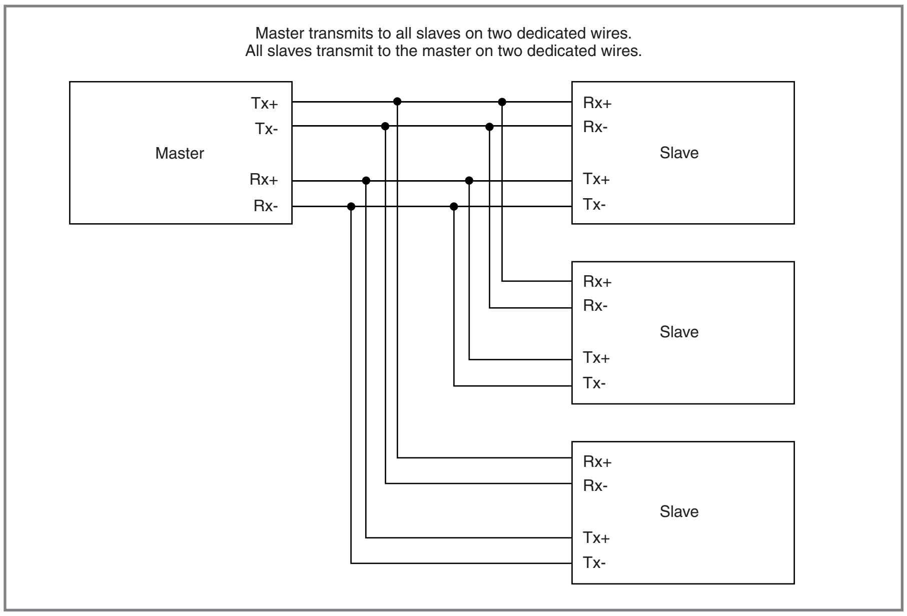

Master transmits to all slaves on two dedicated wires.
All slaves transmit to the master on two dedicated wires.
Master
Tx+
Tx-
Rx+
Rx-
Slave
Rx+
Rx-
Tx+
Tx-
Slave
Slave

Figure 3—Full-duplex networks typically have a master/salve arrangement: one wire pair lets the master transmit to all slaves and the other pair lets the slaves report back to the master . Software ensures only a single slave transmits at a time.

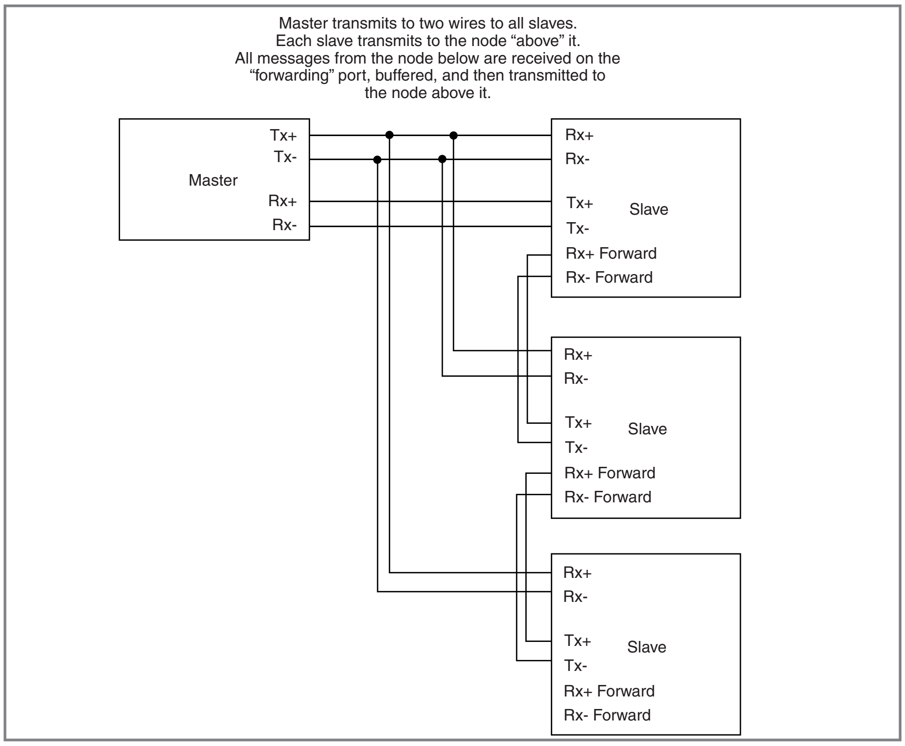

Master transmits to two wires to all slaves.
Each slave transmits to the node "above" it.
All messages from the node below are received on the "forwarding" port, buffered, and then transmitted to the node above it.
Master
Tx+
Tx-
Rx+
Rx-
Slave
Rx+
Rx-
Tx+
Tx-
Rx+ Forward
Rx- Forward
Slave
Slave

Figure 4—The network "store and forward" system is also master/slave. The extra hardware allows each slave to transmit at any time as all slaves buffer messages heading back to the master node.

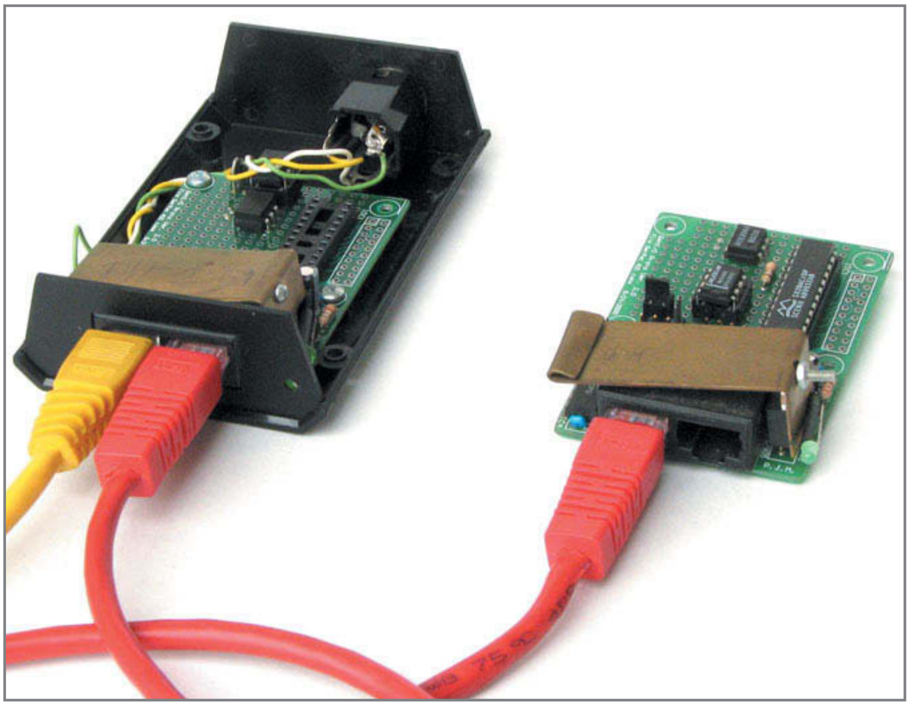


Photo 4—A pair of nodes is daisy-chained together. The left node is in the case with the top removed. The right one shows the "raw" board.

having the greatest delay time sending packets to the master. Forcing the remote to have the slowest response time and having to physically plug it in at the end of the node chain made using it problematic. I wanted to be able to plug it into some fast central location that made walking and using it easier.

HUB DESIGN SPECS

For all of these reasons, I decided the time was right to create the Halloween Serial Network Hub. This is a six-port hub that enables me to plug any node or string of nodes into a port, and it takes all incoming packets, buffers them as needed, serializes them, and then forwards them to the laptop. While there is a delay added, it doesn't accumulate with each node in the chain. My nodes have a jumper so they can operate in either Store and Forward mode, or simply daisy-chain them using the standard full-duplex approach. If I have a group of nodes that only need to transmit when polled, I can set them up in plain daisy-chain mode. This means that there is no additional delay no matter how many nodes are strung together. The sole delay is created by the network hub, and that delay will be just the length of a single packet if no other node is transmitting at that time. In addition, I made the link between the laptop and the hub run at 38,400 bps. This means that when a packet is fully received by the hub, it then forwards it to the laptop four times faster than it was received. This helps minimize the added delay while letting the main network run at a slower, and more robust, 9,600 bps.

I had a certain criteria in mind when I set out to design the hub. I wanted to write as much of the code as possible in C language to make my life easier. Also, I didn't want to have a pile of UART chips on the board. However, finding a six-port UART chip is somewhat tricky.

The hub also needs the UART connected to the laptop to run at RS-232 signal levels, since normal PCs and laptops don't come with RS-485 ports. The Halloween network uses standard CAT-5 cables to not only send and receive data, but also to provide 12-V DC power to run each node. I was sick of devices powered by wall warts and wanted to keep the entire unit self-contained. This meant fitting a small power supply into the case as well.

Finally, I wanted to create a small device that looked as professional as possible. This last requirement was more a personal challenge than an actual requirement; but since I use this equipment year after year, I found the extra time spent on making the packaging solid has paid off in equipment that hasn't been flaky in the field. My work was cut out for me.

In the second part of this article series, I'll explain exactly how I created my RS-485 hub. I'll include details about the software and hardware, as well as some hidden "gotchas" present in RS-485 chips.

Peter Montgomery (PJMonty@csi.com) spent 12 years working as a visual effects supervisor on films such as Mortal Kombat *and* Ace Ventura: When Nature Calls *before becoming a director. He has directed dozens of commercials and made the transition to episodic television with The Disney Channel's* Lizzie McGuire. *Peter is self-taught in both programming and digital hardware design.*

PROJECT FILES

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

SOURCES

NFM-20-12 Power supply
MEAN WELL Enterprises Co. | www.meanwell.com

SX-28 Proto Board
Parallax, Inc. | www.parallax.com

TLV1117 Linear power regulator
Texas Instruments, Inc. | www.ti.com

Z8F4802 Microcontroller
Zilog, Inc. | www.zilog.com

by Marco Aiello

Design and Program a "Minirobot"

Control capability is at the heart of most robotics applications, particularly when object avoidance is a project requirement. This article details how to build a mobile "minirobot" and implement an autonomous navigation system.

Nowadays, Linux is literally everywhere—from toys (e.g., I-Sobot) and mobile phones (e.g. Nokia) to network printers and storage devices. A serious engineer can't afford to ignore this hot topic. So, I took the bull by the horns and used Linux for an embedded project—a mobile robot I refer to as a "minirobot" (see Photo 1).

The wood-framed robot has a couple of Tamiya DC motors and a Tamiya ball caster that enable it to move around. It features a custom power board built around a Microchip Technology dsPIC30F2010 that translates serial commands into the right PWM power signals to drive the two motors and an OLIMEX SBC running a Debian distribution on an Atmel AT91SAM19260 ARM9 CPU (see Photo 2). The design also has a USB 2.0 hub hosting a Creative webcam, two pen drives (one in the EXT3 file system for the Linux root file system and the other in FAT32 for the kernel image and the executable file), and a USB-to-serial converter cable. Another custom power board generates the right voltage for the hub and the SBC, starting with the voltage provided by the battery pack. Four white LEDs serve as a uniform source of illumination. The entire system is battery operated. And once started in Linux, it is fully autonomous.

My autonomous navigation system—which I developed with The MathWorks's MATLAB—drives the robot straight until it reaches an obstacle. It then changes the robot's direction. I used a webcam for this purpose.

In this article, I'll describe the entire project from start to finish. I'll cover artificial vision algorithm development (in MATLAB), the installation of the cross compiler (in Linux Ubuntu 8.04), the kernel-compiling process, the translation in C and subsequent debugging phase, and the fine tuning of the SBC (see Photo 3).

HARDWARE

The "brain" of the robot is an OLIMEX SBC. By coupling its rich hardware endowment with a powerful operating system like Linux, the only limit is your imagination. The board has only one USB host port, so I used a USB 2.0 active hub. I plugged the two USB pen drives, the webcam, and the USB serial dongle into the active hub.

The aforementioned parts make up the COTS portion of the robot. Now let's focus on the motor board, which I developed because my previous Pololu COTS serial DC motor controller broke during the testing phase. I was in a hurry and low on cash, so I developed a replacement for it on a simple protoboard. At the end of the development process, I covered it with a silicone rubber sealant—which you can buy at most hardware stores—to avoid the risk of breaking the tiny cables.

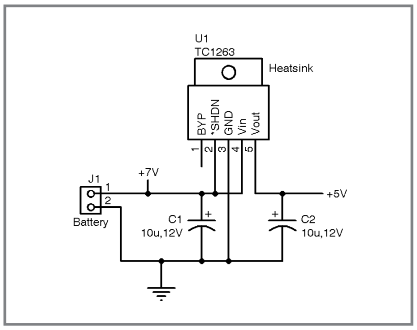


Figure 1—Here's the voltage regulation section of the motor board. It's simply built around a Microchip's linear regulator and few other discrete components.

Four AA batteries power the motor board. As you can see in Figure 1, a Microchip Technology TC1263–5.0 VAT linear regulator and an associated

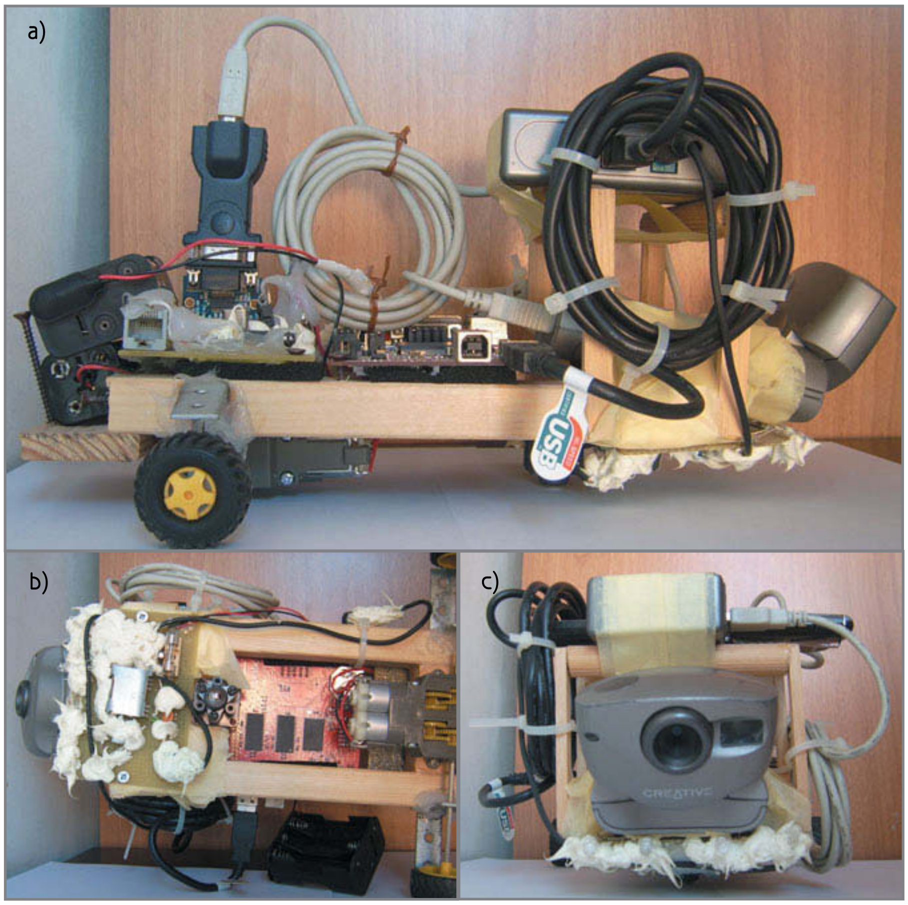


Photo 1a—Here's the robot fully assembled with the 10 batteries housed in their battery holders. Once Linux is started and the navigation program is running, you can freely unplug the serial null modem and let it roam around. b—This is a view from above the design. c—I used a Creative webcam to give the system optical capabilities.

bypass capacitor limit the voltage to the 5 V required by the serial converter built around a Maxim IC. But you can wire-out your own serial converter with an MAX232 chip plus the required capacitors (see Figure 2).

The heart of the board is a dsPIC30F2010 in a DIP package that's especially suited to motor control applications (see Figure 3). It's responsible for generating the PWM signals required to drive the bases of the two power Darlington BJT BDW94s, which provide the driving current required by the two small DC motors in the Tamiya gearbox. The two BJTs are supported by two free-wheeling diodes and two resistors that limit their base current (see Figure 4).

cleverscope

100 MHz MSO 8M Samples 14 bit

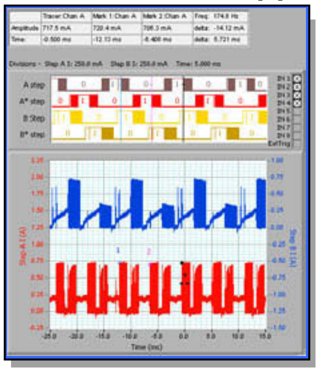
+ Two mixed signal triggers
+ Protocol decoding
+ Spectrum analysis
+ Symbolic maths
+ Custom units
+ Copy & paste
+ Signal generator
+ USB or Ethernet
+ 4 or 8M samples storage
+ 100 MHz sampling
+ Dual 10, 12 or 14 bit ADC
+ Ext Trigger, 8 Digital Inputs
+ 1 MSa/sec charting

Robotics – stepper motor control

Many robotics projects use stepper motors to generate motion. Cleverscope can be used to verify stepper motor operation, check control loop stability, and check on the motor currents being generated. The custom units, math's and filtering, and spectrum analysis features are useful here.

www.cleverscope.com

Robotics - stepper motor control

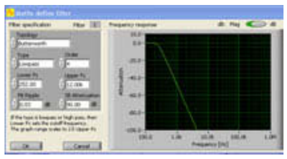


	Tracer:Chan A	Mark 1:Chan A	Mark 2:Chan A	Freq: 174.6 Hz
Amplitude	717.5 mA	720.4 mA	706.3 mA	delta: -14.12 mA
Time	-0.500 ms	-12.13 ms	-6.400 ms	delta: 5.721 ms

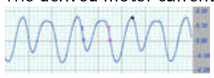
Divisions - Step A 1: 250.0 mA Step B 1: 250.0 mA Time: 5.000 ms

Example:

2 Phase stepper motor control. We use a 2 ohm current sensing resistor, and user units to convert to Amps. The stepper motor was modeled as a low pass filter to estimate the winding current. This is the Filter used:



The derived motor current:



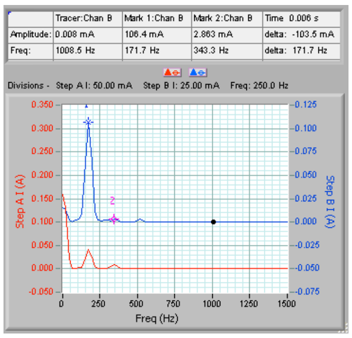
See Examples page at www.cleverscope.com

The difference between the two winding currents is an estimate of the motor torque, which should be sinusoidal. Check stability by ramping up the stepper motor frequency while plotting motor current, with peak hold enabled.

In the USA call:

Saelig .com
unique electronics
1-888-7SAELIG info@saelig.com

The motor current spectrum (Chan B):



	Tracer:Chan B	Mark 1:Chan B	Mark 2:Chan B	Time 0.006 s
Amplitude:	0.008 mA	106.4 mA	2.863 mA	delta: -103.5 mA
Freq:	1008.5 Hz	171.7 Hz	343.3 Hz	delta: 171.7 Hz

Divisions - Step A I: 50.00 mA Step B I: 25.00 mA Freq: 250.0 Hz

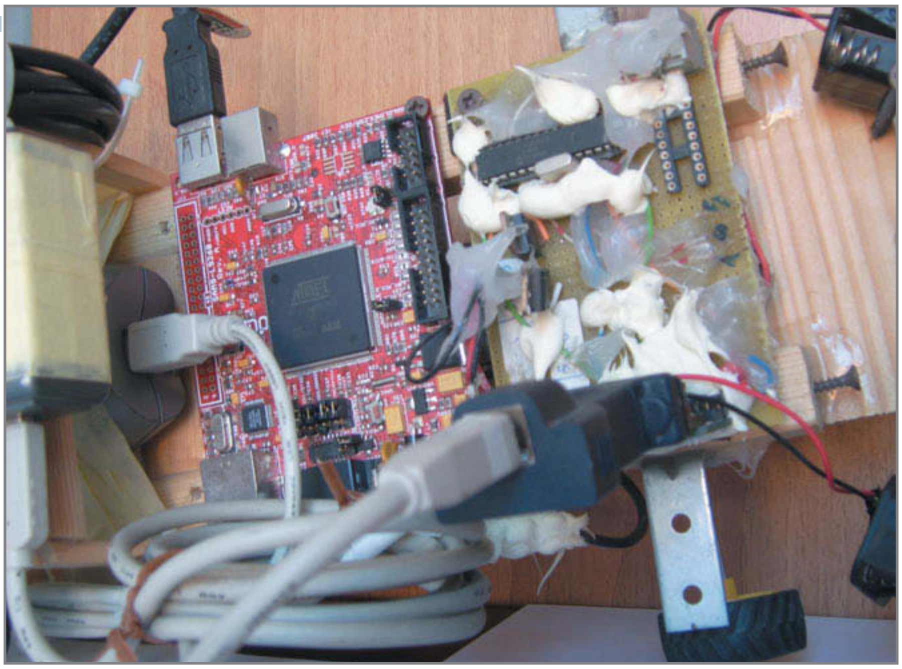


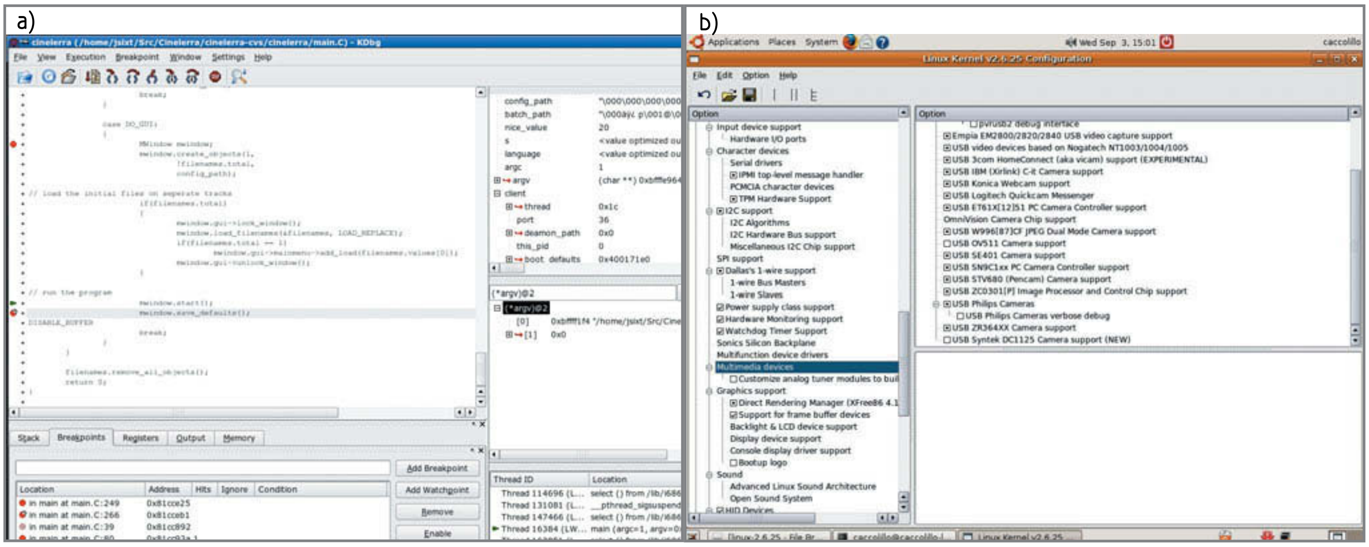
Photo 2—Here's a close-up photo of the motor boar d. It is covered in silicone (to har den it), but you can still see the USB serial dongle plugged in.

The Microchip part works well thanks to its rich peripheral set. The PWM module generates the proper waveform to drive the two BJTs. The UART communicates with the SBC, which determines the navigation direction. Thanks to the application notes and the power of Microchip's development tools, I debugged and programmed with my trusty companion ICD2 in less than three days. I replaced my fallen Pololu controller and implemented the components I already had on hand.

The power board's firmware is simple. Thanks to the free version of the MPLAB C dsPIC compiler, I was able to write all the firmware required by the board in C language. (The code is posted on the *Circuit Cellar* FTP site.) After an initialization phase (made by the MPLAB's visual device initializer), a state machine is activated at each UART's interrupt, which searches for a valid received command packet. Once one is detected, it updates the value of the two duty cycle registers in the corresponding PWM channels.

The power board's design is straightforward (see Figure 5). It's powered by six AA batteries and includes surge protection. (Why not use some affordable protection to safeguard all the costly hardware?) There are two linear LM7805 voltage regulators. One is responsible for powering the SBC board. The other powers the USB hub. Both of them are provided with an adequate heatsink. Two groups of white LEDs, powered directly by the battery pack with associated 220-Ω current limiting resistors provide the illumination required by the Creative webcam for acquiring the frames. I mounted all the boards with self-tapping screws on the wooden frame. I fastened the webcam and hub with tape.

Now let's focus on the frame—which is made of wood glued together—and the arrangement of parts. The power board is positioned in the back of the frame. The SBC is housed in the middle section. The webcam is located in the front. Under the webcam is the power board and behind it is the Tamiya ball caster. The twin Tamiya gearbox is in the rear. The USB active hub is supported with tape. All of the USB and power cables are


a)
cinelerra (/home/jsixt/Src/Cinelerra/cinelerra-cvs/cinelerra/main.C) - KDbg
File View Execution Breakpoint Window Settings Help
Stack Breakpoints Registers Output Memory
Add Breakpoint
Add Watchpoint
Remove
Enable
b)
Applications Places System
Wed Sep 3, 15:01
caccolillo
Linux Kernel v2.6.25 Configuration
File Edit Option Help

```
                        break;
                    }

                case DO_GUI:
                    {
                        MWindow mwindow;
                        mwindow.create_objects(1,
                                !filenames.total,
                                config_path);

// load the initial files on seperate tracks
                        if(filenames.total)
                        {
                                mwindow.gui->lock_window();
                                mwindow.load_filenames(&filenames, LOAD_REPLACE);
                                if(filenames.total == 1)
                                        mwindow.gui->mainmenu->add_load(filenames.values[0]);
                                mwindow.gui->unlock_window();
                        }

// run the program
                        mwindow.start();
                        mwindow.save_defaults();
DISABLE_BUFFER
                        break;
                    }
            }

        filenames.remove_all_objects();
        return 0;
}
```

```
config_path        "\000\000\000\000
batch_path         "\000ajv. p\001@\0
nice_value         20
s                  <value optimized ou
language           <value optimized ou
argc               1
⊞ argv             (char **) 0xbfffe964
⊟ client
   ⊞ thread        0x1c
     port          36
   ⊞ deamon_path   0x0
     this_pid      0
   ⊞ boot_defaults 0x400171e0

(*argv)@2
⊟ (*argv)@2
     [0]    0xbffff1f4 "/home/jsixt/Src/Cine
   ⊞ [1]    0x0
```

Location	Address	Hits	Ignore	Condition
in main at main.C:249	0x81cce25			
in main at main.C:266	0x81cceb1			
in main at main.C:39	0x81cc892			
in main at main.C:80	0x81cc93a 1			

Thread ID	Location
Thread 114696 (L...	select () from /lib/i686
Thread 131081 (L...	__pthread_sigsuspend
Thread 147466 (L...	select () from /lib/i686
Thread 16384 (LW...	main (argc=1, argv=0

```
Option
 ⊖ Input device support
      Hardware I/O ports
 ⊖ Character devices
      Serial drivers
      ⊡ IPMI top-level message handler
      PCMCIA character devices
      ⊡ TPM Hardware Support
 ⊖ ⊡ I2C support
      I2C Algorithms
      I2C Hardware Bus support
      Miscellaneous I2C Chip support
   SPI support
 ⊖ ⊡ Dallas's 1-wire support
      1-wire Bus Masters
      1-wire Slaves
   ☑ Power supply class support
   ☑ Hardware Monitoring support
   ☑ Watchdog Timer Support
   Sonics Silicon Backplane
   Multifunction device drivers
 ⊖ Multimedia devices
      ☐ Customize analog tuner modules to buil
 ⊖ Graphics support
      ⊡ Direct Rendering Manager (XFree86 4.1
      ☑ Support for frame buffer devices
      Backlight & LCD device support
      Display device support
      Console display driver support
      ☐ Bootup logo
 ⊖ Sound
      Advanced Linux Sound Architecture
      Open Sound System
 ⊖ ☑ HID Devices
```

```
Option
      ☐ pvrusb2 debug interface
   ⊡ Empia EM2800/2820/2840 USB video capture support
   ⊡ USB video devices based on Nogatech NT1003/1004/1005
   ⊡ USB 3com HomeConnect (aka vicam) support (EXPERIMENTAL)
   ⊡ USB IBM (Xirlink) C-it Camera support
   ⊡ USB Konica Webcam support
   ⊡ USB Logitech Quickcam Messenger
   ⊡ USB ET61X[12]51 PC Camera Controller support
   OmniVision Camera Chip support
   ⊡ USB W996[87]CF JPEG Dual Mode Camera support
   ☐ USB OV511 Camera support
   ⊡ USB SE401 Camera support
   ⊡ USB SN9C1xx PC Camera Controller support
   ⊡ USB STV680 (Pencam) Camera support
   ⊡ USB ZC0301[P] Image Processor and Control Chip support
 ⊖ ⊡ USB Philips Cameras
      ☐ USB Philips Cameras verbose debug
   ⊡ USB ZR364XX Camera support
   ☐ USB Syntek DC1125 Camera support (NEW)
```

Photo 3a—As you can see, Kdbg is a powerful fully featured graphical debugger . It played a vital role in speeding up the debugging phas e on the laptop. It also can be used to remotely debug an application on the SBC boar d, but I haven't used this feature. b—This is the graphical configuration of the kernel's building process.

Educators and Students:

Register Today

to Receive Exciting Microcontroller Resources!

RENESAS UNIVERSITY PROGRAM

Renesas — the #1 supplier of microcontrollers in the world — is launching Renesas University, an exciting educational program that gives educators a way to teach microcontroller (MCU) technology using a modern architecture and professional-grade tools. It also offers many valuable resources that help students learn about MCUs and how they can be applied in significant embedded system designs.

#1 MCU REACH FURTHER

Renesas is a worldwide leader in:

- Microcontrollers
- Embedded flash microcontrollers
- MCUs in car navigation systems
- Power amplifiers for GSM phones
- LCD controllers for color mobile displays

The Renesas University program nurtures an online community where educators and students come together to share ideas, address technical issues and discuss microcontroller topics. It is characterized by:

Publish ▸ Renesas actively encourages academics and students to publish microcontroller-related papers. We provide assistance in publishing course material and microcontroller related books.

Toolchain ▸ The Renesas integrated development environment with toolchain is the commercial version of our development tools – with full C compiler, assembler, linker, and debugger. It is not a typical capability-reduced "educational" version. The only limitation is a 64KB code size after 60 days of use.

Modern ▸ Renesas microcontrollers utilize a modern architecture designed specifically for C and other high-level languages. Our devices handle the most demanding applications of today and tomorrow.

Complete Development Kits

Renesas Starter Kits provide a USB-powered, MCU-based system board with in-circuit debugger/flash memory programmer. They include a CD containing our integrated development environment with toolchain, plus documentation, example firmware and interesting projects.

- **Free for Educators:** Register at the Renesas University website to receive ten free Starter Kits per semester. In return, we request the submission of material that enriches Renesas University; i.e., code, student projects, technical papers, embedded control designs, etc.
- **Low cost for Students:** If actively enrolled in an educational institution, a Starter Kit can be purchased at a very low cost after registering at the Renesas University website.

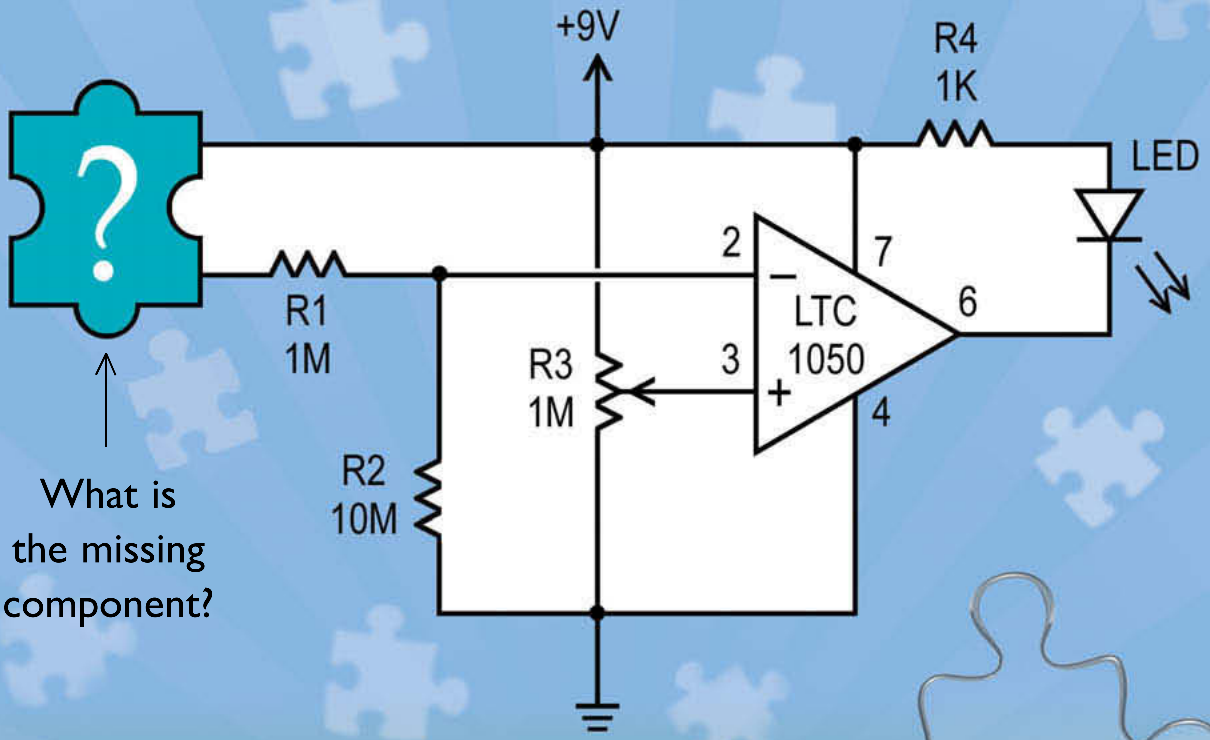
For more information on Renesas University and how to enroll, please visit **www.renesasuniversity.com** or email: University@rta.renesas.com

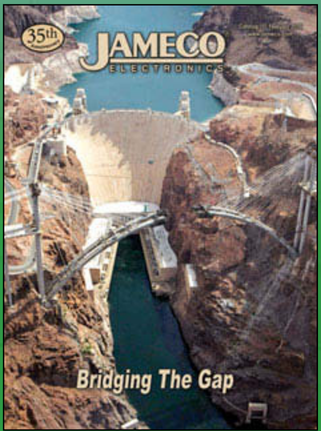
Visit us at ESC Silicon Valley!
Booth #2002

Embedded Systems CONFERENCE SILICON VALLEY

Everywhere you imagine. RENESAS

Are you up for a challenge?





Electronics industry guru Forrest M. Mims III has created yet another stumper. The Ultra Simple Sensors Company assigned its engineering staff to design a circuit that would trigger an LED when a few millimeters of water is present in a basement or boat. What is the water sensor behind the puzzle piece? Go to www.Jameco.com/teaser5 to see if you are correct and while you are there, sign-up for our free full color catalog.

JAMECO® ELECTRONICS

1-800-831-4242 | www.Jameco.com

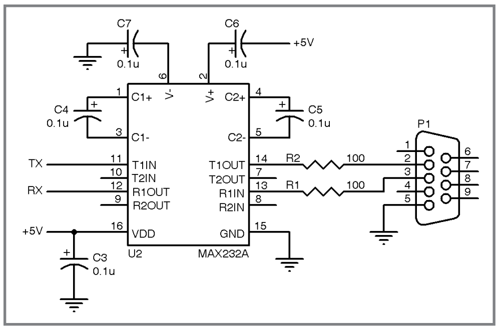


Figure 2—This is the serial voltage level converter. I used a ready-made version on my robot.

arranged and held in place with small plastic bands.

MATLAB ASSISTANCE

MATLAB made developing the navigation algorithm painless. I wrote three MATLAB scripts. One implements the navigation algorithm the robot uses. Another is used when fine-tuning the C source code on the SBC, and its task is to simply show the frames processed during the various steps. The third script is for image size reduction. I wrote it during the SBC testing in an effort to speed up the computation. (In fact, the minimum size of the acquired image supported in hardware by the webcam was too big.) It reduced the frame's dimensions by one-half. The result was a new image in which each pixel represents the average value of four adjacent pixels in the original image. An image-processing step, such as a convolution product, on this "lightweight" images takes a lot less computation time than it would on the original image.

Refer to the MATLAB navigation script posted on *Circuit Cellar*'s FTP site. You start with a color frame acquired with the Creative Webcam Go Control. It's then saved in a file on a desktop and turned into a grayscale image. After that, a Sobel edge detection is performed, followed by a variable thresholding step (based on ambient light intensity) to obtain a binarized image. Next is a search for the nearest obstacle. This task produces an array of the distances to the nearest obstacles. The array is then filtered out by a moving average filter, and then its filtered version is used by a simple expert system to find the right navigation direction (based on a search of the local maximum and area under the graph in each part of the array—left, center, and right).

Thanks to the amazing MATLAB software suite, the preliminary development process was performed quickly without too much trouble. I was inspired by an old MIT robot named "Polly."[1] The navigation script (and also the image reduction script) was used as a starting point for developing the C source code for the robot.

SOFTWARE

I structured the program (which runs in real-time even on the inexpensive SBC board) with a single buffer (also known in literature as the "bounded buffer problem"). The program is organized as follows: the main program (the "parent process") initializes the semaphores and the shared memory, and then opens the webcam (seen in Linux as `/dev/video0`). The USB-to-serial converter (seen as `/dev/ttyUSB0`) then creates two new "child processes." The first is the "producer" that uses the webcam. The other is the "consumer" that owns the serial converter. The parent process then goes to sleep and waits for the CTRL+C key combination to terminate the child processes, release all the resources, and exit and terminate itself.

Once separated, the producer and consumer begin executing (they alternate) and use two semaphores to stay synchronized. This goes on until they get the "kill" signal issued by parent process. The producer's job is simply to get a frame in grayscale format by `/dev/video0` and to place it in the shared buffer.

The consumer is more complex. Once it takes the image from the shared buffer, it reduces its dimension (as showed in the MATLAB image reduction script) and then it implements the aforementioned navigation algorithm. Moreover, it's responsible for sending the right sequence of bytes via the `/dev/ttUSB0` device to the power board to set the proper navigation direction.

Now let's take a look at the development environment. One of the advantages associated with using Linux is that you can develop an application on a laptop or desktop computer (the x86 architecture in practice). You to take full advantage of available resources like its computing power and software resources—such as compilers and graphical debuggers—and then use an adequate cross compiler and port your program to a new architecture. The only change required is the use of a cross-compiler suited for the particular architecture you use (ARM9 in my case).

I used my laptop with Ubuntu 8.04 to get the concurrent version of the navigation algorithm. I installed the package using the "synaptic" package manager. (Of course, you'll need a working Internet connection and the root password.) I then installed the "Kdbg," a simple and powerful visual debugger that's been vitally important during all the testing and developing phases in Linux.

To compile the source code

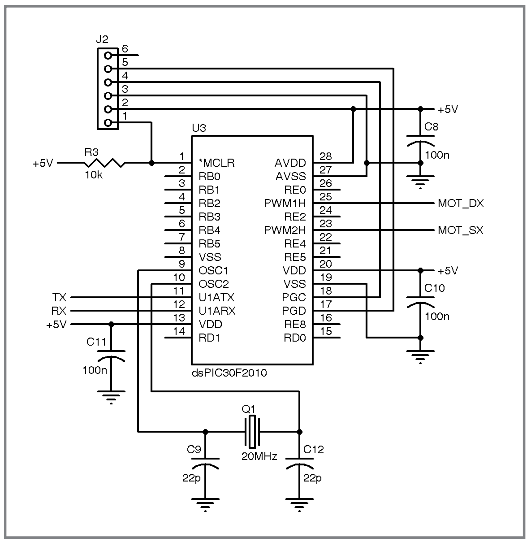


Figure 3—Here's the dsPIC and its connections to the other parts of the board. If you have some time, you can change the firmware and use its internal RC oscillator to save the cost of the quartz.

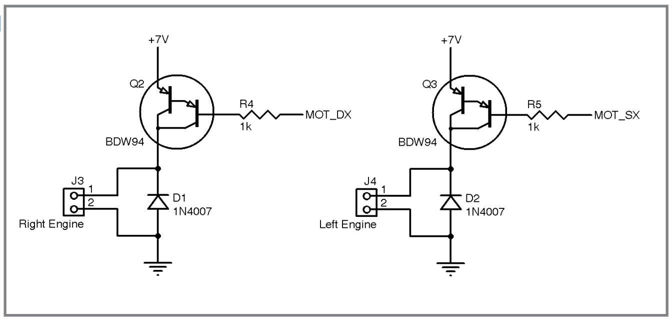


Figure 4—I used the BJTs because I had them on hand. In fact, they were quite oversized for the amount of current required by the DC motors.

(assuming it's named "pcarm.c") in the local folder, I opened a terminal window and used "gcc –g –o pcarm pcarm.c" to compile it and produce the executable "pcarm." I issued `./pcarm` to run it.

MIGRATION TO ARM9

The OLIMEX SBC comes with Linux Kernel 2.6.23 pre-installed and a full Debian distribution. There was only one drawback. Once I started the system, logged in, and issued the `dmesg` command after plugging in the hub, serial converter, and webcam, I discovered the two peripherals weren't supported by the kernel. That was disappointing.

This led me to download the archive with the kernel version 2.6.25 at www.kernel.org. I also downloaded the kernel patch for the processor at http://maxim.org.za.

The documentation on OLIMEX's CD can be a bit difficult to understand, so let me explain some of the details. I installed the cross-compiler provided on OLIMEX's CD on my Ubuntu host PC. Don't feel discouraged. I'll take you step by step so you can have a working cross-compiler too.

When you open the terminal window, type "su" and press Enter. Next, using the "nautilus" file browser, copy the two archives located in the directory named "cross compilers" into your home folder. Via the terminal window, use "tar xjCf / arm-linux-gcc-3.4.3-1.0.1.tar.bz2" to extract the first archive. Then use "tar xjCf / crosstool-linux-gcc-4.0.1-glibc-2.3.5.tar.bz2" to extract the second one.

The next step is to use "gedit /etc/environment," with which you'll open a text editor. Then add the ":/opt/crosstool/gcc-4.0.1-glibc-2.3.5/arm-unknown-linux-gnu/bin" string on the end of the first row where you see the "PATH=..." path. You then save the file and exit to return to the terminal window.

After you close the terminal window, restart Ubuntu. Next, open a terminal window and use "echo $PATH" to check for the "/opt/crosstool/gcc-4.0.1-glibc-2.3.5/arm-unknown-linux-gnu/bin" string. At this point, you have a fully functional cross-compiler for the SBC board. The next step is to compile the kernel. Prior to doing this, use synaptic and install some packages: U-boot mkimage, Linux-libc-dev, Fakeroot, Build-essential, Qt3-apps-dev, Libssl-dev, Libncurses5-dev, Libssl-dev, Bison, Flex, Texinfo, Zlib1g-dev, Gettext, Autoconf, gparted.

Now place the kernel archive in your home folder and extract it. Next, open a terminal window and move to the directory in which it has been extracted (using the "cd " command). There you must copy the patch file for the kernel. Following this, apply the patch with the command "patch –p1 –dry-run < 2.6.25-rc3-at91.patch" and use "patch –p1 < 2.6.25-rc3-at91.patch."

Using "make ARCH=arm xconfig" opens a window for the graphical configuration of the kernel's building process of the kernel (see Photo 3b). Use the "load" command in the menu to open the file "/arch/arm/configs/sam9_l9260_defconfig." Once it's open, select "Multimedia devices" and enable 'Video for Linux." Then go into the "V4L USB DEVICES" submenu and enable every device. Next, go to "SERIAL CONVERTER" and enable every USB serial converter that you find. After performing these steps, the preliminary kernel configuration file will be ready. In the File menu, use the "save" command and return to the terminal window to leave the graphical configuration environment.

The following tells the compiler which configuration file to use for an architecture: "make ARCH=arm CROSS_COMPILE=arm-unknown-linux-gnu sam9_l9260_defconfig." Issuing "make ARCH=arm CROSS_COMPILE=arm-unknown-linux-gnu uImage" starts the compiler. At the end of the process, you find a file named "uImage" in the "./arch/arm/boot" directory. It's a monolithic file containing the kernel plus additional information (e.g., a checksum). The U-Boot bootloader requires this format installed into the SBC.

To be safe, use two additional USB pen drives to prevent the risk of unintentionally damaging the root file system and the kernel on the board's NAND flash. The first drive is formatted with a FAT16 file system. It hosts the uImage file and the robot's executable program. (You can format it with "gparted.") The second drive is formatted with the ext3 file system. Extract the archive in the "root images"

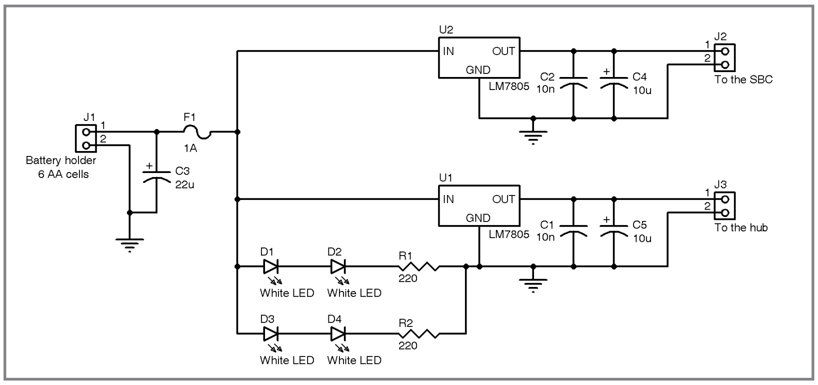


Figure 5—This is the power board. Don't forget to place a couple of heatsinks on the two linear regulators. They tend to get hot after a while.

folder on the OLIMEX CD.

Congratulations! Now you have all the software needed to boot and run the system.

APP DEVELOPMENT

Once I developed and assembled the hardware, it was time to boot my robot. Using "synaptic," I installed the "cutecom" package and launched it. I set it for 115,200 bps, 8 data bits, 1 stop bit, no parity, and no handshake. After I opened the COM port, I attached a null modem cable between the COM port connector on the OLIMEX board and the PC running cutecom.

Next, I placed the batteries in the two battery holders. I didn't see any smoke, so I assumed everything was OK. (I'm joking.) I did, however, see cutecom messages on the screen. To break the boot process, I pressed Enter until I received the "U-Boot>" prompt. I then issued the "usb reset" and "usb scan" commands. Next, I used "fatload usb 0:1 0x21500000 uImage" and then "setenv bootargs mem=64M console=ttyS0,115200, noinitrd root=/dev/sdb1 rootdelay=10." After using "bootm 21500000," the yellow LED on the SBC began blinking.

When the Linux OS started, I logged in with the user ID "root" and the password "olimex." I used "mount /dev/sda1 /mnt/usb" and then "/mnt/usb/pcarm" to run the program. The robot then started its task.

If you have some trouble when you try this, change the position of the pen drives on the hub until everything is fine. To stop the robot, simply press Enter.

DESIGN SUCCESS

Compared with the MATLAB version, the C version uses integer arithmetic whenever it's possible. Furthermore, the kernel matrix used in the blurring operation is much smaller. I've used many other programming tricks (e.g., image size reduction) to speed up the algorithm's execution on the board. (You really feel the lack of an FPU here.) Even though the code on the PC is the same as what's on the SBC, you can see the ARM architecture is less powerful than a laptop's CPU. But bear in mind that the SBC draws much less power and is smaller.

The simple system used to drive the robot has limitations. At some point, I will replace it with a neural network (an LVQ or an MLP). In fact, you can improve the system's performance by using the filtered vector of the distances as an input to the network and the output of the network to set the right direction. Using MATLAB and its neural network toolbox, playing with neural networks should be a piece of cake.

I'm very pleased with this project. It was well worth the effort. ■

Author's note: A project video is posted at www.youtube.com/watch?v=_4KwmhHec2Q.

Marco Aiello (caccolillo@yahoo.com) was a fifth-year student studying Electrical Engineering at the University of Naples Federico Secondo in Italy when he wrote this article. His interests include artificial intelligence, VHDL design, embedded systems, and digital control. In his spare time, Marco enjoys listening to music and watching old Japanese films.

PROJECT FILES

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

REFERENCE

[1] I. Horswill, "A Simple Cheap and Robust Visual Navigation System," *SAB92*, 1992.

RESOURCES

E.R. Davies, *Machine Vision, Theory, Algorithms, Practicalities*, Third Edition, Morgan Kaufmann Publishers, 2004.

M. S. Nixon and A. S. Aguado, *Feature Extraction and Image Processing*, First Edition, Elsevier, 2002.

C. Hallinan, *Embedded Linux Primer: A Practical Real-World Approach*, Prentice Hall Open Source Software Development Series, 2006.

K. Haviland, D. Grey, B. Salama, *UNIX System Programming: A Programmer's Guide to Software Development*, Second Edition, Addison-Wesley, 1999.

C. Hollabaugh, *Embedded Linux, Hardware, Software, and Interfacing*, Addison Wesley, 2002.

C. Negus, *LINUX Bible*, Wiley, 2008.

M. Palmer, J. Dent, and T. Gaddis, *Guide to UNIX Using Linux*, Course Technology, 2002.

R. Saikkonen, *Linux I/O Port Programming Mini—HOWTO*, Version 3.0, 2000.

M. Schimek, B. Dirks, H. Verkuil, and M. Rubli, *Video for Linux Two API Specification*, Revision 0.24, 1999. (GNU Free Documentation License)

SOURCES

dsPIC30F2010 Microcontroller and TC1263–5.0 VAT linear regulator
Microchip Technology, Inc. | www.microchip.com

AT91SAM9260 Development board
Olimex | www.olimex.com

MATLAB
The MathWorks, Inc. | www.mathworks.com

by Larry Cicchinelli

Calibrated Decibel Meter Design

You can build your own calibrated decibel meter with a little know-how and four main parts: an audio amplifier, a meter circuit, a display, and a power supply. With this design, you can accurately measure the output level of most audio systems.

I have been interested in measuring various electrical properties since my earliest days working with ham radio. My profession for over 30 years was to design and implement test equipment, and I have been working with public address systems of one sort or another for almost as long. The project described in this article is an outgrowth of both of these experiences.

I built a calibrated decibel meter to accurately measure the output level of almost any audio system. The system I finally implemented contains four parts: an audio amplifier, a meter circuit, a display, and a power supply (see Photo 1). I actually built two versions, which I will detail after I describe the circuits. Figure 1 shows the first system's wiring. Figure 2 shows the second system's wiring.

dB VS. dBm

Let me start off with a few comments about decibels (dB) and decibels referenced to 1 mW (dBm). Whenever a specification talks about "dB," it is a relative value referencing gain, which can just as easily be positive or negative (loss). When the discussion is about "dBm," or any other "dB" with a third letter, then there is an absolute involved. In the case of "dBm," the reference is 1 mW to a known resistance—usually either 500 or 600 Ω for audio systems and 50 Ω when discussing RF systems. For this system, I chose a reference of 500 Ω. This yielded 0.707 V_{RMS}, or 2 V_{PP} for 0 dBm.

AUDIO AMPLIFIER

The audio amplifier contains four gain stages, three of which have selectable gain: 0/40 dB, 0/20 dB, and 0/10 dB (see Figure 3). The total gain of the subsystem is selectable in 10 dB steps from 0 to 70 dB. This is sensitive enough for most microphones. I did not implement phantom power in my system, but it would be easy enough to add. One method I've used involves inserting a transformer with a center tap on the primary. The center tap would be connected to the phantom power supply line. Most professional systems use 48 V, but the 15 or 23 V available in this system may work for you.

In dealing with the gain values used in this subsystem, the ratio of the resistor values is more important than

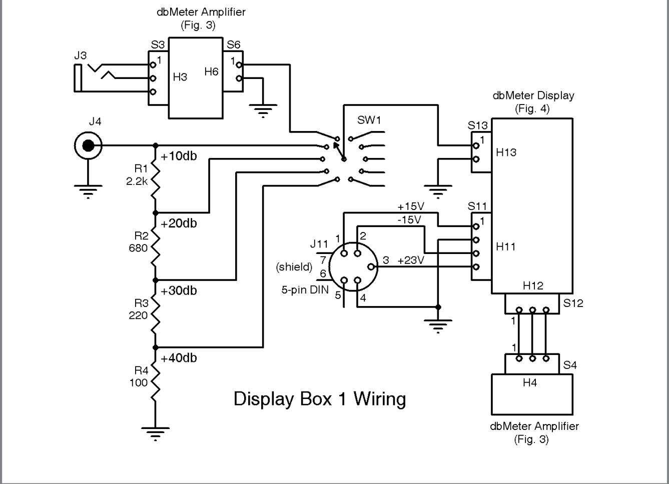


Figure 1—This is a wiring diagram of the unit (System 1) with the amplifier and attenuator.

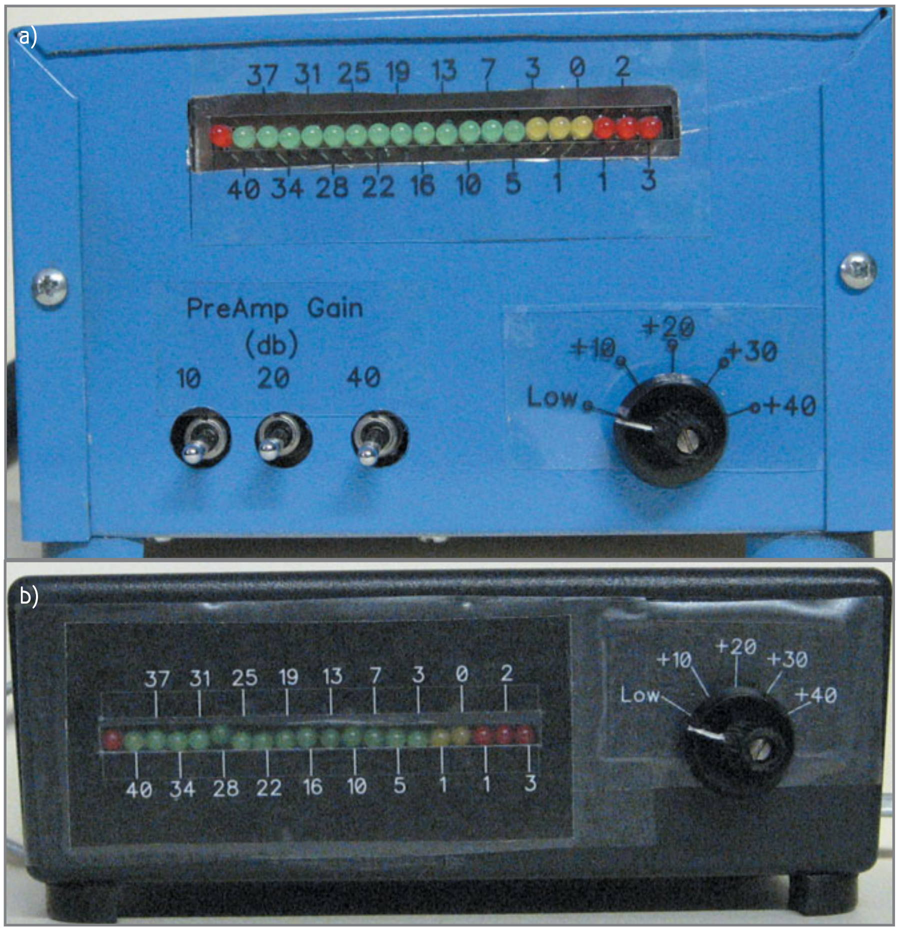


Photo 1a—System 1 is the unit that has both the selectable gain amplifier and a step attenuator. b—System 2 is the unit with just the attenuator.

their absolute values. Also, most of the resistors I used were 1% since they cost the same as 5% resistors; however, even 5% values would yield acceptable results for most applications. Just to prove this with numbers, let's calculate the gain of the 40-dB stage using one set of "worst-case" 5% values. Let R6 = 950 Ω (5% low) and let R8 = 105 kΩ (5% high). The gain of the stage would then be 106K/950 = 111.6, which is equivalent to almost 41 dB. For most applications this would be quite acceptable.

All of the op-amps in the subsystem are National Semiconductor LM833s, which I've used in various audio systems for the past 30 years and find that their specs hold up well. The first stage has 0-dB gain and simply converts a balanced signal to an unbalanced one. The input signal can just as easily be an unbalanced signal. The second stage has a selectable gain of either 0 or 40 dB. I wanted to put the highest-gain circuit first so that a low-level signal would get amplified early in order to maintain the signal-to-noise ratio.

The remaining two selectable gain stages are similar to the first. In each of the three gain stages, the lower value gain is enabled by shorting out the higher value resistor in the feedback loop. The switches I used were SPDT (because I had them on hand), but SPST would do just as well. Each gain stage has provision for capacitors in the feedback loop. I found that these were necessary in order to eliminate oscillations due to the high gain values. I designed the circuits so that each can have a capacitor, but you may find that the lower gain stages do not need it. With the specified values, the system has a bandwidth of at least 20 kHz.

Note that both the 40- and 20-dB stages have offset potentiometers. These allow you to adjust for the input offset voltage of the op-amps. They should be adjusted with no signal input and with the higher gain selected for the stage being adjusted. You should do one stage at a time: select the higher gain value and adjust the potentiometer for 0 V at H6.

The last op-amp in the circuit provides another 10 dB of gain for driving the display amplifier. I wanted a signal with a higher amplitude than 0 dBm to drive the rectifier circuit. With all three gain switches set to 0 dB, a 0-dBm signal applied to the input will present a 10-dBm (2.23 V_{RMS}) signal to the input of the display amplifier.

Notice that there is an "extra" op-amp stage at the output driving H7. This is an auxiliary output that can be used to drive whatever you may want, perhaps a set of headphones, without affecting the main output. The two resistors in the circuit (R25 and R26) set the gain to 2. However, you can just as easily change the values to whatever you think appropriate—within reason! One way to do this would be to replace R26 with a variable resistor which would allow you to adjust the gain. If you do this, I suggest that you include a fixed resistor as well in order to keep from

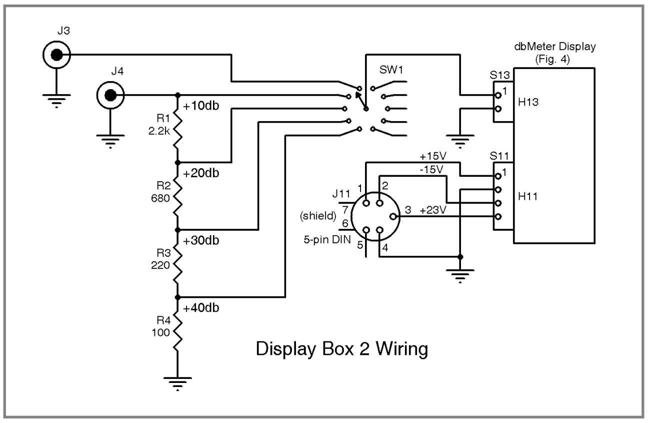


Figure 2—This is a wiring diagram of the unit (System 2) with just the attenuator showing the provision for having the amplifier as a separate unit.

Elektor Shop

The world of electronics at your fingertips!

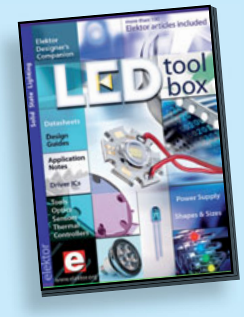
CD/DVD-ROMs

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. The DVD also contains (free of charge) the entire 'The Elektor Datasheet Collection 1...5' CD-ROM series, with the original full datasheets of semiconductors, memory ICs, microcontrollers, and more.

ISBN 978-0-905705-76-7 • $111.30



More than 100 Elektor articles included

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.

ISBN 978-90-5381-245-7 • $46.00

OBD2 Analyser NG

The compact OBD2 Analyzer in the June 2007 issue was an enormous success — not surprising for an affordable handheld onboard diagnostics device with automatic protocol recognition and error codes explained in plain language. Now enhanced with a graphical display, Cortex M3 processor and an Open Source user interface, the next generation of Elektor's standalone analyser sets new standards fora DIY OBD2 project. The OBD2 Analyser NG is self-contained and can plug into any OBD diagnostic port.

Kit of parts including DXM Module, PCB SMD-prefitted, case, mounting materials and cable

Art.# 090451-71 • $135.50

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

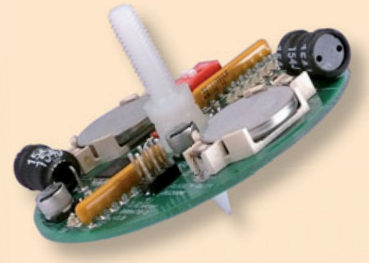
Elektor US
4 Park Street
Vernon CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: sales@elektor.com

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



LED Top with Special Effects

If you fit a line of LEDs on a circular PCB and power them on continuously, they generate rings of light when the board is spun. If you add a microcontroller, you can use the same set of LEDs to obtain a more interesting effect by generating a 'virtual' text display. The article also describes a simple technique for using the Earth's magnetic field to generate a synchronisation pulse. The potential applications extend from rotation counters to an electronic compass.

Kit of parts incl. SMD-stuffed PCB and programmed controller

Art. # 080678-71 • $67.80

Kits & Modules

adjusting to an infinite gain!

DISPLAY DRIVER

The display driver (see Figure 4) uses LM3916 and LM3915 devices to drive the LEDs. The LM3916 has outputs that are close to those of the standard VU meter markings: +3, +2, +1, 0, –1, –3, –5, –7, –10, and –20 dB. The only one missing is the –2-dB indication. This subsystem does not use the –20-dB output of the LM3916. The LM3915 is used to extend the range to –40 in 3-dB steps starting at –13 dB.

I found that the rectifier circuit I used has excellent linearity from close to 0-V input up to a signal which yields about a 7-VDC output. Because of this, I calculated R2 and R3 such that the comparators in the two LED drivers have close to a 7-V reference value. This is the voltage necessary to turn on the highest LED. The resistors that control the gain of the rectifier (R6, R7, and R8) were chosen so that R7 can be adjusted to display 0 dB with a 10-dBm input signal. One easy way of calibrating the unit is to apply –1.7 VDC to the junction C4, R6, and R12 and then adjust R7 until the 0-dB LED turns on. This may get you close enough so you don't need to use a known audio level.

The rectifier circuit and the gain circuit of IC3B were both derived from the LM3916's datasheet. The reason for the 16 dB gain for IC3B is so that when a –3-dBm signal is presented to the rectifier it will cause the LM3915 to turn on its most significant LED

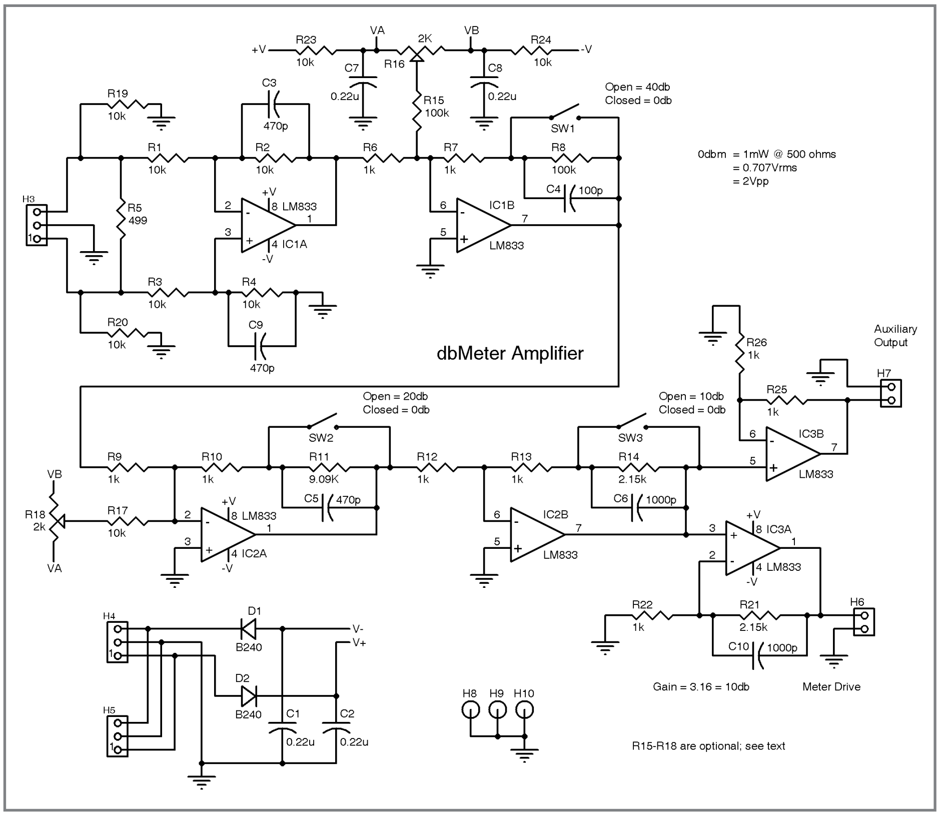


Figure 3—This is a schematic of the amplifier's circuit board with selectable gain from 0 dB to 70 dB in 10-dB steps.

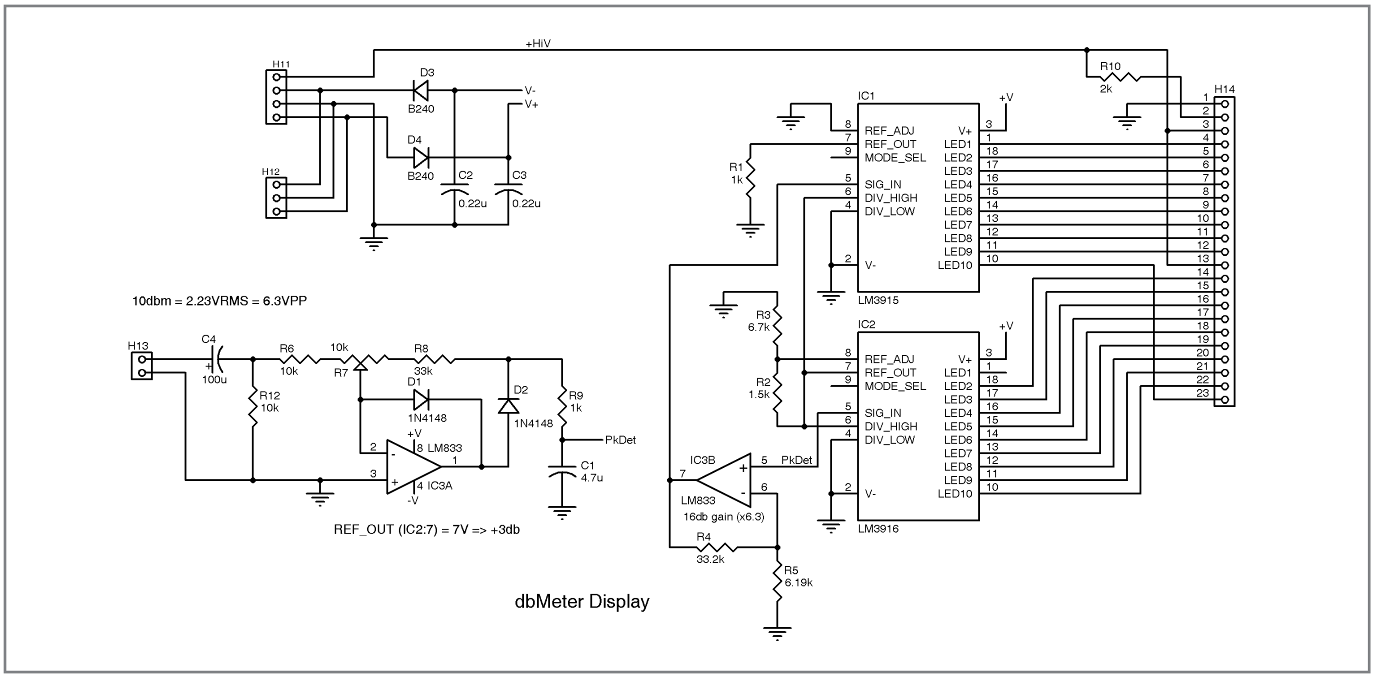


Figure 4—This is the audio rectifier and display driver.

(see Table 1). Notice that there is 16-dB difference between the top LEDs of the two devices.

LED BOARD

The LED board is nothing more than a small PCB for mounting the LEDs (see Figure 5). At one time, I thought about using LED bar arrays, but they are expensive and it is hard to find them with appropriate color patterns. I designed the display driver and LED boards so they can be connected using a right angle pin header with 0.1″ spacing. You can solder the header onto both boards or solder the header to one board and use a socket on the other. I used a right angle header on the LED board and a socket on the display board. I suggest you mount the connectors first without soldering so you can see how they will assemble together.

Be careful when installing the LEDs. You need to ensure that you install them in the correct order as well as orientation. On the LEDs specified in the BOM, the cathode lead has a more pronounced bend. Theoretically, you can install the LEDs on either side of the board. I installed them on the front—the side without the copper traces. Note that there is a symbol etched in copper showing the LED orientation on both sides of the board. Also, the position of LED20 (the power indicator) is etched on the bottom side; this will make it easier for you to determine which color LEDs go where. See Figure 6 for System 1. You can see in Photo 1B that I replaced D6 with a green LED simply because I ran out of orange ones (see Figure 7).

	Signal Level (dBm)	Indication (dBm)
	13	3
	12	2
	11	1
	10	0
	9	–1
LM3916	7	–3
	5	–5
	3	–7
	0	–10
	–3	–13
	–6	–16
	–9	–19
	–12	–22
LM3918	–15	–25
	–18	–28
	–21	–31
	–24	–34
	–27	–37
	–30	–40

Table 1—This table shows the signal levels into the two display drivers and the LED indication values.

SYSTEM POWER

This subsystem develops three DC voltages: 15 V, –15 V, and 23 V (see Figure 8 and Figure 9). The LED drivers have two modes of operation. One mode essentially puts the LED strings in series for each of the drivers individually. The other mode has each LED drawing its current separately. I chose the first mode because it enables both strings of LEDs to draw a constant current as long as at least one LED is on and should cause less noise. However, this mode requires a supply voltage high enough to handle 10 LEDs in series.

In the unit I built, the 23 V is actually developed by stacking an 8-V regulator on "top" of the 15 V. Each regulator gets its input from a half-wave rectifier. The regulator subsystem is powered by a 24-VAC "wall wart." Anyone building the power supply can just as easily use a 24-V regulator instead the of stacking method I used. There is a jumper (H3) on the board that allows you to select either method. Connecting pins 1 and 2 is for a single 24-V regulator. Connecting pins 2 and 3 is for stacking an 8-V regulator on top of the 15-V regulator. You do not have to install a header in this position. You should just insert a wire between the appropriate points and solder it in place.

All of the regulators are linear because I did not want the noise associated with switchers. I also built the regulators in a separate box because I did not want the 120 VAC in the same box as a system capable of 70-dB gain.

SYSTEM ASSEMBLY

The first version I built (see Figure 6) has all circuits, except the power supply, in one aluminum box. This system can be used to measure low-level signals with a practical lower limit of about –90 dBm. If you try to use the maximum available gain of the system you will probably have quite a few LEDs lit just due to noise. I have found that I can reliably use the system with up to 60-dB gain selected.

The system has two inputs: low level and high level. The low-level signals are routed to the 70-dB amplifier while the high level signals are routed to the voltage divider. The five-position switch allows you to select which of the two sources are being measured. With this arrangement, you can measure signals from about –90 to +40 dBm (70 V_{RMS}) referenced to 500 Ω—a range of 130 dB.

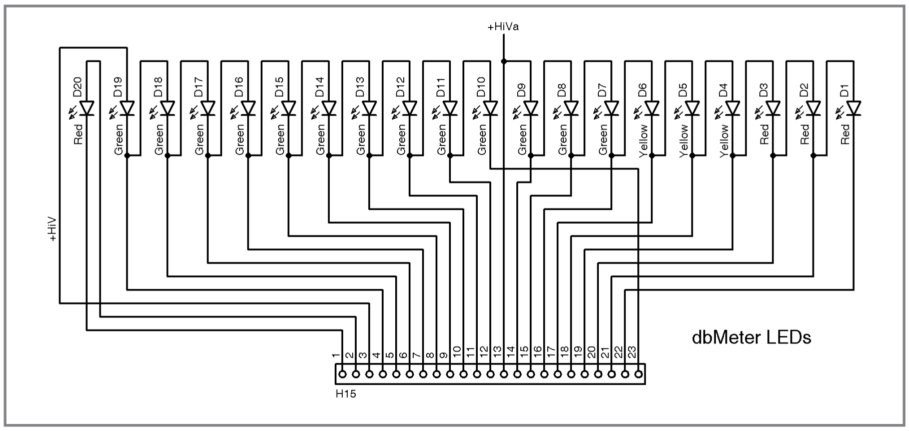


Figure 5—The LED circuit board

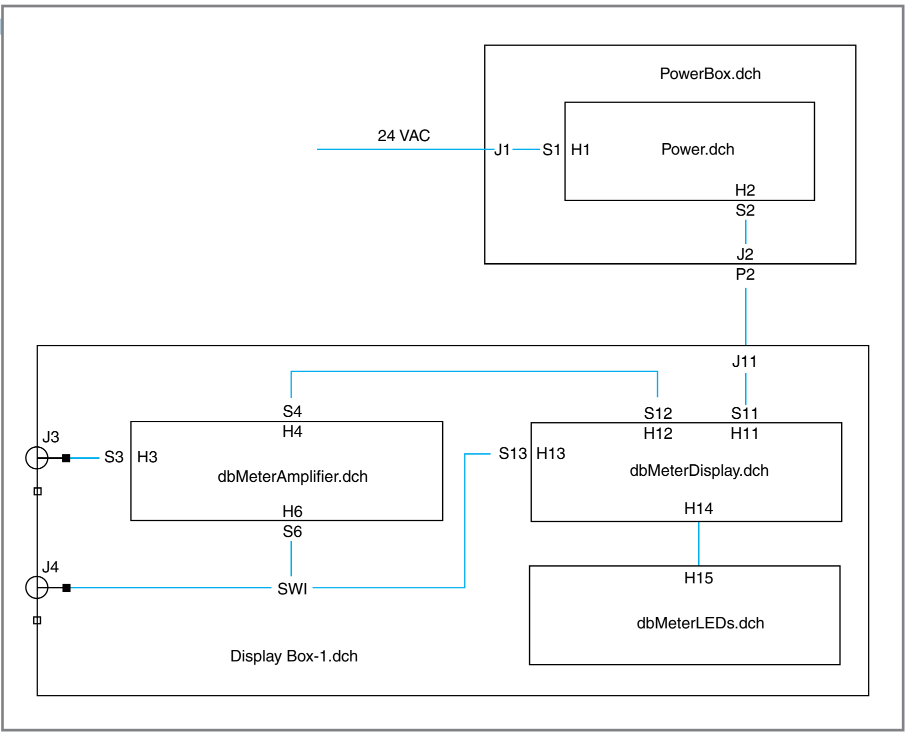


Figure 6—The unit (System 1) with the amplifier and attenuator.

Some care needs to be used in selecting the resistors if you want to measure the high-level signals. The total resistance is 3.2 kΩ, and with 70 V across, it there will be about 22 mA through each of the resistors. This means that R1 will have to dissipate a little over 1 W. If you implement this circuit, I strongly recommend you use a 3-W resistor—just to be on the safe side. Also, R2 will dissipate about 0.3 W, so a 1-W resistor would be a good idea. The remaining two can be 0.25 W.

Figure 7 does not have the 70-dB audio amplifier and can be built in a smaller enclosure. It has the capability of directly measuring the aforementioned higher-level signals. It also has two input connectors. One is connected directly to the attenuator while the other is connected to the switch. This scheme allows you to build the 70-dB amplifier in a separate enclosure and connect its output to the second input.

You can also use this system to measure signals which are not based on 500 Ω. Suppose you want to monitor the output of an amplifier driving an 8-Ω load. All you have to do is calibrate the system for the appropriate voltage values. As designed, 0 dB will be indicated when 2.23 V_{RMS} is applied to the input of the display driver. This is about 0.6 Ω, which is 28 dBm into 8 Ω. However, 30 dBm is equivalent to 2.83 V_{RMS} into 8 Ω. The adjustment range of the circuit will easily allow you to set this as the "0 dB" indication. You can then use this circuit, without modification, with calibrated ranges of 30, 40, 50, and 60 dBm into 8 Ω. An interesting point to note is that 30 dBm is equivalent to 0 dBW – a signal level referred to 1 W. What that means is that the display will show values of 0, 10, 20, and 30 dBW.

FILES

Several file types are included: .dch (the DipTrace schematic), .dip (the DipTrace PCB), .gbr (Gerber), .drl (drill), and .xls (BOM). The main file names should be fairly obvious as to which circuit they apply. The Gerber files and drill files can be sent to a PCB manufacturer to make the boards. I have had good success with this for several of my designs using this software. You can also use the PCB files and print them full size if you want to etch your own. Although I have not made the SilkScreen files, you can do so from the PCB files. There is a free version of DipTrace available on their website.

The dbMeterBOM.xls file on the *Circuit Cellar* FTP site is a combined BOM of all four circuits individually as well as a consolidated BOM. However, I did not add the quantities in the consolidated BOM. The FrontPanelLabels.dip file has the front panel labels for the LEDs and the five position switch I used. This was created using the DipTrace printed circuit program. By using various Print options, I was able to print both a negative and a positive image, as can be seen on the pictures of the two systems. The systems work as expected, but my construction skills are not the best.

The headers and mating

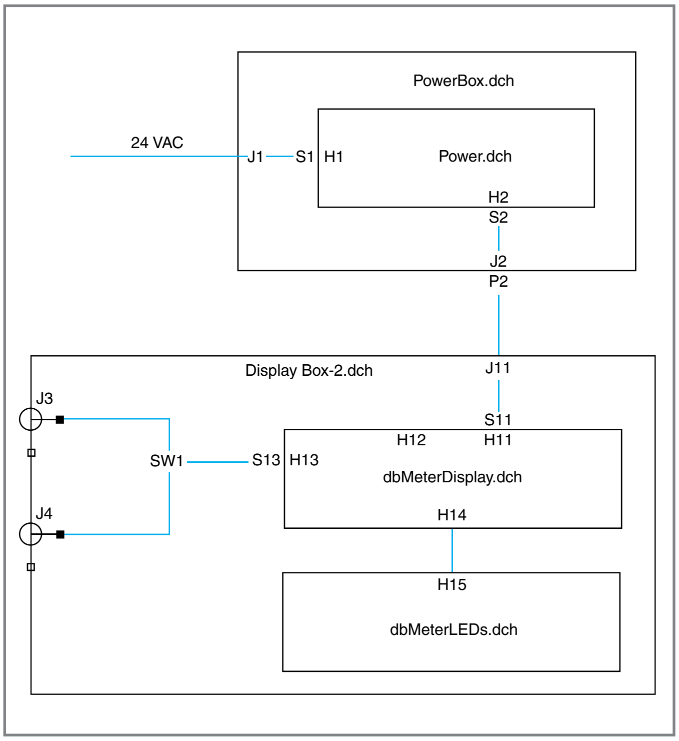


Figure 7—This is the unit (System 2) with just the attenuator.

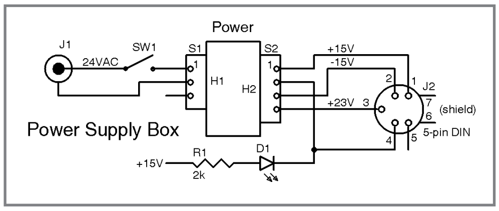


Figure 8—Power supply wiring diagram

the connection between the display board and the LED board. I've used this method of connection for most of my projects. It's been quite useful and worth the extra cost—about $0.16 per connection. I use a small pair of needle nose pliers to crimp the wire into the socket pins while

connectors were built using four parts from Jameco: a pin header (160882), pin header housing (103158), socket pins (100766), and a socket (200783, only if you use this socket on the display driver board). The total number of connections using these in System 1 (see Figure 6) is approximately 35—not including viewing the process using a 3× magnifying lens. A crimp tool would be a lot easier, but it would be expensive. You do not need to use this method. You can just as easily insert wires into the pad locations and solder them.

Feel free to email me if you have any questions about this project.

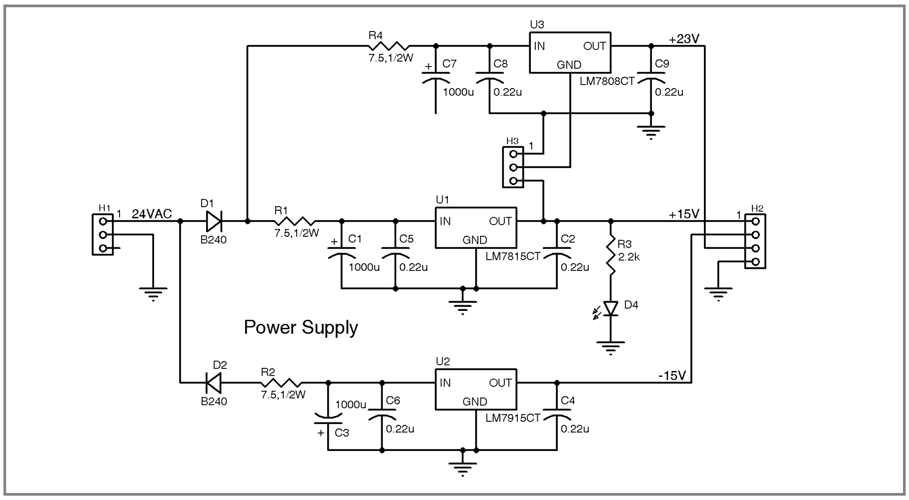


Figure 9—Power supply schematic

Larry Cicchinelli (k3pto@arrl.net) holds a BSEE from The Drexel Institute of Technology and an MSES from Pennsylvania State University. He has been a technical support manager at Digi International (Rabbit Brand) since 2000. From 1967 to 2000, Larry worked for Ford Motor Company. He has been licensed as K3PTO since 1961.

PROJECT FILES

There is no code associated with this article. To download additional project files, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/236.

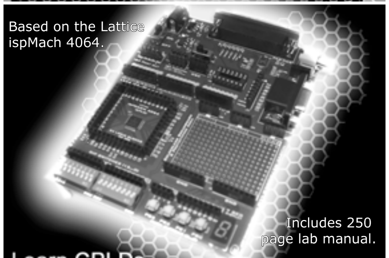
RESOURCE

National Semiconductor Corp., "LM3916 Dot/Bar Display Driver," DS007971, 2000, www.national.com/mpf/LM/LM3916.html.

SOURCES

LM3915/LM3916 Display drivers and LM833 op-amp
National Semiconductor Corp. | www.national.com

MACH64
PROGRAMMABLE LOGIC STARTER KIT
NURVE NETWORKS LLC



Based on the Lattice ispMach 4064.

Includes 250 page lab manual.

Learn CPLDs the fun way with the MACH64! This complete kit comes with everything you need to take you from mystery to mastery with CPLDs and programmable logic.
Learn to turn *software* into *hardware!*

www.XGAMESTATION.com

PIC-SERVO
MOTION CONTROL

MOTION CONTROLLERS FOR BRUSH, BRUSHLESS AND STEPPER MOTORS.

- controller chips
- controller boards

www.picservo.com
JEFFREY KERR, LLC

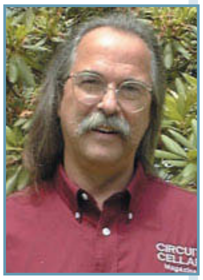
Add **USB**
to Your Designs

Chips, code, protocols, embedded hosts, wireless options, debugging, **USB 3.0** and **SuperSpeed** too!

USB Complete
The Developer's Guide
Fourth Edition
Jan Axelson

ISBN 978-1-931448-08-6 $54.95
Lakeview Research LLC **www.Lvr.com**
By the author of *Serial Port Complete*

by Jeff Bachiochi



Sun Tracker (Part 2)

Power Up and Start Tracking

The first part of this series detailed a custom time-keeping design called a "sun tracker." Now it's time to power up the tracker, start keeping time, and display the data on an LCD.

Sunrise, sunset. Sunrise, sunset. Today is just another day in the life of our universe, a blip in the endless orbital calendar of one insignificant planet. I don't like the thought of being insignificant, but as we peer deeper into the universe, life as we know it becomes an infinitesimally small part of the bigger picture. We can only hope to preserve our precious existence by not squandering the natural resources we have been given to work with. Creating landfill in support of a small percentage of inhabitants seems like poor resource management. Every so often, I'm guilty of having the "I, me, mine" mentality, but I long to live more positively. I want to be part of the solution, not part of the reason for our extinction.

Each person's personal frame of reference isn't an eternity; it's a lifetime. For the average person, that lifetime is defined by days, numbering about 30,000. Our ancestors rose with the break of day and rested with the setting sun. But today, technology continues to change how each successive generation lives. Devices like PDAs enable us to break up our days into bite-sized chunks of time. And so it goes, we are captives of time.

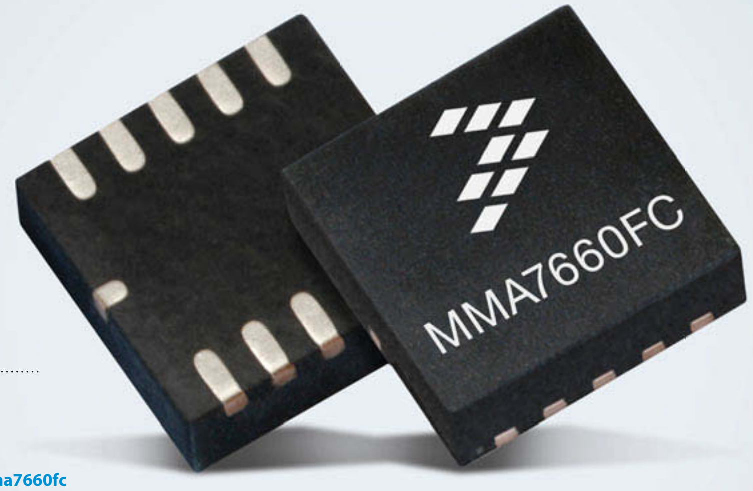
Last month, I began a project that, at first glance, may seem like a waste of (drum roll) time (cymbal crash). But if you look deeper, you will find a lot that is applicable to other projects you might consider. It started with the design of a homemade quadrature light sensor based on a tiny Intersil ISL29102 SMT device. This 3.3-V module has four voltage outputs based on the amount of light received by each of four sensors placed around a shadow-producing post. If the post design is correctly, when pointed directly at a light source (on-axis with the sun), each sensor is slightly kissed by the edge of the post's shadow. As the sensor deviates from on-axis, the post's shadow will encroach further on one or more sensors, reducing their output.

The sensor module is mounted on a pan-and-tilt unit made from RC servos (see Photo 1). A single microcontroller measures the light sensor module's outputs and drive the position of the x and y servos. The application attempts to keep the light sensor module pointed at the brightest object in the area, in this case the sun. At the end of Part 1, I discussed using the hardware PWMs to drive the servomotors. Based on the x-servo requirement to pan a complete 360 degrees, the on-time pulse width of the servo's 20-ms cycle must vary from 0.5 to 2.5 ms. The usable portion of a 0- to 2,500-µs time base is 80% (500 to 2,500). Twenty percent of the 10-bit resolution would be wasted giving a useful range of 1,024 × 0.8, or approximately 819 counts to cover 360 degrees: 360°/819 counts is 0.43°/count. This strategy would also require operating the PWMs in one-shot mode to increase a 2,500-µs cycle time to the 20 ms required. If we based the PWM on 20 ms, the usable portion of a 20-ms time base is 10% (20 ms/2 ms). Ninety percent of the 10-bit resolution would be wasted giving a useful range of 1,024 × 0.1 (or approximately 102 counts) to cover 360 degrees, 360°/102 counts or 3.53°/count.

You know that the sun travels 1° in 4 minutes (24 hours × 60 minutes per 360°), so you need at least 0.25°/bit resolution to achieve a

The Newest Products For Your Newest Designs

We're committed to Engineers.

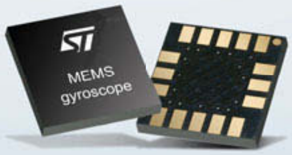


freescale semiconductor

MMA7660FC 3-Axis Accelerometer
www.mouser.com/freescale_mma7660fc

Panasonic Panasonic Electric Works Corporation of America

NaPiOn MP Motion Sensors
www.mouser.com/panasonicew/a



ST

Single- and Two-Axis MEMS Gyroscopes
www.mouser.com/ stmemsgyroscopes

OMRON ELECTRONIC COMPONENTS

Micro Tilt Sensor
www.mouser.com/omron/a

WARNING: Designing with Hot, New Products May Cause A Time-to-Market Advantage.

With the newest motion sensing products and technologies Mouser is committed to Engineers. Experience Mouser's time-to-market advantage with no minimums and same-day shipping of the newest products from more than 400 leading suppliers.

MOUSER ELECTRONICS® a tti company

www.mouser.com (800) 346-6873

Pololu
Robotics & Electronics

Robot Kits
Line followers
Robot arms
Chassis

Save 10%
With coupon code
CCLLR3PIJ

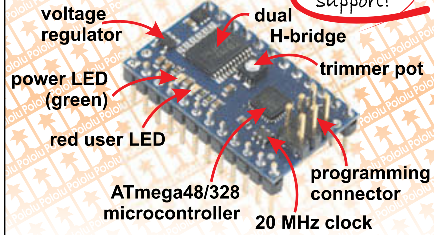
3pi Robot
$99.95

High-performance, C-programmable, ATmega328P-based robot (with Arduino support)! Items #975, #1306

Mechanical Components
Motors, servos
Wheels, ball casters

Motion Control
Motor controllers
Servo controllers
Now with USB!

Robot Controllers
with Arduino support!



Solder Paste Stencils
Use our low-cost solder paste stencils to quickly assemble your surface-mount designs.
From $25

Custom Laser Cutting
From $25

Cut your own custom chassis, front panels, and more!
1-877-7-POLOLU
www.pololu.com
3095 E. Patrick Ln. #12, Las Vegas, NV 89120

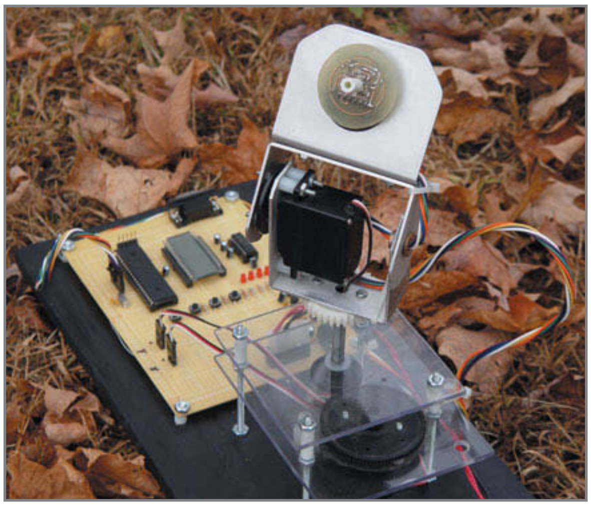


Photo 1—Here's the finished project out in the yard, where it's happily awaiting the point of maximum azimuth, the mid-day sun (noon).

design goal of determining the time of day from the sun's movement to within 1 minute. This means you cannot use the hardware PWM to drive the servomotors with the accuracy required. Let's get back to this after looking at more of the system design that will affect this timing.

RTC

This project's aim is to set the time of day. Let's face it; a working clock isn't too useful if it isn't able to tell the time with any degree of accuracy. For a while now, many microcontrollers have been including at least some amount of support for adding an RTC to your application. The obvious choice would be a microcontroller that contained the dedicated counter and register hardware that can keep track of time autonomously. This is based on a separate 32-kHz crystal input that can stay alive even when the microcontroller's main oscillator is turned off to conserve power. With the minimum hardware of only an extra 32-kHz oscillator, the user has to use software to create an RTC. While the microcontroller I'm using does have this extra oscillator input, I chose to use a 4.194304-MHz crystal running the whole system so I could use the internal 4× phase-locked loop (PLL) to reach a system clock of over 16 MHz while still retaining a value that divides evenly for a RTC (see Figure 1).

Timer0 is used to produce an interrupt once every 16-bit rollover, 1/64 second. To keep any interrupt routine to a minimum, this routine simply keeps track of a counter setting a `Tic` flag every 64 interrupts. This flag is used in the main loop to call `DoNewSecond` once every second. I implemented a simple hours, minutes, and seconds routine to keep track of the time of day in military 24-hour format. When enabled, the hour and minute is updated to the display every second.

DISPLAY

Many of the seven-segment LCDs available are static (nonmultiplexed) displays—that is, they have one common with many segment connections. These are fine when you have plenty of I/O pins available, but cannot be used when you have limited I/O. For instance, to display time using a four-digit display requires 7(hour:tens digit) + 7(hour:units digit) + 7(minute:tens digit) + 7(minute:units digit) + 1(colon) = 29 segments plus one common for a total of 30 outputs. A multiplexed

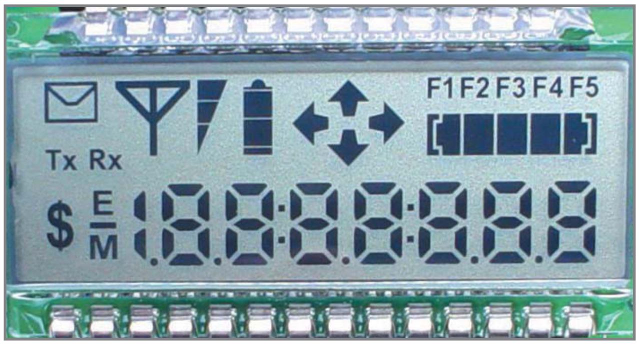
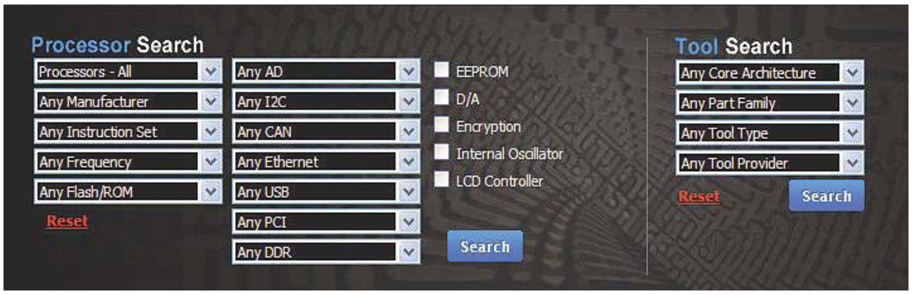


Photo 2—This LCD has lots of little glyphs, like arrows and such, besides the 7.5 digits. Looks like this may have been originally intended for a cell phone. Remember when cell phones didn't have graphic displays?

Pick a Chip. Any Chip.

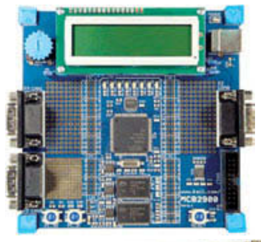
Find a Solution to your next Embedded Challenge.
Do the Research you should, but never had time for.

Embedded Developer's intuitive research engine helps you speed your chip evaluation time. You don't have to know the manufacturer, chip family or part number--just select the features you want and let us do the rest.



Part Number	AT91SAM7X	MCF5208	LPC2923
Manufacturer	ATMEL	freescale semiconductor	NXP
Core Variant	ARM7TDMI	ColdFire V2	ARM968E-S
Flash	262144	0	262144
RAM	65536	16384	16384
Max. Freq.	55	166	125
Dhrystone MIPS	50	159	156
Timer Bits	16	32	32

We help you research your best option. Nowhere else can you compare your best options side-by-side from different manufacturers. Click on the device you want, and a product page lets you select Distributor Buy / Quote options, send RFQs, download datasheets, and more. Plus--Hearst stock check gives you up-to-date inventory on every device.



Once you have the chip that meets your needs, review and compare the hardware and software development tools that support it from multiple manufacturers, and buy them on-line through our shopping cart.

Shave days off your schedule with Embedded Developer, the only site in the world where you're only clicks away from finding the chips and tools to get you up and running, quickly. Try EmbeddedDeveloper.com, or EmbeddedDeveloper.cn in Chinese.

EMBEDDEDDEVELOPER.COM
FIND. COMPARE. BUY.

EMBEDDEDDEVELOPER.CN
查询. 选型. 采购

The Sites for Engineers with a Job to Do.

Free SpYder Discovery Kit with an annual subscription to Elektor!*
www.elektor.com/cc

Special Offer to Circuit Cellar readers!

Keep $68.50 in your pocket!

Subscribe to Elektor now!

- Creative, fresh and unusual articles since 1961
- Electronics projects to delight and educate in diverse areas of electronics
- Each project is built, tested, and approved by Elektor's laboratory staff
- Access to a wide range of products to support magazine projects in our online shop
- Cheaper than 11 issues from the newsstand: Save **57%** of the cover price of $92.45

The SpYder Discovery kit from Freescale (valued at $16.00) comprises a USB programmer / debugger BDM, a CD-ROM with a special version of CodeWarrior, and one MCS9S08 8-pin PDIP microcontroller.

You save:

	$79.50	10 Magazines
+	$12.95	1 Double Issue (July/August)
+	$16.00	SpYder Discovery Kit
=	$108.45	**Total value of individual items**
–	$39.95	**Elektor year subscription***
=	**$ 68.50**	**Your saving**

* Offer limited to stocks and available in US and Canada only. Please allow four weeks delivery time.

Take out your subscription now:
www.elektor.com/cc • Phone 860-875-2199

display of 4 digits can reduce this to 7(digit) + 1(colon) = 8 segments plus four commons for a total of 12 outputs (see Photo 2).

Using a microcontroller that has an integrated LCD peripheral will generate the on-chip multiplexing signals necessary to directly handle the display properly to prevent damage. LCDs must be driven by an AC signal. Prolonged DC operation will cause electrochemical reactions inside the display, significantly reducing its life. This may begin to show as a "fuzzy" appearance of some segments.

A static display can be simply driven with an alternating signal (approximately 30 to 100 Hz) of VSS and VLCD on the common line with each segment driven in sync (segment off) or out of sync (segment on). A multiplexed display requires additional voltage levels, in addition to VLCD and VSS, based on the number of commons used. When using four commons, two additional levels are required: two-thirds VLCD and one-third VLCD. The Microchip Technology PIC16F1934 microcontroller used in this project has an LCD peripheral and can create the necessary bias voltages internally (or externally).

The 'F1934 can drive a total of 24 segments, with up to four commons. When four commons are used, each segment actually controls four LCD glyphs, one during each of the four common phases. Each common has a three-register bank (24 bits) associated with it. Each segment owns 1 bit in each of the four common's register banks. These bits are used by the LCD hardware to determine the control signals necessary to apply a maximum voltage (glyph on) or minimum voltage (glyph off) between each segment and common over one timeframe period. For more information on how multiplexing signals are created, refer to the PIC16F1934's datasheet or review my article about "Demystifying LCD MUXing" (*Circuit Cellar* 108, 1999).

I had some difficulty finding a multiplexed LCD that could display hours and minutes with a colon. The combination of multiplexed LCD and with a colon was the tricky part. I found an inexpensive display from MicroController Pros Corporation. The SBLCDA4 26-pin display has a bunch of other useful glyphs beyond the digits I am interested in. This LCD has 22 segment connections and four commons (26 pins). In the microcontroller, each bit of the 12 segment registers (four banks of three registers each) corresponds to a single potential glyph on the LCD. This project is using 40 bits (i.e., 10 × 4) out of the 96 possible.

Here's the tricky part. The unused segments can't just be grounded or tied to VLCD because their glyphs are being actively driven by the commons. This will not only apply a varying voltage across them, causing them to flicker, but also there will be a DC component causing eventual failure. To prevent this from happening, all segments must be driven. Since my I/O allotment for the display is 14, one segment is set aside to drive all the unneeded glyphs. They will be all driven as off by the same output.

The 10-segment outputs are used as follows: two segments for each digit (digits 7:4) plus a segment for glyphs F1:F4 and a segment for all other glyphs. You can see these connections in Figure 2. After setting up the LCD peripheral, the user only needs to set or clear the particular glyph bit in the appropriate segment register to turn that glyph on or off. The microcontroller handles driving all of the common and segment outputs to their proper levels. Note: This LCD operates from 2.7 to 3.6 V. Make sure you set up the LCD

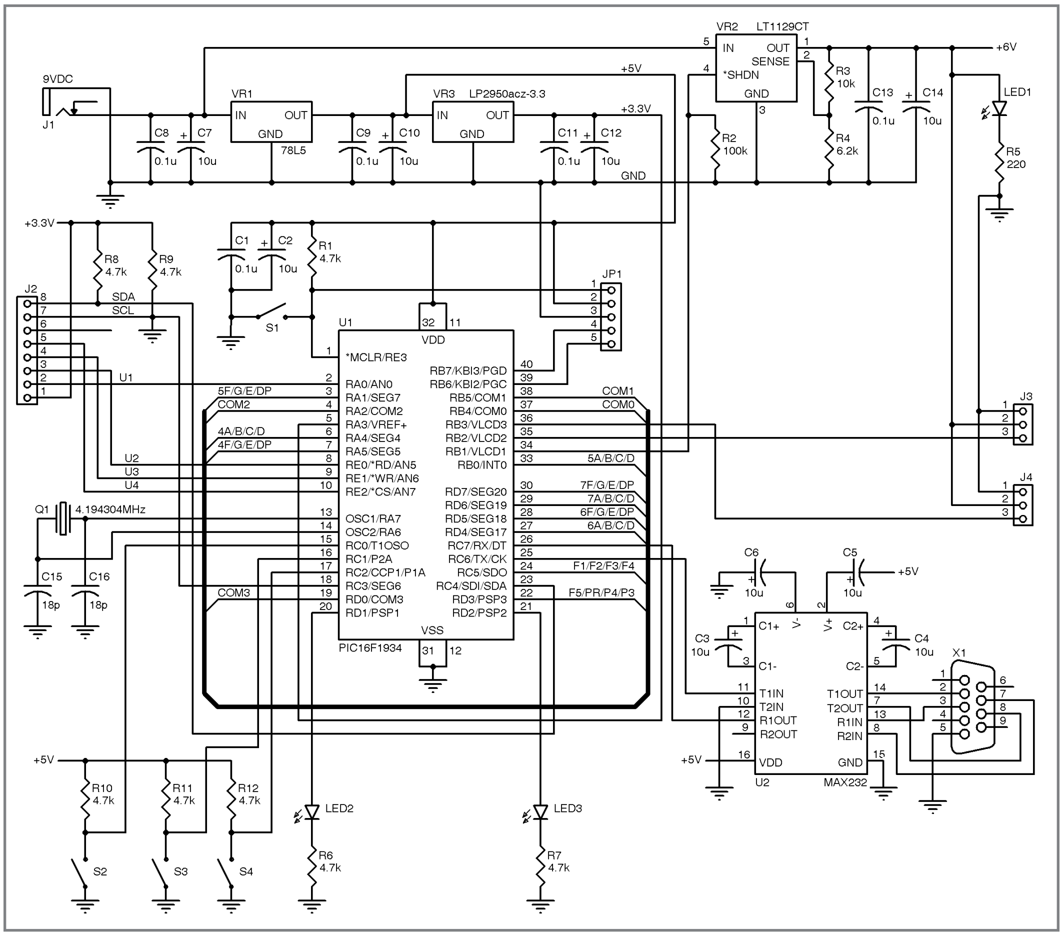


Figure 1—The schematic shows what looks like a random selection of segment drivers. However, these were left over after selecting the on-chip peripherals, UART, I^{2}C, A/D w/external reference, and external crystal connections.

peripheral correctly so it does not overdrive the device.

UI

The unused I/O is shared between push buttons and LEDs. These are used to enable you to change functions and display modes of operation. Along the development road, I've used these to provide feedback for various functions. For instance, the LEDs have been used to show PWM signals, various interrupts, and routine in-process indicators. These are outputs that can serve as visual indicators or they can be measured with a scope. This is great feedback for determining that you've calculated things correctly.

As they stand now, LEDs 2:3 display the present direction of the pan and tilt movement. They will be most active during the sun-following phase, continuously fidgeting around the optimum position. LED1 is tied to the servo power indicating when the servos are energized. Besides reset, other functions initiated by the push buttons include the ability to force a "search for the sun" routine and a "find noon" routine.

PWM

While there's a strong interest in using the hardware PWM, we've seen that it just doesn't have the resolution required here. I used Timer1 for this operation. Remember that the main crystal oscillator is 4.194304 MHz with a 4× PLL. FOSC is therefore 16.777216 MHz. Timer1 is one of the few peripherals that can use FOSC as its clock source. This gives approximately 60-ns resolution with an almost 4-ms maximum rollover (16 bits). This fits within the on-time window that we need 0.5 to 2.5 ms. Within this 4-ms window, the on-time count would be 0.5 – 2.5 ms/60 ns (or 2,000,000/60 = 33,333 counts). That would be 360°/33,333 counts = 0.01°/count. While this is a resolution of the sun's movement to under 3 s, we can't expect this accuracy with the mechanical slop of the system, but it shows a theoretical possibility.

You may have noted a problem with this since the rest of the timers are only 8 bits and we have two servos. However, the requirements of the servos help solve this problem. The on time (max 2.5 ms) is only about one-eighth of the repetition rate (20 ms). Therefore, if you use another timer to give 10-ms interrupts, you can then alternate using Timer1 for servo X at the first 10-ms interrupt and for servo Y at the next 10-ms interrupt. Each will have a repetition rate of 20 ms.

The total interrupt time is kept very short and limited to two timers. In fact, I think I could have combined the two. Timer0 interrupts every 12.5 ms and only sets one second flag. Timer2 interrupts every 10 ms and sets the servo output, reloads Timer1, waits for the rollover, and clears the servo output. Since the servo timing is the only critical routine, it takes place totally as an interrupt. All other functions are considered noncritical and take place in the main loop.

BREAKING IT DOWN

You may have noticed an extra bit of hardware hanging off the microcontroller—a serial port. While it isn't actually used in the finished application, it served me well during development as a debugging tool for identifying what was happening inside routines. For instance, I could check on execution path decisions. This assured me that the code I had written was actually executing as intended. It helps to point out logic errors immediately instead of having to guess why the execution went wild. This is most helpful when performing calculations and branching on the result.

The application can be broken down into three routines: find the sun, follow the sun, and find noon. There is a simple prerequisite for each user: place the project in a level position and point it in a generally southern direction. Both of these can be handled in software but have not been implemented at this point. Let me explain. If you put down the unit, it is possible that part of the sun's path will be outside the servo's mechanical panning limit. Since this can be detected, software could swing the servo around 360° when necessary; but I thought this wide swing in the pan makes the prerequisite a better choice.

Following the sun is the most important routine in this application (and used by the others), so let's look at it first. There are three parts to the routine: ADCs, A/D channel result comparisons, and movement adjustments. Each of the four sensors are connected to the A/D inputs of the microcontroller, read in quick succession, and saved as 10-bit values. The 3.3-V source for the light sensors is fed into an optional positive reference input to the ADC and used (along with GND) as the voltage reference for the sensor's output voltage. Brighter light equals higher output voltages and thus higher conversion values. When the conversions have finished, each axis pair is compared to determine an error direction. On the x-axis (pan), if the left sensor (x-) is receiving less light than the right sensor (x+), then you get an x- error. A minus error requests the x-axis to move the sensor module in the x+ direction. In this case, pan toward the sensor that is brighter, which will (eventually) reduce the shadow on the x- sensor and increase the shadow on the x+ sensor. When a point of equilibrium is obtained, the axis is considered to be on track. The y-axis (tilt) is handled the same way.

The "follow the sun" routine should

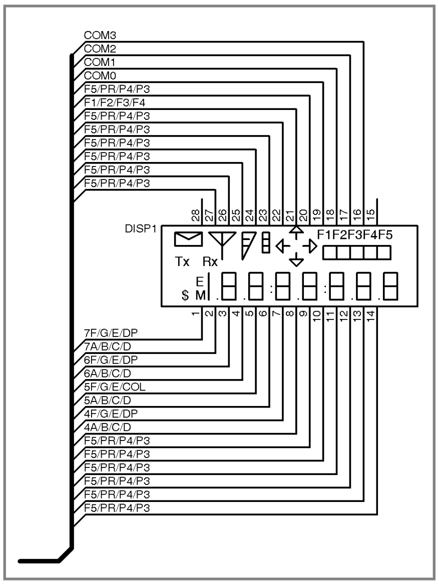


Figure 2—Of the 21-segments and 4 commons required to individually access all 84 glyphs on this LCD, I'm using only 14-output drivers. Note that one driver simultaneously drives 11 unused segments (OFF).

be sufficient to find the sun on its own, but there may be instances where it might not move at all or find some false (not the brightest) source. This routine pans the complete horizon for the brightest direction and then returns to that position. Then the tilt goes from horizon to overhead, returning to the sun. If some minimum light output isn't seen, it can go to sleep for a while and try again in an hour (assumes night).

Once the sun has been found and can be tracked, you just need to find noon to set the clock. The "find noon" routine continuously resets the time (to noon) whenever the light module requests a higher y-axis position. You need two flag bits, one to indicate that the y-axis is increasing over the last span of time (it's before noon) and a second to indicate when the y-axis is decreasing over the last span of time (it's after noon). The span of time for these bit flags must be outside of the normal dithering. If the y-axis hasn't risen, then it's too late in the day to find noon. In this case, you can sleep for at least 12 hours and try again. When both bits are found, the clock, which continues to tick from the last time noon was reset, can now display the correct time on the LCD. Up to this point, a blinking colon is all that has been displayed indicating an unknown time of day.

POWER

I haven't been overly concerned about power-up. Current for running everything but the servos is approximately 16 mA. I can cut that in half just by removing the RS-232. The servos, however, require around 150 mA each to operate. I've seen some high-torque gymnastics exceed 700 mA. I've included a 6-V regulator for the servos that includes a logic-controlled On/Off switch. This allows the servos to be disconnected when they are not needed. The servos seem to be happy remaining where they are when turned off. However, it doesn't take a whole lot of muscle to move them in that state. The sensor module is light enough to allow the pan and tilt to remain stationary with the servo power removed. A one square foot (5-W) solar panel, on the other hand, would need to be well balanced to remain in place unpowered.

I don't want to get too far off track here with a discussion of power. The intent of this project was to automatically detect the time of day and set an RTC from tracking the sun through the sky. While the route might be considered to be a bit over the top by some, the lessons learned along the way serve to educate and entertain. I get excited during these projects. Sometimes so much so that I fall into the engineering trap of continuously redesigning the ongoing project. I stalled a number of times here, thinking about solar tracking, the high currents of servos, sensor module improvements, and so on. Luckily, I was able to limit the extent of change to that of adding digital pots to the sensor module. One-megaohm digital pots on each of the sensitivity controls will allow the sensors to be dynamically adjusted to cover a much larger range, from the faintest moonlight to the brightest sun. While I haven't manufactured a new sensor module PCB or added any dynamic control code for the sensors yet, I rearranged some of the I/O on the microcontroller to release the I^{2}C pins for this use and added a couple of pins to the original six-pin sensor module connection (as shown in this article's schematics).

A FINAL NOTE

If the charging and power supply was designed efficiently, a system using that 5-W solar panel might create enough power to keep the panel tracking the sun—and have a bit left over for charging a battery. There is a big debate going on about whether the increased output you get from tracking the sun makes up for the higher system costs of the mechanical and electrical complexity needed to do so. Many of the utilities are beginning to accept power back from their customers, just ask Steve Ciarcia (*Circuit Cellar* 209 and 210). I want the smart power grid. I can see resurgence in home control. I believe solving our country's problems will be the source of our nation's recovery. It's fodder for entrepreneurs. Give a shout. What are you thinking about? ■

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.*

RESOURCES

Microchip Technology, "PIC16F193X/LF193X Data Sheet: 28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with LCD Driver and nanoWatt XLP Technology," DS41364C, 2009, ww1.microchip.com/downloads/en/DeviceDoc/41364C.pdf.

Sundial information, Thunderstruck Observatory, http://thunderstruckobservatory.com/sundials/index.html.

Survival Topics, "Using Shadows to Determine Direction," www.survivaltopics.com/survival/using-shadows-to-determine-direction/.

———, "Using Time as a Compass," www.survivaltopics.com/survival/using-time-as-a-compass/.

SOURCES

ISL29102 Converter
Intersil Americas, Inc. | www.intersil.com

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

ViewMate Gerber viewer
PentaLogix| www.pentalogix.com

PCB "Fab-In-A-Box" Kit
Pulsar Professional FX | www.pulsarprofx.com

by George Martin

Put C Language to the Test (Part 1)

A Sudoku Puzzle-Solving Program

You have a handle on C language. Now it's time to write a complete program from start to finish. How about a program for solving a Sudoku puzzle? Read on to learn how it's done.

It's time to write your first complete C program, from "soup to nuts"—or, more appropriately, from design to debug. It's a chance to put together all that I've covered in my past several articles. Remember: my articles about learning C programming are intended for the readers who understand hardware but are not yet C programmers. Perhaps you've been using assembly language to program an embedded processor. There are many introductory evaluation boards that include simple hardware and basic C code to read a push button switch and turn on an LED indicator. Refer to the Texas Instruments webpage in the Resources section of this article for information about third-party development kits. Every CPU vendor will have such a reference list, and these are a good way to get started. Attending a seminar at your local electronics distributor will also get you reduced (perhaps even free) access to these kits and tools. I'm not going to put this project on any specific evaluation board. My plans are to write it on a PC and then talk about how to transfer it to an embedded system.

We need a problem to solve. Blinking an LED just won't cut it. You guys and gals are too sophisticated for that. So, I'm proposing that we write a program to solve Sudoku puzzles. You've seen the puzzles everywhere (e.g., newspapers, magazines, online). And if you do a web search for "Sudoku" or "Sudoku Solver Programs," you'll come up with a long list of resources. Please don't look at these just yet. I plan to implement the design in such a way that you will be able to take the design and then add what you want to the project. Sudoku is well documented on the Internet, so we don't have to spend a lot of time and effort defining our problem. This program is not an embedded real-time system, but the process and technique for creating this program will be the same as it would be for real-time embedded programs.

OUR MISSION

Let's write a program that can solve 2 × 2, 3 × 3, and 4 × 4 Sudoku puzzles. You'll find the 2 × 2 puzzle useful for spending time with your children while teaching them problem-solving techniques and critical-thinking skills in general. The 3 × 3 puzzle is everywhere, and perhaps what we're doing will be of some value. The 4 × 4 puzzle is found in some technical magazines. Sometimes prizes are awarded for sending in the correct solution. So maybe we'll get rich. But if you learn something about C and have some fun, well, that will be good enough.

In addition to generating a puzzle and working it, I want to be able to load a puzzle and save a puzzle. Also I want any of the rules we come up with to be isolated so that you can understand them, improve them, and add to them. Actually, I'm trying to keep everything isolated to make changes easier. This code should also give you hints about solving the puzzle.

OUR TOOLS

We need a C compiler and computer to develop our code. I was going to recommend that you use a PC (running Windows XP SR3) and a free copy of Borland's C++ compiler. But as I tried to make sure it was still available, I didn't see a version with an Integrated Development Environment (IDE). I next tried Bloodshed Software's DEVC++, which had an IDE and seemed to work well. However, when I added a second file to the project, everything compiled and seemed to link, but I got linkage errors. I liked the setup, but I couldn't figure out how to work with multiple source files. If you succeed, please let me know and I'll pass it along.

I looked at other options—such as, Dev-C++ 5.0 beta 9.2 (4.9.9.2) (9.0 MB) with Mingw/GCC 3.4.2—but I settled on LADSoft's CC386IDE, which is an IDE that handles multiple files and seems to work well. I have a scroll wheel on my mouse, and it's oblivious to that feature. Also, it's sensitive to mouse positioning (i.e., sometimes I mark more or less text than I mean to mark). But, so far, it's working. This program is released under the GNU GPL. It's not released for commercial use. That said, give the author some feedback and let him know if you're using his work.

DESIGN

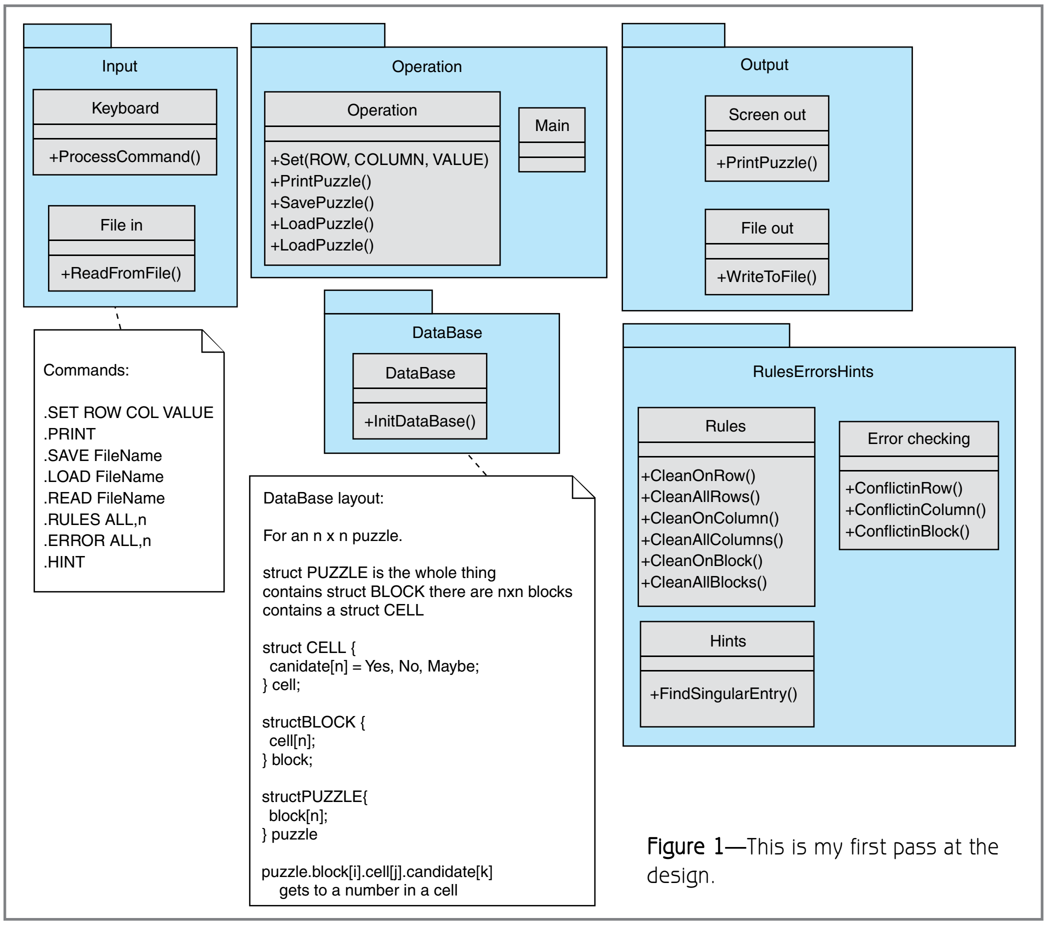
How do you start this design? I used to jump right to coding, which gave me a feel for how the user interface and the design were going to lay out. But I (and you) can't do that anymore. It's a waste of time, and the problems we are assigned to solve keep getting larger and more complicated. You need to be more prepared with your design before you start coding. I now start with the UML diagramming tools to lay out what I think should be done and what I need in the form of routines to accomplish the task. I use No Magic's MagicDraw v.9.5. There are much newer versions, but I'm satisfied with what I've got and I use it in many projects. I suspect that I'm not using UML in the manner the originally intended. I am *not* a UML expert by any stretch of the imagination.

Figure 1 depicts my first pass at the design. The blue boxes represent code modules (files in our case). The names of the blue boxes will probably become the file names of the modules. Greyish containers are in each of the blue boxes. These labeled containers hold the routines that will become a part of the design. I suspect they will also represent objects in an object-oriented language like C++. Don't worry, we won't go there just yet.

So, at this point, I'm reading the project requirements (that are only in my head at this time) and creating routines to perform operations I think I'll need. I'm grouping these routines based on the function and parameters required. I would probably define the variables for each routine at the beginning of the file that contains the routines. I'm reducing parameter passing and also isolating the details of the design. If each routine is only exposed to the data it needs, your code, design, testing process, and life in general will be simplified. I cannot state this much more emphatically. Keep your data isolated and you'll have more fun.

The initial design, partition, and variables required seem to make sense. When I start to code and debug, we'll see how these initial decisions hold up. I'll need a database to hold the data. It's more of a data structure array than a classic database. Also, it seems to make sense to put the Rules and Errors routines in one module. Take a look at the notes attached to the Input module. I'm starting to define the commands that will be decoded by the command processor. In addition, the notes on the database module define the database's structure. I feel good about the design approach in these areas.

Look at the operation module. It's got Main and Operation included in its contents. Main has little information, and Operation seems to have relevant procedures defined. But I just don't feel good



```
Commands:

.SET ROW COL VALUE
.PRINT
.SAVE FileName
.LOAD FileName
.READ FileName
.RULES ALL,n
.ERROR ALL,n
.HINT
```

```
DataBase layout:

For an n x n puzzle.

struct PUZZLE is the whole thing
contains struct BLOCK there are nxn blocks
contains a struct CELL

struct CELL {
  canidate[n] = Yes, No, Maybe;
} cell;

structBLOCK {
  cell[n];
} block;

structPUZZLE{
  block[n];
} puzzle

puzzle.block[i].cell[j].candidate[k]
   gets to a number in a cell
```

Figure 1—This is my first pass at the design.

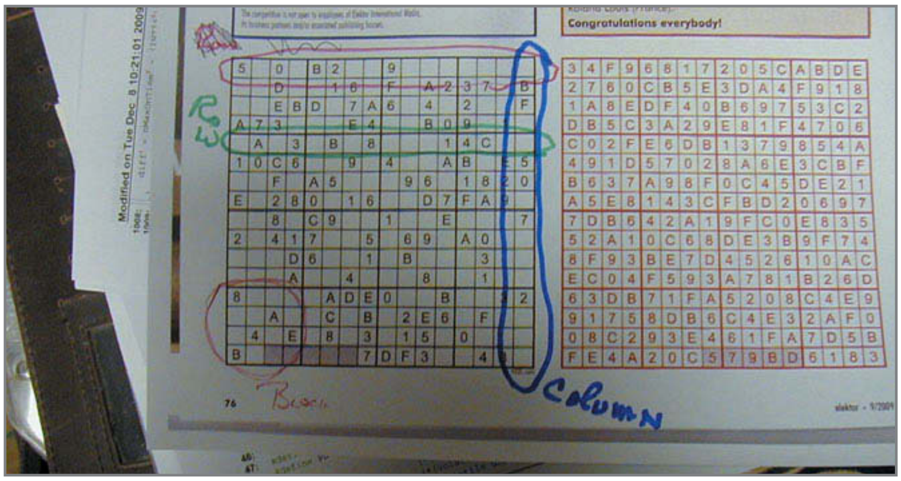


Photo 1—This is a 4 × 4 Sudoku puzzle. (Source: *Elektor*, September, 2009.)

about this module. Too much is unknown. Also, look at the input and output modules. They are both a mix of keyboard and file routines. I don't like that. It seems like a recipe for confusion.

As you see from the results in Figure 1, I put down what I thought I'd need to solve the problem. I probably did not get a good grouping of the routines, but at least I have a place to keep them and I could present this design approach for a review. In any case, I can print it out and give it more thought. I believe this UML diagram is a much better approach than just jumping into the design and coding.

DATABASE

Let's talk more about the database. A 4 × 4 Sudoku puzzle is shown in Photo 1. You see these everywhere. I heard of a story about a designer/inventor who created a program for generating these puzzles. He tried to sell his program to newspapers, but had no luck. So, he gave away a limited use program to one newspaper. After the reader interest increased, he then approached the competing newspapers and offered his program again. This time he met with great success. I don't know if it's a true story, so I would appreciate your feedback about its accuracy.

Let's start with some definitions. The *puzzle* is the entire puzzle. In our design, it could be 2 × 2, 3 × 3, or 4 × 4. A *block* is the area circled in red in Photo 1. In a block, no digits may repeat. The *row* is outlined in green in Photo 1. In a row, no digits may repeat. The *column* is circled in blue in Photo 1. In a column, no digits may repeat. These definitions will be used as a basis for our first three rules.

A *cell* is the smallest element of the puzzle. It can be blank or contain one of the valid numbers. The valid numbers for a 2 × 2 puzzle are 0, 1, 2, or 3. For a

ICbank
Gateway to Electronics

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

· myPIC

Only
$160 qty 100 $220 qty 1

Powerful feature

- MP3 Encoding, Real time decoding (320Kbps)
- Free charge MPLAB C-Compiler student-edition apply
- Spectrum Analyzer
- Application: Focusing for evaluation based on PIC
- Offer full source code, schematic

Specification
Microchip dsPIC33FJ256GP710 / 16-bit, 40MIPs DSC
VLSI Solution VS1033 MP3 CODEC
NXP UDA1330 Stereo Audio DAC
Texas Instrument TPA6110A2 Headphone Amp(150mW)
320x240 TFT LCD
Touch screen
SD/SDHC/MMC Card
External extension port (UART, SPI, I2C, I2S)

· myWave (MP3 DIY KIT SD card Interface)

Only
$150 qty 100
$200 qty 1

Powerful feature

- Play, MP3 Information, Reward, forward, Vol+/-
- Focusing for MP3 Player
- SD Card interface
- Power: battery
- offer full source code, schematic

Item	Specification
MCU	Atmel ATmega128L
MP3 Decoder	VS1002 / VS1003(WMA)
IDE Interface	Standard IDE type HDD(2.5", 3.5")
Power	12V, 1.5A
LCD	128 x 64 Graphic LCD
Etc	Firmware download/update with AVR ISP connector

· myAudio (MP3 DIY KIT IDE)

Only
$180 qty 100
$220 qty 1

Powerful feature

- Play, MP3 Information, Reward, forward, Vol+/-
- Focusing for full MP3 Player (Without case)
- IDE Interface
- Power: Adapter
- Offer full source code, schematic

ICbank
Gateway to Electronics

www.icbank.com

3 × 3 puzzle, they are 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. For a 4 × 4, they are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, or F. Keep in mind that a cell could have a number as its value, or several possible numbers as its value. In this latter case, we should report the value as blank.

Listing 1 is the database design for our puzzle. Reading from the bottom up, it declares space for a variable named `puzzle` that is a structure of type `PUZZLE`. Next you'll see that the structure `PUZZLE` is a collection of structures of type `BLOCK`, and `BLOCK` is a collection of structures of type `CELL`. And `CELL` is an array of the variable `INT16 numb`. Each member of the array `numb` can have the value of YES, NO, or MAYBE. I define YES, NO, and MAYBE as CCI_YES, CCI_NO, and CCI_MAYBE. This keeps me from mixing up definitions among all my projects.

The statement `puzzle.block[iBlock].cell[iCell].numb[iNumb] = CCI_MAYBE;` sets the one of the numbers (`iNumb`) in one of the cells (`iCell`) in one of the blocks (`iBlock`) to the value of CCI_MAYBE. I know this is confusing, and I thought about different approaches, but I truly believe this is a good representation for the data. Go back to the puzzle diagram in Figure 1. I hope you can see how the `iBlock` and `iCell` variables can get us to any cell in the puzzle. I would just number the elements from left to right, from top to bottom. So, the top left cell is `iCell` = 0 and `iBlock` = 0.

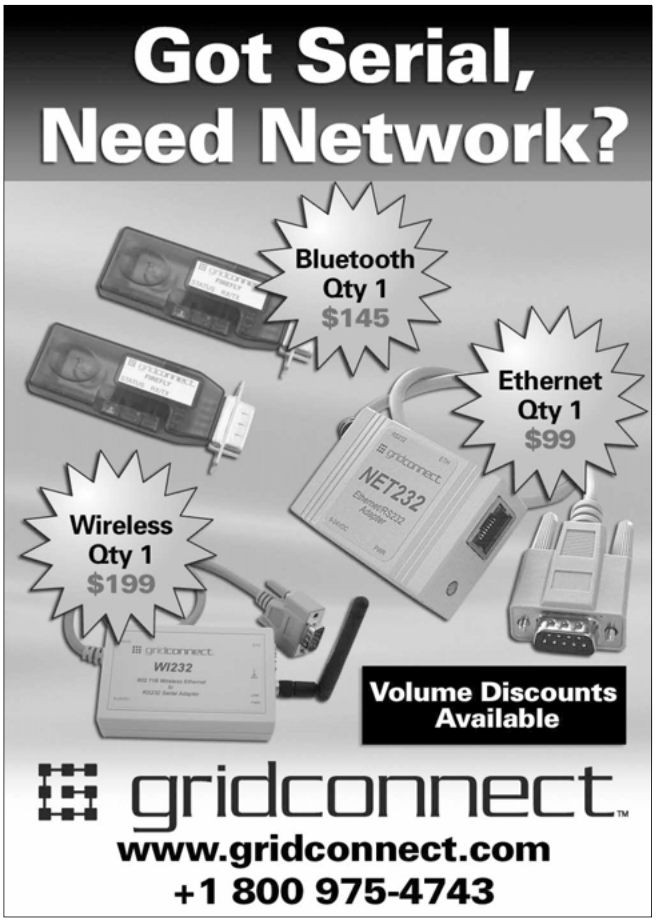
Listing 1—The database design for our Sudoku puzzle.

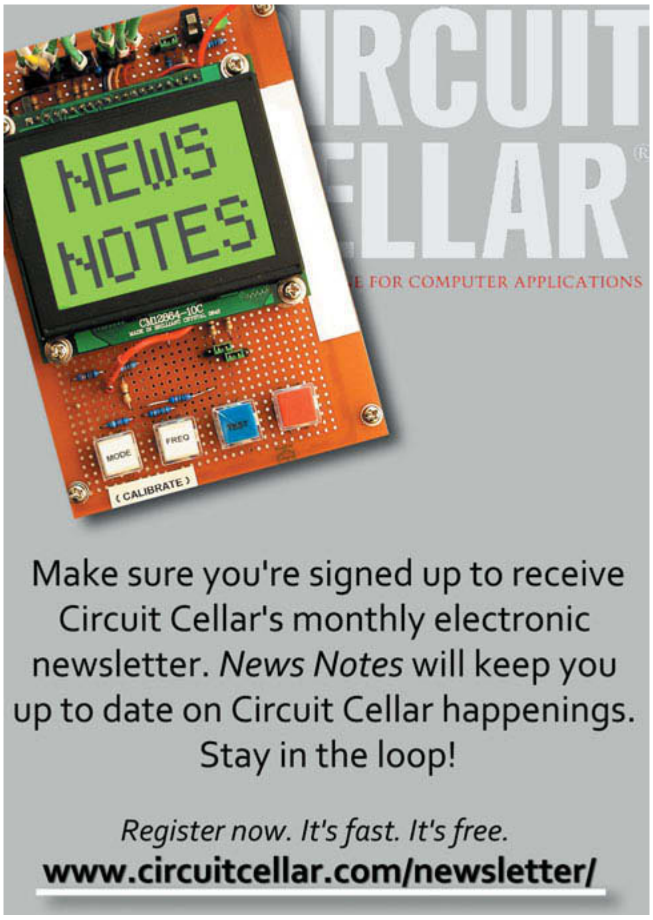
```
// Set up the data structures for each Cell
//
struct CELL {
   int numb[MAX_CANDIDATES];   // CCI_YES, CCI_NO, CCI_MAYBE
};

// Set up the data structures for each Block
//
struct BLOCK {
   struct CELL cell[MAX_CELLS];   // CCI_YES, CCI_NO, CCI_MAYBE
};

// Set up data structures for the Puzzle
//
struct PUZZLE {
   struct BLOCK block[MAX_BLOCKS];
};

struct PUZZLE puzzle;   // reserve space for the largest puzzle
```

THE CODE

The code that I'm presenting in this article is not fully functional. I thought it would be of value to show you the design as it progresses and becomes a working program.

First, look at the `main()` function.




CIRCUIT CELLAR
NEWS NOTES
Make sure you're signed up to receive Circuit Cellar's monthly electronic newsletter. News Notes will keep you up to date on Circuit Cellar happenings. Stay in the loop!
Register now. It's fast. It's free.
www.circuitcellar.com/newsletter/

It's located in the file main.c and is just a simple loop that looks at commands entered on the keyboard. The `kbhit()` routine returns TRUE when a key is hit on the PC's keyboard. On a PC, when you hit a function key, `getch()` returns a two-key sequence. The sequence is first a zero and then a value representing the function key. I am not using function keys in this design, but the code will not crash if a user enters one. If you press the Escape key, a value of 27 is returned. And that signals us that it's time to exit the program.

This is our complete main loop. I would think you'll agree it's rather simple. I used some new standard routines in writing this program. In `main()`, you'll see the `printf()` statement. I could spend many articles describing the `printf()` statement, and if we were doing a payroll program, that explanation would be valuable. But since we're not doing payroll-type programs, I'm going to suggest that you search the Internet (or other sources) for information about how `printf()` works. For now, let's just say it prints to the screen on the PC. The routine `getch()` routine is the classic C routine for inputting characters from the standard input device. `toupper()` is a C library function to convert to uppercase. C comes with a great set of libraries used for this everyday work. Many of the library routines (such as `getch()`) make a physical connection to the hardware your running. If you're using hardware you designed, you need to write this low-level interface to the hardware.

In the `main` loop, when a complete command line is received, it's passed to the `ProcessCmd()` routine. In that routine, you will find the command decoding. I used the `strncmp()` function from the C library. This routine compares two strings for n characters in each string. If the strings match, a zero is returned. I test the input line against all the possible commands one at a time. If a match is found, I call the appropriate routine and then return a value indicating successful decoding of the command (CCI_OK). If no match is found, I return a value indicating failure (CCI_FAIL). I use the #defines CCI_OK and CCI_FAIL to keep me sane. (Remember constants are in CAPITAL letters.) Working on several projects with several customers, I find sometimes PASS is a one and other times it's a zero. Putting the customer or project in the #define name helps isolate these common values in my world.

Looking at the commands decoded so far, you find TEST, INIT, PRINT, SETR, and READ. TEST is a routine I use to test my entire row/column math. More on this later. INIT is a command that initializes the puzzle space to all empty. In it, I hardcoded a 4 × 4 puzzle. Creating puzzles of different sizes is on my to-do list. The PRINT command prints the puzzle. In the PRINT function, you will find more `printf()` statements that produce a reasonable printout on the PC's screen. The `SETR` command will fill the values of a row with data. I found this is the easiest way to load the puzzle database. And finally, the READ command will transfer the input source for the program from the PC's keyboard to a file. In that file, we read and process lines of characters as if they come to us from the keyboard.

Although this code is stable and will run on a PC, it probably won't do what I (or you) expect it to do. "Your assignment, Mr. Phelps": get a C compiler up and running on a PC; get this code imported and compiled; and run the `INIT` and then `PRINT` commands. Next time, we'll get into the `SETR` command and develop our rules packages.

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 rev olver shooter.

PROJECT FILES

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

RESOURCES

C. Heng, Free C/C++ Compilers and Interpreters, www.thefreecountry.com/compilers/cpp.shtml.

Texas Instruments, Development Tools and Software, http://focus.ti.com/mcu/docs/mcudevtoolsandsw.tsp?sectionId=96&tabId=1503&familyId=4&toolTypeId=1.

SOURCES

DEVC++
Bloodshed Software | www.bloodshed.net/devcpp.html

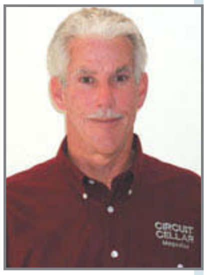
Borland C++ Compiler
Embarcadero Technologies, Inc. | www.codegear.com/downloads/free/cppbuilder

Dev-C++ 5.0 beta 9.2 (4.9.9.2) (9.0 MB) with Mingw/GCC 3.4.2 (File name: devcpp-4.9.9.2_setup.exe)
Geeknet, Inc. | www.sourceforge.net

CC386IDE 32-bit C Compiler
LADSoft | http://ladsoft.tripod.com/cc386.htm

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

by Tom Cantrell



Tales from the Crypto

A Look at Embedded Design Security

The bad guys are out to steal your design, your code, your tools, your customers, and anything else that isn't nailed down. Forget about trust, every designer needs to pay more attention to verification. Fortunately, you can use some silicon to help you keep your secrets safe.

It's interesting to contemplate how much of a person's net worth devolves to a few kilobytes of account information scattered around the ether. I keep thinking someday we'll all wake up and sign onto our accounts as usual only to find every financial asset we thought we owned was zeroed overnight. Easy come, but even easier to go, when all it takes is a "security lapse"—a *noxymoron* (i.e., two words that should go together)—and a few milliseconds to steal your bits.

Identity theft is a big deal, even for embedded apps. Consider the ongoing brouhaha over counterfeit consumables like laptop batteries and inkjet printer cartridges.[1] There are arguments on both sides of the issue, but no argument the stakes are high.

If breaking into silicon can happen at the speed of light, the solution is more silicon that can stop the nefarious bit busters in their tracks. Let's take a look at some new silicon security guards from Atmel that you can deploy to harden your design against attack.

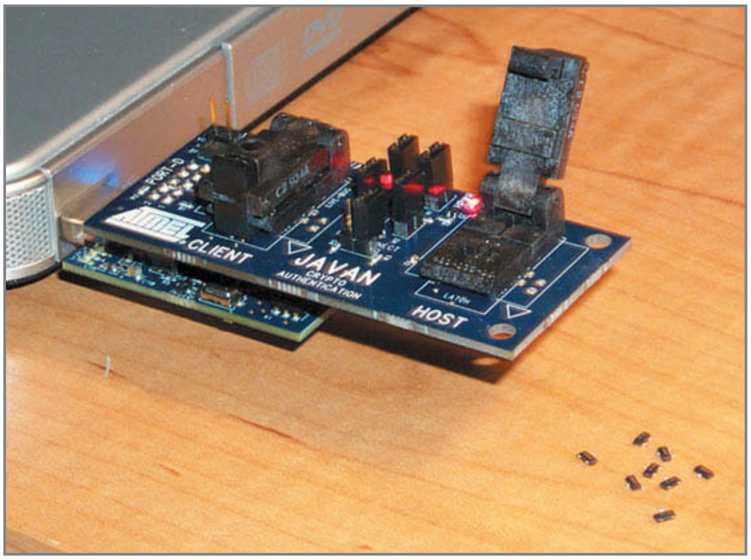


Photo 1—The JAVAN Crypto-Authentication starter kit makes it easy to put the '1025 and '10HS through their paces, unless you get an attack of the dropsies. The fumble-fingered need not apply.

WHO GOES THERE?

I asked him if he thought Detroit would win the World Series. He said no, but they put up a bloody good fight. We pulled them out of the Jeep because the World Series had been between the Cardinals and Browns.[2]—*Robert Gravilin, "The 3AD in the Battle of the Bulge"*

A fundamental challenge for any security regime is knowing who you're talking to, and that's exactly the capability Atmel's new "Crypto-Authentication" chips bring to the party. In the case of the

AT88SA102S, the protection comes at a remarkably small price. Not only is the per-chip cost low at just $0.72 (in quantities of 100), but as you can see (barely) in Photo 1, board space for these tiny three-pin padlocks isn't an issue (although finding one you drop on the floor is). Running on anything from 2.5 to 5 V, the '102S does consume a healthy 10 mA during the time it takes to actively vet credentials. However, average power consumption will be much less, presuming the chip can spend most of its time off-duty sleeping at just 100 nA. Indeed, in many applications (such as the aforementioned batteries and inkjet cartridges), authentication may take place just once during initial installation.

The essence of authentication is that the host confirms that a would-be client knows a shared secret as in a typical password arrangement. You could configure a '102S to work that way and be done with it. With a whopping 256-bit "password," attempting to break-in by trying every combination will be slow going indeed. It's all the more true when you realize a brute-force attack is limited by '102S throughput to maybe 10 trials per second.

Hmm, with more than 10^{77} possible combinations, it will take more than 10^{76} seconds to try them all, which is a real long time, about a zillion years (OK, 10^{68}) plus or minus. But remember, chances are the crooks won't have to try every combination before stumbling on the right one. OK, so on average, it will only take half a zillion years. Do you feel lucky, punk? Well, do 'ya?

You'd think the evildoers would be better off stealing lottery tickets. However, there's an all too obvious weakness with a simple password scheme—namely, that the password can be overheard and replayed by an eavesdropper or "man in the middle." The spy-versus-spy solution calls for an elaborate scheme by which the spymaster (i.e., host) confirms that a to-be-vetted agent (i.e., client) knows a secret without requiring the agent to actually divulge the secret (in which case it might be overheard). If a password is never passed, it can't be stolen.

Here's how it works with a '102S. The host issues a 256-bit "challenge" to the '102S. In turn, the '102S uses that number along with information hidden on-chip (e.g., a 256-bit "key," 48-bit unique chip ID, 64 bits of user-programmed "secret" fuses) as inputs to a "Secure Hashing Algorithm" (SHA-256).[3] The 256-bit result of the SHA-256 calculation (called the "digest") is returned to the host in response to the challenge. Meanwhile,

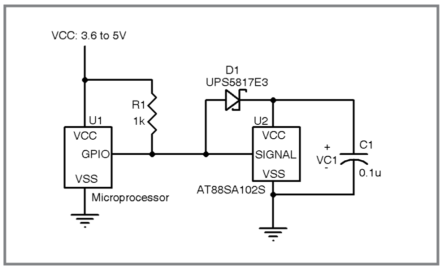
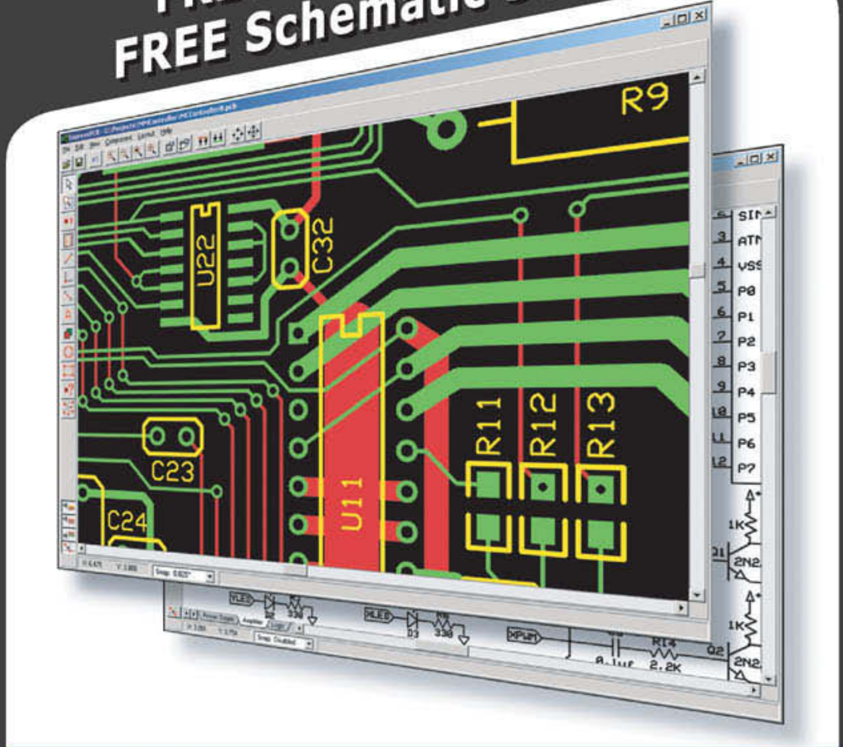


Figure 1—You can get by connecting a '102S with just two wires by tapping power from the signal line. Since the pull-up resistor (the '102S output is open drain) and bypass capacitor are required in any case, all it takes is the addition of a Schottky diode.

$51 For 3 PCBs

FREE Layout Software!
FREE Schematic Software!



01 DOWNLOAD our free CAD software
02 DESIGN your two or four layer PC board
03 SEND us your design with just a click
04 RECEIVE top quality boards in just days

expresspcb.com

the host—knowing both the challenge it issued and the secret information that should be in an authentic client—can compare the expected result to that received from the '102S. If the digests match, the client is authenticated without the secret information ever passing hands.

Hashing is fundamentally a compression scheme that encodes a big—and possibly variable in length—chunk of data in a small piece of fixed length data. It's a concept that's fundamental to many aspects of computing. For instance, virtual memory and caching both rely on hashing to translate memory addresses from big spaces into smaller ones. A file system can use hashing to map, for example, an arbitrary customer name and address to a unique record number on disk. Heck, even the lowly CRC and checksum are examples of hashing functions that distill a packet down to a byte or two.

Security apps call for hashing functions with particular features to stymie would-be crackers. For instance, the chances that two different inputs to the hashing function will produce the same output (called a "collision") must be minimized. The output of the hashing function should change in a seemingly random way, even for minor changes (i.e., a bit at a time) of the input. Oh yeah, it helps if the calculations involved don't require a supercomputer.

Far be it from me to weigh in on cryptographic nuances and controversies. I'll let Atmel speak for themselves when they say: "The NSA and universities involved in cryptographic research all agree that a brute force attack on SHA-256 is essentially impossible."[4] At the same time, the algorithm (mainly a bunch of shifting and Boolean logic) is quite manageable (time and space) for even an 8-bit MCU.

Sounds good, but the scheme is still subject to eavesdropping if the challenge is always the same. However, by taking advantage of the user-programmed fuses (i.e., secrets) and unique chip ID, the jeopardy can be limited to a single host-client pairing. Eavesdropping on a '102S that works with a particular host won't help the bad guy break into another host that expects a '102S with a different chip ID or fuse secrets.

A more robust solution calls for the host to vary the challenge similar to the way your car door lock remote control uses a "rolling code." Now a transient eavesdropper is up a creek because the challenge changes each time and capturing a single challenge-and-response transaction does little good.

The simplest option has the host store a number of precomputed challenge and response pairings at 64 bytes per pair. The more the pairings, the longer it will take an eavesdropper to capture them all. A better solution—although it's one that requires the host to perform the SHA-256 calculation—factors a random number into the challenge, keeping in mind the usual caveats (i.e., how truly random your random number generator is).

It's clear a '102S can harden a design against brute-force and eavesdropping attacks, but what about an inside job? A possible weak link in the aforementioned scheme is that it depends on the security of the shared secrets stored in the host, secrets which may be stolen (e.g., source code), reverse-engineered (e.g., execution trace), or

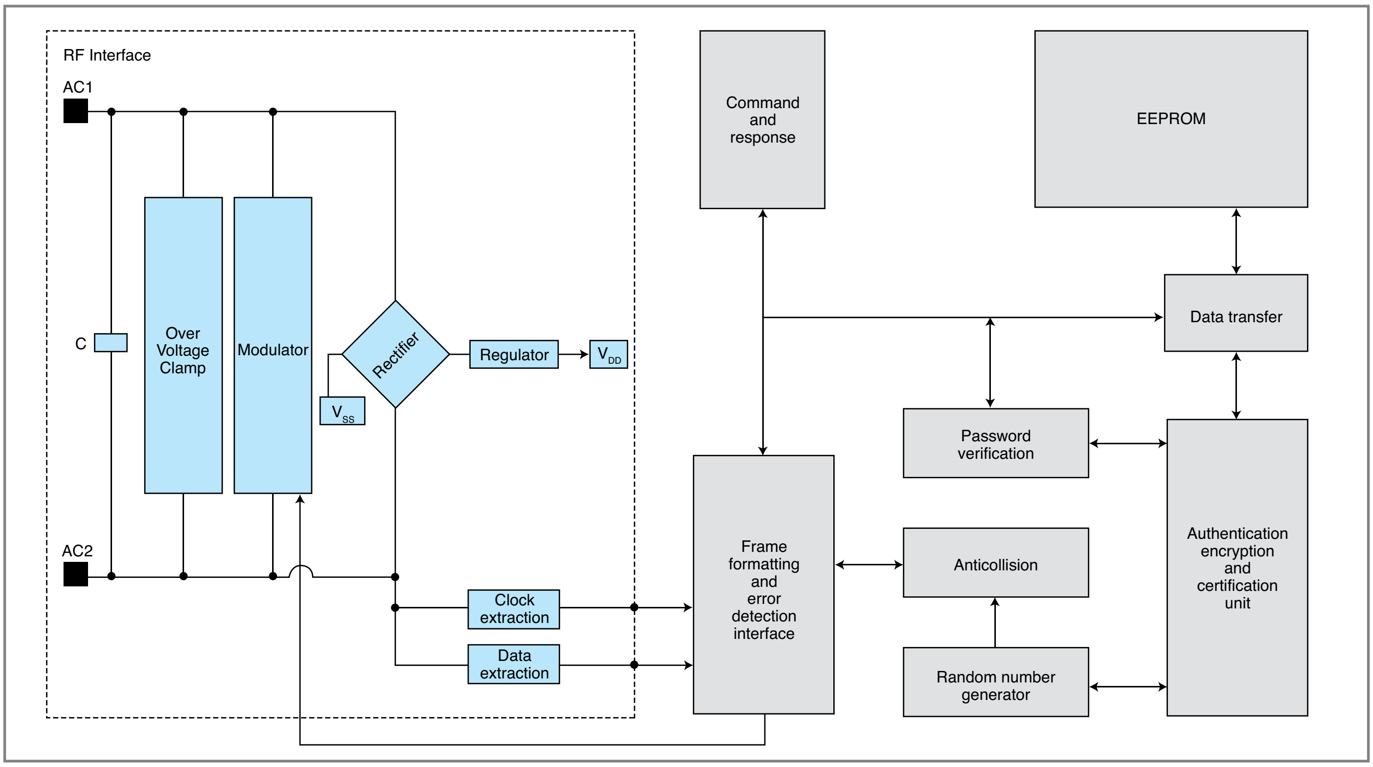


Figure 2—The RF04C is a passive RFID tag with a dif ference—namely, EEPROM to store your secrets and bulletproof security to keep them safe.

Command Name	Description
Abort	Exit command in progress
Clear	Exit command in progress, clear buffer, turn RF OFF
Poll Continuous	Poll continuously for Type B PICCs
Poll Single	Poll once for Type B PICCs
Read Buffer	Read data buffer
Read Register	Read configuration register
RF OFF	Turn off 13.56 MHz RF field
RF ON	Turn on 13.56 MHz RF field
Sleep	Activate standby mode
TX Data	Transmit data to PICC and receive the response
Write Buffer	Write data buffer
Write Register	Write configuration register

Table 1—I know little about RFID protocols and standar ds. And thanks to the intelligence embedded in the AT88RF1354 reader-on-a-chip, I never will. It hides the gory details—managing the radio, packet-handling collision resolution and so on—behind a simple serial interface and concise menu of high-level commands.

inadvertently exposed (e.g., software bug).

To that end, Atmel offers the AT88SA10HS, which is quite similar to—and designed to work in tandem with—the '102S. In this configuration, the host issues a challenge to both chips and delivers the response digest calculated by the '102S to the '10HS. In turn, the '10HS decides whether the '102S response is authentic and simply gives the host a yes/no result. Note that none of the secret information or results of any calculations are ever known (or knowable) by the host and thus can't be leaked.

That leaves a direct attack on the Atmel silicon—truly black-hat stuff—as the last resort. For obvious reasons, the documentation doesn't go into a lot of detail, but mention is made of a variety of physical protection mechanisms. These include chip shielding, tamper and pin-attack detection, internal clock dithering, and secrets dispersed across the die, not just sitting in a ROM.

LESS IS MORE

Needless to say, with just three pins (VCC, VSS, and SIGNAL), the hardware interface is easy. In fact, you can get by with just two pins using the old trick of tapping the I/O line to power the chip (see Figure 1). Atmel's application note titled "Crypto-Authentication" describes the details, such as how to size the capacitor and pull-up resistor considering the power supply (i.e. worst-case SIGNAL pin duty cycle) and demand (chip is awake, sleeping, calculating, performing I/O, etc.).[4]

Having less pins to deal with may make the hardware easy, but it does require more work from your software driver. The single SIGNAL pin has to serve half-duplex duty in both directions, so the driver is responsible for issuing commands to turn the bus around and avoid collisions.

Some applications might require the simultaneous authentication of multiple clients. The good news is that multiple '102S chips can ride on a single SIGNAL line. The bad news is that the protocol is a little goofy as it requires you to issue a command to each chip you don't want to talk to (telling it you don't want to talk to it) before you can talk to the chip you want to talk to.

Absent a pin for a clock, signaling is naturally asynchronous. The bit timing is a little bizarre in that it's asymmetric (i.e., 39 µs from the host to the '102S versus 60 µs in the other direction). I was figuring some bizarre bit-banging software would be required, but it turns out you can fake it with a standard UART running at 230.4 kbps. (Each byte transferred by the UART accomplishes a single bit transfer on the SIGNAL line.)

Chips without a RESET pin always make me nervous. It's almost inevitable that a hardware or software glitch will lead to a dead-end or loss of control that requires a power cycle. The '102S addresses the problem in two ways. First there's an I/O timeout that puts the chip to sleep if I/O gets out of sync. That way the chip won't get hung up waiting indefinitely for bits the host isn't sending. Similarly, a glitch on the SIGNAL line could wake up a sleeping '102S. Not knowing the '102S woke up, the host wouldn't know it needs to issue (or reissue) a SLEEP command. To solve this dilemma, the '102S has a watchdog timer that puts the chip to sleep a few seconds after it wakes up, no matter what (i.e., even if it is in the middle of executing a command or I/O). This strategy—which is,

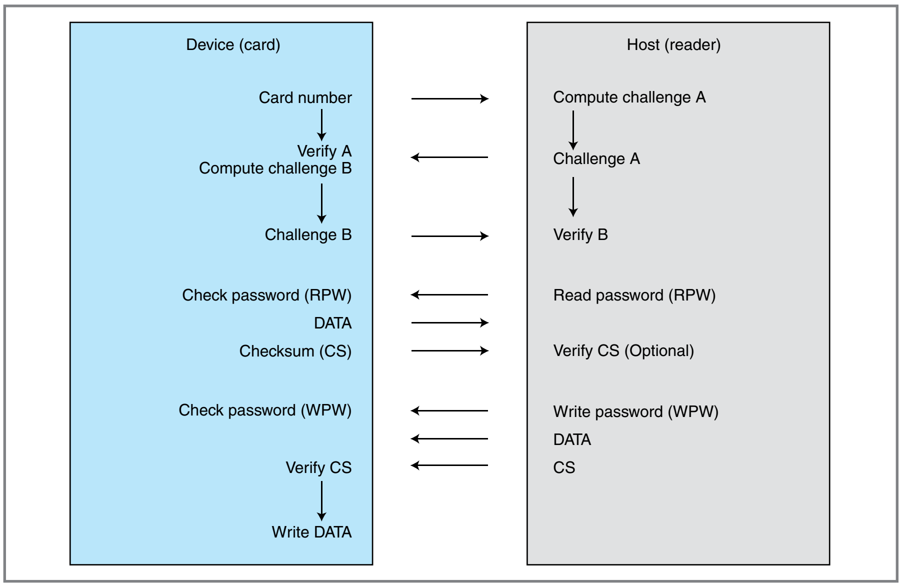
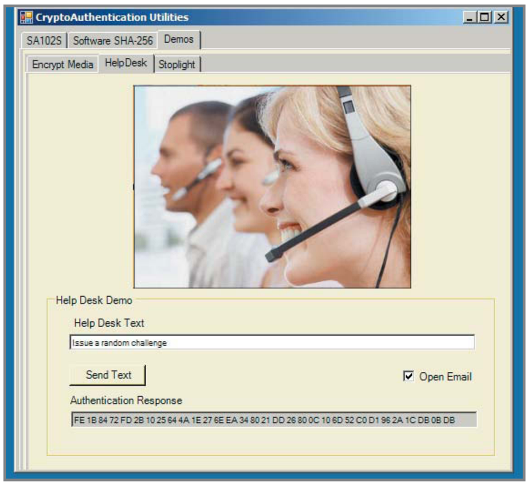


Figure 3—The RF04C features mutual (i.e., two-way) authentication in which both the host and client verify each other's identity and the integrity of messages is protected in both directions.



Help Desk Demo

Help Desk Text: Issue a random challenge

[✓] Open Email

Authentication Response: FE 1B 84 72 FD 2B 10 25 64 4A 1E 27 6E EA 34 80 21 DD 26 80 0C 10 6D 52 C0 D1 96 2A 1C DB 0B DB

Photo 2—The JAVAN software exercises Crypto-Authentication chip features and also demos some example applications. Going beyond a password or "your mother's maiden name," the chips could be used to verify that a person on the other end of a phone call or e-mail truly holds the key.

"if in doubt, sleep"—is all well and good, but it means your driver code has to be cognizant of timing less it run afoul of the automatic failsafes.

It's all a bit cumbersome, but maybe that's appropriate. The bad guys will have to deal with this too. Why make it easier?

On the other hand, Atmel does just that with the JAVAN EV kit ($99.95, quantity of one) that connects to your PC via USB. Thankfully, the board includes some cute little sockets, one for a "Client" (e.g., '102S) and one for a "Host" (e.g., '10HS). Otherwise, I'm just not sure how you would homebrew a programming and prototyping lash-up for these laughably tiny chips.

Better yet, the JAVAN kit comes with some handy PC utility software to access, program, and demo the chips (see Photo 2). Debugging security hardware and software can be tough because it's just that—secure. There's really no way to test that you've set up and programmed the Atmel chips correctly after the fact, other than trying them out. If you're having problems, is it your chip setup or hardware interface or driver software that's to blame? Or maybe it's all of the above? Having a stable and proven platform like JAVAN to start with really helps reduce hassles and head-scratching as you craft your own design.

TRICKY TAG TEAM

The Crypto-Authentication chips aren't Atmel's first foray into security. Earlier offerings included a line of CryptoMemory chips that I covered back in 2007 ("Silicon Secrets," *Circuit Cellar* 205). You'll recall the CryptoMemory chips combine security features (e.g., keys, passwords, authentication, and encryption) and defenses against attack (e.g., password/authentication attempt counters, tamper detection, silicon shielding, and secret obfuscation) with 4 to 64 Kb of EEPROM.

Now Atmel offers "CryptoRF" parts that extend the CryptoMemory concept with the addition of RFID capability. Consider the AT88RF04C ($0.95, quantity 100), which combines a 4-Kb CryptoMemory and a standard (ISO/IEC 14443 Type B) 13.56-MHz RFID interface (see Figure 2). I'm not an expert in the RFID field, but it appears this "13.56-MHz Type B" is a very widely used standard, which means the 'RF04C should be compatible with a lot of different readers. Or, for that matter, it's easy to design your own since Atmel is also introducing the AT88RF1354, virtually a complete reader on a chip ($1.59, quantity 100). You can put the pair through their paces with the "Bamboo" CryptoRF evaluation kit for just $99.95 (see Photo 3).

The 'RF04C is a so-called "passive" tag. That means it's completely powered from the RF field generated by the reader, so the tag itself doesn't need a battery and will keep working indefinitely. And despite their fragile appearance, the tags are robust enough to use across the industrial temperature range of –40° to 85°C, the only caveat being EEPROM data retention is derated at higher temperatures (e.g., 10 years at 55°C versus 30 years at 35°C). Write endurance is 100,000, cycles which seems more than adequate, even for long-life applications.

Combining "passive" power via RF with EEPROM poses unique challenges. Foremost among them is the fact that power may disappear at any time, possibly during writes to the EEPROM, which is a real no-no. To protect the integrity of EEPROM, the 'RF04C incorporates a unique two-stage "anti-tearing" write cycle. When enabled, EEPROM write data is first sent to a temporary EEPROM buffer and then, after that write completes successfully, on to the main array. If power fails during the first (buffer) write, the main array data remains unaffected. If power fails during the second (main array) write, the chip tries again (i.e., writes the main array from the buffer) at the next power-up.

RFID is trickier than its lowly bar code and price-tag

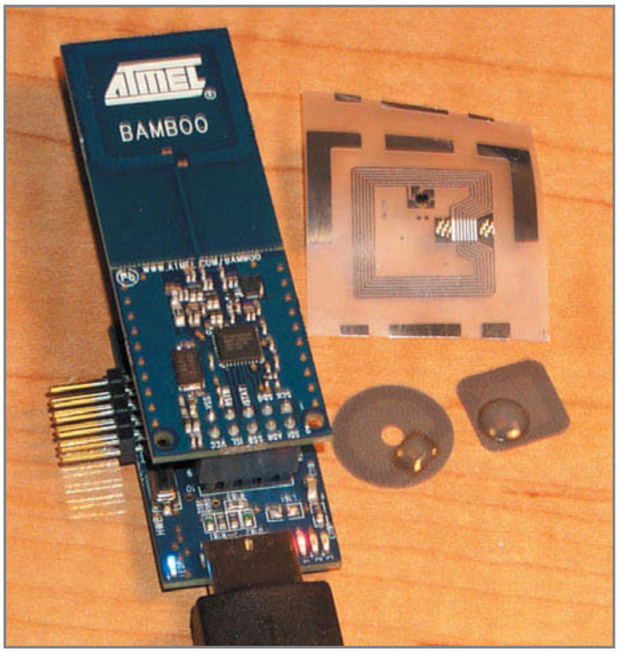
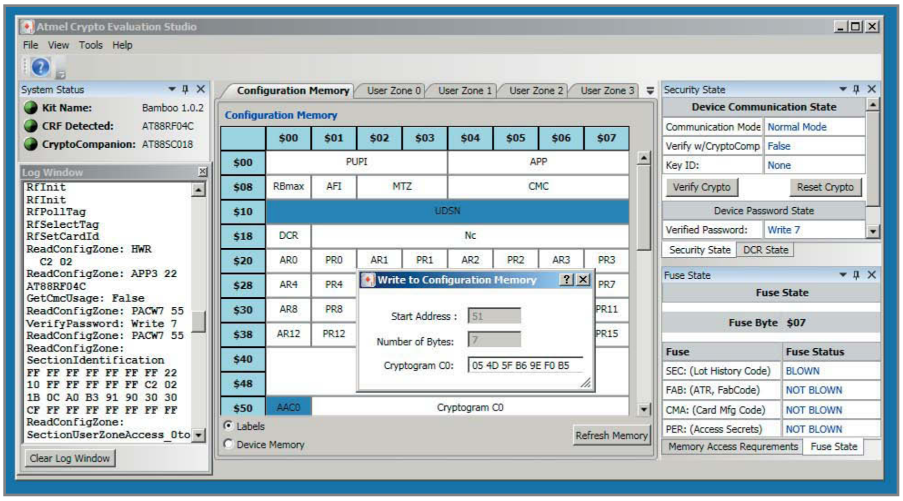


Photo 3—The BAMBOO CryptoRF starter kit comes with an USB plug-in AT88RF1354-based RF reader and quiver of RF04C CryptoRF tags.



Kit Name:	Bamboo 1.0.2
CRF Detected:	AT88RF04C
CryptoCompanion:	AT88SC018

```
RfInit
RfInit
RfPollTag
RfSelectTag
RfSetCardId
ReadConfigZone: HWR
  C2 02
ReadConfigZone: APP3 22
AT88RF04C
GetCmcUsage: False
ReadConfigZone: PACW7 55
VerifyPassword: Write 7
ReadConfigZone: PACW7 55
ReadConfigZone:
SectionIdentification
FF FF FF FF FF FF FF 22
10 FF FF FF FF FF C2 02
1B 0C A0 B3 91 90 30 30
CF FF FF FF FF FF FF FF
ReadConfigZone:
SectionUserZoneAccess_0to
```

	$00	$01	$02	$03	$04	$05	$06	$07
$00	PUPI				APP			
$08	RBmax	AFI	MTZ		CMC			
$10	UDSN							
$18	DCR	Nc						
$20	AR0	PR0	AR1	PR1	AR2	PR2	AR3	PR3
$28	AR4	PR4						PR7
$30	AR8	PR8						PR11
$38	AR12	PR12						PR15
$40								
$48								
$50	AAC0	Cryptogram C0						

Start Address :	51
Number of Bytes:	7
Cryptogram C0:	05 4D 5F B6 9E F0 B5

Device Communication State	
Communication Mode	Normal Mode
Verify w/CryptoComp	False
Key ID:	None

Device Password State	
Verified Password:	Write 7

Fuse State

Fuse Byte $07

Fuse	Fuse Status
SEC: (Lot History Code)	BLOWN
FAB: (ATR, FabCode)	NOT BLOWN
CMA: (Card Mfg Code)	NOT BLOWN
PER: (Access Secrets)	NOT BLOWN

Photo 4—The BAMBOO kit software does two useful things. First, it lets you see what' s happening on the radio link, a must for debugging any wireless app. Second, it lets you get under the hood of the CryptoMemory chip and twiddle bits (EEPR OM, passwords, configuration, fuses, etc.) to your heart's content.

roots might imply. Notably, there's the issue of "collision" to deal with. That is, what happens when multiple tags are simultaneously within the reader's range?

The ISO standard incorporates an anti-collision protocol to sort things out. The tag and reader both support a user-programmable "Application Family Identifier," which would presumably allow the reader to selectively poll amongst a group of tags. If you refer to the 'RF04C tag block diagram, you'll notice how the box labeled "Anticollision" utilizes the on-chip random number generator. Presumably, the random number provides the basis for tags to schedule around and otherwise resolve collisions.

Fortunately, there's no need to grapple with such low-level machinations since the 'RF1354 reader IC handles all the gory details behind the scenes. All you need to deal with is a simple hardware interface (SPI or TWI) and command set (see Table 1). Power up the reader in the presence of multiple tags and it will automatically sort through them to connect to just one (see Photo 4).

Speaking of powering up, do remember that delivering the juice to power the tag from the reader via RF is a nifty concept, but it isn't real efficient. During tag interrogation, the 'RF1354 can burn more than 1 W, which is plenty for such a small part. To avoid overheating that might occur, most likely when using the "Poll Continuous" command, the surface-mount package incorporates a thermal pad on the bottom that should be soldered to thermal vias that dissipate the heat from the chip around your PCB.

Any time the subject of RFID and "smart tags" comes up, folks worry crooks will be sniffing RF for secrets and Big Brother will be tracking their every move. That's not likely with 'RF04C security features that include 64-bit mutual authentication and encryption (see Figure 3), zone-by-zone keys and passwords, and access attempt counters. There's also the practical matter that the range for these "near-field" (i.e., inductive) tags is just 15 mm (less than an inch), although Atmel also offers a larger "smart card" tag that's good for up to 10 cm (a few inches). In any case, if a crook wants to try to eavesdrop or smooth talk an 'RF04C, they're going to have to either get real up close and personal or homebrew their own high-power reader.

WHAT WILL YOU DO?

As if credit card fraud, identity theft, stolen e-mail, and the like weren't enough, I imagine it's only a matter of time before embedded apps come under increased security scrutiny. Take a close look at your own application. How much do you stand to lose if someone steals the family jewels, or worse, sabotages operation in such a way that someone gets hurt? At less than a buck, Atmel's security add-ons provide an easy and inexpensive way to armor existing applications without going through the hoops of a ground-up redesign.

Trust me (and only me) when I say it seems like a small price to pay for peace of mind. ◘

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.

REFERENCES

[1] E. Ogg, "HP Sues Ink Cartridge Maker for Unfair Competition," http://news.cnet.com/8301-10784_3-9731037-7.html.

[2] R. Gravilin, "The 3AD in the Battle of the Bulge," July 1, 2004, http://www.3ad.org/members_pages/newsletter_archives/2004_newsletters/Volume_04_Issue_3.pdf.

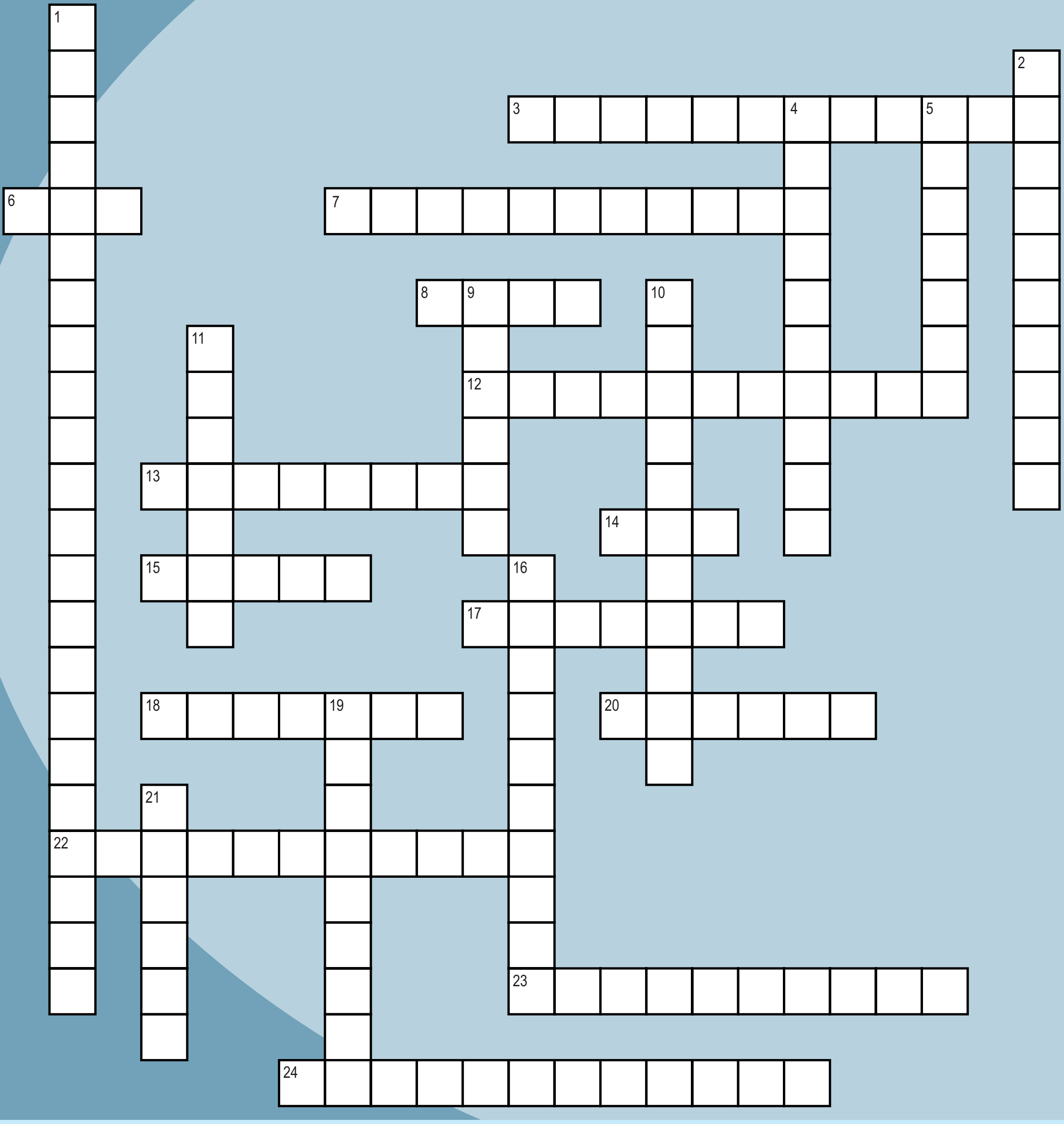
[3] NIST, "Secure Hash Standard," Federal Information Processing Standards 180-2, 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.

[4] Atmel Corp., "Crypto-Authentication: High Level Security Models," 8666A–SMIC, 2009, www.atmel.com/dyn/resourcesprod_documents/doc8666.pdf.

SOURCES

AT88SA102S Crypto-Authentication chip and CryptoRF RFID tags
Atmel Corp. | www.atmel.com

CROSSWORD



Across

3. The "10" in 10BaseT [two words]
6. Wozniak
7. 10^{-15} amp; fA
8. NIST radio station
12. .nl
13. (Max + Min)/2
14. 1,055 J
15. Mark-8 minicomputer ; Virginia Tech, 1973
17. GW
18. V = R × I [two words]
20. Pa
22. What is "–3" in the term $-3x^3y$?
23. Silver, graphite, copper
24. ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2010/236 [two words]

Down

1. An electrician's "bible" [three words]
2. Energy in the Earth's crust
4. EXE
5. Leetspeak: *addy*
9. 1-d-r
10. Op
11. 2^{70}
16. An insulator
19. *The TV Cheap Video Cookbook*, 1978
21. Old hardware in a new design

The answers are available at www.circuitcellar.com/crossword.

IDEA BOX

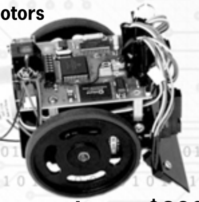
THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital files that meet our specifications (www .circuitcellar.com/advertise). ALL TEXT AND OTHER ELEME NTS MUST FIT WITHIN A 2" x 3" FORMAT. E-mail adcopy@circuitcellar.com with your file or send it to IDEA BOX, Cir cuit Cellar, PO Box 180, Vernon, CT 06066. For current rates, deadlines, and more information contact Peter W ostrel 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 pr oducts and services.

CCS Talking Robotics Development Kit with PIC16LF877A

Kit Includes:
- Optimized C Compiler for PIC® MCUs
- In-Circuit Programmer/Debugger
- Control board with electronic compass, proximity, and IR detection
- Durable chassis with dual ball bearing servo motors



Kits starting at $209
262.522.6500 x35
sales@ccsinfo.com
www.ccsinfo.com/BOT10

Weather Instruments for PCs

AAG electrónica

www.aagelectronica.com

bob basic overlay board

BOB-4

Low-cost video data overlay module

Features:
Automatic text scroll and crawl
'TTL-232' and SPI control ports
Any size for custom fonts
Vector and bitmap graphics

www.decadenet.com

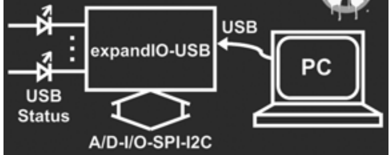
DECADE ENGINEERING
503-743-3194 Turner, OR, USA

For Any PCB - GHz Bandwidth BGA/QFN Sockets

Quick-Turn Custom Sockets
• Use on any existing PCBs
• No mounting holes
• No soldering
• Patented placement/ epoxy system
• 0.32mm larger than IC (per side)
• Bandwidth to 40 GHz
• 0.3mm pitch & up
RoHS

Ironwood ELECTRONICS 1-800-404-0204
www.ironwoodelectronics.com

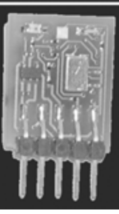
Revolutionary new expandIO-USB chip



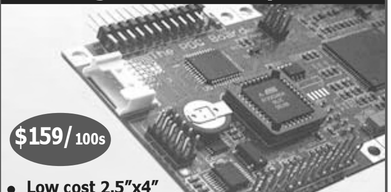
- Ideal for adding USB to sensors & peripherals
- No drivers needed for Windows, Mac, Linux
- No microcontroller programming required
- Also check out our USB-232 USB to UART

www.hexwax.com - Buy from Mouser & Farnell

Amazing PIC programmer
Most devices supported
ICSP, SQTP, & copy limits
$32 at Digikey & Mouser
www.flexipanel.com



PDQ Board™ - A Fast I/O-Rich Single Board Computer



- Low cost 2.5"x4" C-programmable computer
- 16-bit HCS12 processor clocked at 40 MHz
- 8 PWM, 8 counter/timer, and 8 digital I/O
- 16 10-bit A/D inputs
- Dual RS232/485 ports, SPI and I²C ports
- 512K on-chip Flash, 512K RAM with Flash backup
- Plug-in I/O expansion, including Ethernet, Wi-Fi, GPS, 24-bit data acquisition, UART, USB, Compact Flash card, relays, and more ...

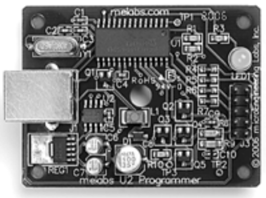
Mosaic Industries Inc.
tel: 510-790-1255 fax: 510-790-0925
www.mosaic-industries.com

Order online at:
www.melabs.com

microEngineering Labs, Inc.
Development Tools for PIC® Microcontrollers

Phone: (719) 520-5323
Fax: (719) 520-1867
Box 60039
Colorado Springs, CO 80960

USB Programmer for PIC® MCUs **$89.95** (as shown)



RoHS Compliant

Programs PIC MCUs including low-voltage (3.3V) devices

Includes Software for Windows 98, Me, 2K, & XP

With Accessories for $119.95:
Includes Programmer, Software, USB Cable, and Programming Adapter for 8 to 40-pin DIP.

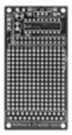
Serial LCDs
2-line x 16 $39.95
4-line x 20 $49.95
Quantity Discounts Available!

LAB-X Experimenter Boards

Pre-Assembled Boards
Available for 8, 14, 18, 28, and 40-pin PIC® MCUs
2-line, 20-char LCD Module
9-pin Serial Port
Sample Programs
Full Schematic Diagram

Pricing from $79.95 to $349.95

PICPROTO™ Prototyping Boards



Double-Sided with Plate-Thru Holes
Circuitry for Power Supply and Clock
Large Prototype Area
Boards Available for Most PIC® MCUs
Documentation and Schematic

Pricing from $8.95 to $19.95

BASIC Compilers for PICmicro®

Easy-To-Use BASIC Commands
Windows 9x/Me/2K/XP Interface

PICBASIC™ Compiler $99.95
BASIC Stamp 1 Compatible
Supports most 14-bit Core PICs
Built-In Serial Comm Commands

PICBASIC PRO™ Compiler $249.95
32-bit signed variables and math operations
Supports Microchip PIC10, PIC12, PIC14, PIC16, PIC17, and PIC18 microcontrollers
Direct Access to Internal Registers
Supports In-Line Assembly Language
Interrupts in PICBASIC and Assembly
Built-In USB, I2C, RS-232 and More
Source Level Debugging

See our full range of products, including books, accessories, and components at:
www.melabs.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

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

Make sure you're signed up to receive Circuit Cellar's monthly electronic newsletter. *News Notes* will keep you up to date on Circuit Cellar happenings. Stay in the loop!

Register now. It's fast. It's free.
www.circuitcellar.com/newsletter/

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
Fast checkout through Google

www.stx104.com
Apex Embedded Systems
help@stx104.com • 608-256-0767 x24

CIRCUIT CELLAR
Designer's Notification Network

Circuit Cellar design contest entrants have received thousands of valuable development tools and product samples. Because of their contest participation, these engineers receive advance e-mail notice from Circuit Cellar as soon as new samples become available. Now you too can benefit from this early notification.

Welcome to the Designer's Notification Network. Print subscribers are invited to join the Network for advance notice about our new sample distribution programs.

Find out more at www.circuitcellar.com/network.

PROFESSORS

The Circuit Cellar college program puts quality engineering information in the hands of your students every month. Sign up now to get Circuit Cellar distributed to your class this semester.

ELECTRONIC COMMUNICATIONS
Op Amp Design Techniques
MATHEMATICS IN ELECTRONICS
Linear IC Technology
BASIC INSTRUMENTATION
Introductory Circuit Analysis

To update your professor account or to find out more about our college program, visit www.circuitcellar.com/products/collegeprogram

KETEREX®

Turn your PC into a SPI, I2C, SMBus, or GPIO Controller

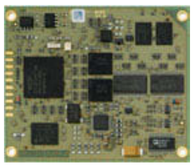
Affordable Test and Measurement Tools

www.keterex.com

PHYTEC

Inside great products. Behind great ideas.

phyCORE® System on Modules:
- shorten time-to-market
- reduce development costs and avoid substantial design issues and risks
- Windows® Embedded CE and Linux BSPs (processor-dependent)
- $129/unit benchmark price at 1K for ARM9-based SOM
- Design Services available to assist with deployment into target applications


phyCORE-LPC3250

ARM11: i.MX35, i.MX31
ARM9: i.MX27, LPC3250, LPC3180
Cortex M3: STM32F103
ARM7: LPC2294
XScale: PXA270
x86: Z510, Z520, Z530 (Atom®)
Blackfin: ADSP-BF537
Coldfire: MCF5485
PowerPC: MPC5554, MPC5567, MPC5200B, MPC565, MPC555

phyCORE® Rapid Development Kits include SOM, Carrier Board, LCD (kit specific), schematics, software, free BSP for applicable kits and a start-up guarantee. The Carrier Board serves as a target reference design, allowing the SOM to easily port to the user's target hardware.

www.phytec.com | 800.278.9913 | www.phycore.com

iMAGEcraft C

C development tools for AVR • PSoC Pro • CPU12 ARM • MSP430 Propeller

NEW: V8 IDE!

Dinkumware libraries

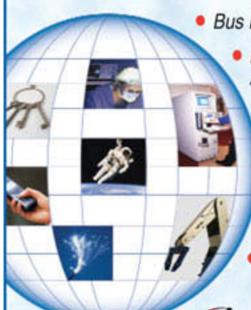
- Power and Simplicity -

Plus: Hardware CANUSB, CAN232 ELMICRO kits, etc.

Full-featured 45-day demos on our website!

www.imagecraft.com
info@imagecraft.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

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

RC™ Industrial Analog Signal Acquisition and Control

Starting at $99 OEM board

- 5.0 x 3.7", 5-24V DC power
- C/C++ programmable
- 16 ±10V analog inputs, 8 ±5V analog outputs
- 15+ TTL, 7 Solenoid drivers, Temperature Sensor
- CompactFlash, Ethernet, USB
- LCD, RS-232/ RS485/wireless
- 10+ years life time, Power-on-run within 1 second

100+ Low Cost Controllers with ADC, DAC, UARTs, 300 I/Os, solenoid, relays, CompactFlash, 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

ezLCD - The "Smart Display" makes integrating a GUI ez!

*Versatile Programming LCD module
*USB, SPI, RS232 Interfaces
*Bright 250 Nit LED Display (800 x 600)
*Integrated Touch Screen
*LUA Scripting Language capable-
For stand alone embedded apps
*Memory 3.8 MB + SD to 2G
*ezLCD's are also available in 2.7", 3.5", 5.6", 6.4", 10.4"

www.earthlcd.com

Call for Custom Display Configurations

CROSSWARE®

Tools for Embedded Development

ARM®
8051
COLDFIRE®

Three quality tool suites

- ❑ C/C++*
- ❑ Code Wizards
- ❑ Debugging
- ❑ Simulation
- ❑ Support

* Embedded C++ for ARM and ColdFire only

Advanced software tools since 1984

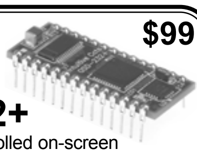
www.crossware.com
info@crossware.com

Low Cost CAD Software for Windows XP, NT and Vista

- Circuit design package with schematic entry circuit-board layout with autorouting and simulation for only $499!
- Buy modules starting at $119 (SuperCAD, SuperPCB, mentalSPICE & SuperSIM)
- Order and download instantly
- Full up package allows up to 16 layers plus 4 power planes
- Manufacture circuit boards at any board house

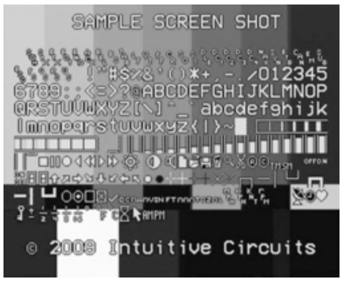
Mental Automation, Inc.
253-858-8104
www.mentala.com

New! $99



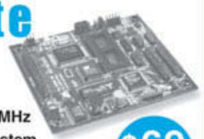
OSD-232+

RS-232/TTL controlled on-screen composite video character and graphic overlay in a small 28 pin dip package.


SAMPLE SCREEN SHOT
!"#$%&'()*+,-./012345
6789:;<=>?@ABCDEFGHIJKLMNOP
QRSTUVWXYZ[\]^_ abcdefghijk
lmnopqrstuvwxyz{|}~
© 2009 Intuitive Circuits

Intuitive Circuits
www.icircuits.com
(248) 588-4400

Flashlite 186



- 186 processor @ 33 MHz
- DOS w/ Flash File system
- 44 Digital I/O lines w/ CPLD
- Console / Debug Serial Port
- 7-34V DC or 5V DC power
- 2 Serial Ports
- Accepts 8MB DiskOnChip
- 2 16-bit Timers
- 512K DRAM & 512K Flash
- Watchdog Timer
- Expansion options with Peripheral Boards

$69 QTY 1

$99 Development System

Development kit includes:

- Flashlite 186 controller
- Borland C/C++ ver 4.52
- FREE Email Tech Support
- Serial Driver library
- AC Adapter and cable

Call 530-297-6073 Email sales@jkmicro.com
On the web at www.jkmicro.com

JK microsystems

ALL ELECTRONICS CORPORATION

Electronic and Electro-mechanical Devices, Parts and Supplies. Wall Transformers, Alarms, Fuses, Relays, Opto Electronics, Knobs, Video Accessories, Sirens, Solder Accessories, Motors, Heat Sinks, Terminal Strips, L.E.D.S., Displays, Fans, Solar Cells, Buzzers, Batteries, Magnets, Cameras, Panel Meters, Switches, Speakers, Peltier Devices, and much more....

www.allelectronics.com
Free 96 page catalog
1-800-826-5432

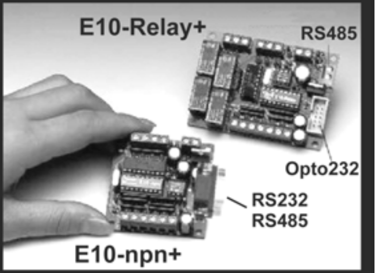
I2C SPI 1 Wire

3 Separate Buses
5V & 3V
Simple ASCII Interface
Cross Platform - All OS

www.i2cchip.com

The $69 PLC

Work as Stand-Alone Ladder Logic PLC.
Or as Smart Remote I/Os of PC/ PLCs.
RS485 allows 256 units to be networked.



Incredibly Easy to Program!
Our software is used by many colleges for teaching PLCs!

Get Free Ladder Logic Simulator:
www.tri-plc.com/cci.htm

Triangle Research

Tel: 1-877-874-7527 - PLC specialist since 1993

It writes your Full Speed USB Code!

***NEW!* HIDmaker FS for Full Speed FLASH PIC18F4550**

Creates complete PC and Peripheral programs that talk to each other over USB. Ready to compile and run!

- Large data Reports
- 64,000 bytes/sec per Interface
- Easily creates devices with multiple Interfaces, even multiple Identities!
- Automatically does MULTITASKING
- Makes standard or special USB HID devices

NEW! "Developers Guide for USB HID Peripherals" shows you how to make devices for special requirements.

Both PC and Peripheral programs understand your data items (even odd sized ones), and give you convenient variables to handle them.

PIC18F Compilers: PICBASIC Pro, MPASM, C18, Hi-Tech C.

PIC16C Compilers: PICBASIC Pro, MPASM, Hi-Tech C, CCS C.

PC Compilers: Delphi, C++ Builder, Visual Basic 6.

HIDmaker FS Combo: Only $599.95

DOWNLOAD the HIDmaker FS Test Drive today!

www.TraceSystemsInc.com
301-262-0300

Trace SYSTEMS, Inc.

INDEX OF ADVERTISERS

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

Page		Page		Page		Page	
75	AAG Electronica, LLC	48, 49	Elektor	13, 32	JKmicrosystems, Inc.	53	Nurve Networks LLC
12	AP Circuits	58	Elektor	76, 78	JKmicrosystems, Inc.	13	PCBCore
78	All Electronics Corp.	32	Elprotronic	42	Jameco	10	PCB-Pool
76	Apex Embedded Systems	57	Embedded Developer	53	Jeffrey Kerr, LLC	C4	Parallax, Inc.
33	CWAV	69	ExpressPCB	21	Keil Software	77	Phytec America LLC
39	Cleverscope	64	ezPCB	77	Keterex, Inc.	20	PoLabs
17	Comfile Technology, Inc.	75	FlexiPanel Ltd.	53	Lakeview Research	31, 56	Pololu Corp.
27	Copeland Electronics	12	Front Panel Express LLC	76	Lawicel AB	41	Renesas USA
78	Crossware Products, Inc.	66	Grid Connect, Inc.	11	Lemos International Co. Inc.	29	Sealevel Systems
75	Custom Computer Services, Inc.	27	HobbyLab, LLC	77	MCC (Micro Computer Control)	2, 3	Technologic Systems
75	Decade Engineering	78	I2CChip	77	Maxbotix, Inc.	77	Tern, Inc.
27	DesignNotes	65	ICbank, Inc.	78	Mental Automation, Inc.	5	Texas Instruments
C3	Digi International	77	Imagecraft Creations, Inc.	76	microEngineering Labs, Inc.	7	TI DesignStellaris 2010 Contest
10	EMAC, Inc.	1	Imagineering, Inc.	75	Mosaic Industries, Inc.	78	Trace Systems, Inc.
34	ESC - West	76	Intuitive Circuits LLC	55	Mouser Electronics	78	Triangle Research Int'l, Inc.
77	Earth Computer Technologies	75	Ironwood Electronics	C2	NetBurner	19	WIZnet: iMCU Design Contest 2010

PREVIEW of April Issue 237

Theme: **Embedded Programming**

Home Automation for an Energy-Efficient House (Part 1): System Design and Server Software

Converter Performance Management: Design Tips for Working with On-chip ADCs

Smart Control: An Innovative Approach to Reflow Soldering

Serial Network Hub (Part 2): Circuit Design and Usage

THE DARKER SIDE **Picoammeter Design**

ABOVE THE GROUND PLANE **Totally Featureless Clock (Part 2):** Hardware

FROM THE BENCH **Read-Only Memories:** Audio Applications with a Playback IC

SILICON UPDATE **Wizard Lizard:** Start Working with a Multidimensional SBC

ATTENTION ADVERTISERS

May Issue 238 Deadlines

Space Close: Mar. 12
Material Close: Mar. 22

Theme
Measurement & Sensors

Bonus Distribution
Sensors Expo

Call Peter Wostrel
now to reserve your space!
800.454.3741 or 978.281.7708
e-mail: peter@smmarketing.us

by Steve Ciarcia, Founder and Editorial Director

A Matter of History

A couple months ago, a columnist in one of the major American car magazines was recollecting about all the cars he had personally owned over the years. It was an interesting list, ranging from consumer vehicles to European exotics, and it spanned quite a few years. Certainly, plunking down personal money for a Porsche or two added critical experience and credibility when it came to writing test reports on the latest Audi R8, but I really think he just wanted an excuse to reminisce with his longtime readers who'd shared similar ownership experiences.

Considering many of us have been hanging around together here and at *BYTE* for 30-plus years, I guess it's OK for me to reminisce once in a while too. I'd love to describe my experience owing a few European exotics, but I suspect that this isn't the place to find a lot of similar ownership experiences. Instead, at *Circuit Cellar*, I guess we just sit down with a glass of cognac and reminisce about how we all got into the current mess—computers, that is. ;-)

For me, it started in 1973. I was an engineer at Control Data Corporation performing the '70s-era version of embedded control (gluing minicomputers onto production processes). As a reader of *Radio Electronics*, I avidly read about and then built the 16 line × 32 character TV typewriter (by Don Lancaster) presented in the September 1973 edition. I quickly got it to work, but I fondly remember that the entire digital design—which also included some analog circuit tweaking—functioned pretty much like one big "controlled" race condition. But hey, it worked.

Having a serial terminal didn't mean much by itself, but that was soon remedied by the Mark-8 Computer that appeared in the July 1974 issue of *Radio Electronics*. Presented by Jon Titus, the Mark-8 used the new Intel 8008 and, in my opinion, started both the personal computer revolution and the point of no return for my career. Jon's article was revolutionary, but a bit too DIY for most. The article offered complete plans and sets of PC boards but no complete kits. The Mark-8 newsletter, started as a support for those of us scrounging for parts, contained the first published project written by yours truly (about power supply design).

I never actually built the Mark-8, because immediately after Jon's article was published, I discovered that barely 50 miles south of me Scelbi Computers was selling an 8008-based computer. Fortunately, I was able to convince Nat Wadsworth to sell a PC board set to a poor newbie engineer. Within a couple months, I had an 8008 computer with 16 KB of memory, a cassette interface, a TV typewriter, and a vector graphics display.

From then on, developments came in rapid succession. The January 1975 issue of Popular Electronics featured the Altair 8800 based on Intel's next-generation 8080 processor and presented the S-100 bus architecture for the first time. While there was a kit available for the Altair, I was skeptical because the price for the basic kit was less than what Intel was quoting as the price of the 8080 processor by itself. Many of us didn't order Altair kits because delivery was slow and we feared it might contain a "drop-out" (and justifiably lower priced) processor. Before I could satisfy my gripes and prejudices to order an Altair, I discovered the Digital Group and *BYTE* magazine.

BYTE started publishing in September 1975. My first article, the vector graphic display for my 8008 computer, appeared in November 1976. This first project used the 8008 because that's all there was, but it wasn't the article's focus. Especially when I had a regular design column to consider, I had to think about generic compatibility. Like all my early projects, this vector display and subsequent hardware interfaces were parallel port driven. The separate I/O bus, along with interchangeable CPU cards on the Digital Group computer, enabled me to easily interface these monthly projects. I upgraded to a Z80 CPU because it was faster than the 8080/8008 and wrote the interface code using a universal BASIC interpreter because it was understood by a wide segment of the readership. I certainly appreciated the utility of Apple, KIM, TRS-80, VIC-20, and so on, but I wanted to avoid brand-specific projects and CPU-specific code until a brand became ubiquitous enough to justify a specific discussion. During the early days, when many of you were KIM or Osborne Computer owners, I wasn't discriminating. I was attempting to stay generic.

Of course, nothing lasts forever. Subsequent evolutions in computer architecture and software have redefined development computers and implementation electronics again and again. Today, universal adoption of the desktop PC virtually dictates that all software development starts there, while the continued advancement of inexpensive mega-function microcontrollers (aka 100 kilos in a 1-gram bag) prescribes "smart-dust" control for virtually everything except a rock.

So, reminiscing about my past computer selections only serves to remind me that all the computers I've owned had more to do with writing/managing articles than warm fuzzy personal compatibility. With all that said, I don't think I missed any life-changing experiences by failing to be up close and personal with a Sinclair ZX-80, but I certainly would have missed a lot without a Ferrari.

steve.ciarcia@circuitcellar.com

Steve

Winning Ideas Start Small and Grow Big

Digi's Wireless Green Design Contest

$20,000
in Cash Prizes

Check out our video at
www.digi.com/gogreen
to see how easy it is.

Digi International is challenging anyone with a good idea, no matter how big or how small, to participate in our Wireless Green Design Contest. iDigi Starter Kits provide the hardware, software and services to grow a good idea into a great green solution. Buy a $149 kit, submit your design and you could win!

iDigi™

Go green. Go Digi.

Digi®

www.digi.com/gogreen

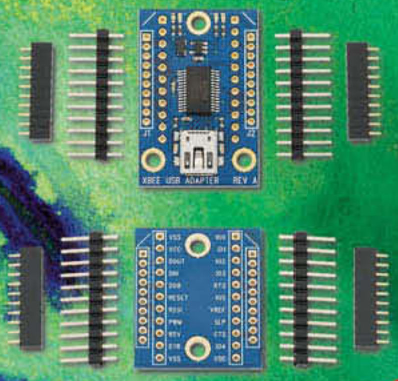
XBee Wireless Solutions

The **Digi XBee 802.15.4 modules** are the easiest-to-use, most reliable and cost-effective RF devices we've experienced. The XBee modules provide two friendly modes of communication – a simple serial method of transmit/receive or a framed mode providing advanced features. XBees are ready to use out of the package, or they can be configured through the X-CTU utility or from your microcontroller. These modules can communicate point to point, from one point to a PC, or in a mesh network. Picking the right XBee module is best accomplished by choosing an antenna style (chip or wire) and power level (2 mW for up to 300 feet and 60 mW for up to 1 mile).

Digi XBee 802.15.4 modules

Power Level	2 mW	2 mW	60 mW	60 mW
Antenna	Chip	Wire	Chip	Wire
Price	$19.00	$19.00	$32.00	$32.00
Stock Code #	#32404	#32405	#32406	#32407

XBee modules may be used with our XBee adapters and the multicore Propeller chip (3.3 V). Non 3.3 V microcontrollers like the BASIC Stamp will need a 3.3 V power supply and signal buffer chip for communication to the XBee module.



Because the XBee modules have 2 mm pin spacing, we recommend one of our adapter boards for each module. Our adapter boards provide several advantages to the XBee modules such breadboard-friendly standard 0.1 inch pin spacing, mounting holes, and easy-to-solder connections. Even if you are communicating point-to-point without a PC, we still recommend that you always have at least one XBee USB Adapter (#32400; $24.99) so you can easily configure and test each XBee module prior to putting it in a point-to-point application. Also, always have at least one of our affordable XBee Adapter Boards (#32403; $2.99) for each XBee module you purchase.

Order **XBee RF Modules and Accessories** at www.parallax.com or call our Sales Department toll-free: 888-512-1024 (Mon- Fri, 7am-5pm, PT).

PARALLAX INC

www.parallax.com

Friendly microcontrollers, legendary resources.™

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

BONUS ARTICLE

by Titus Gabriel Petrut

The Arduino-Based "MiniEric" Robot

The "MiniEric" is a general-purpose robot used for experimentation. It includes an Arduino board and several microcontrollers for the smaller modules. You can build a similar robot to experiment with at your workbench.

I created the "MiniEric" general-purpose robot for experimental purposes, and also so I could participate in robotic competitions (e.g., line following, fire fighting, search-and-rescue, and "robo waiter" events). MiniEric is a complex robot (modular in concept) that's still evolving (see Photo 1). An Arduino serves as the robot's "brain," and several microcontrollers are used for the smaller modules in the system.

Unlike many robotics enthusiasts with experience working with Arduino, and who are always looking to implement it in new projects, I discovered Arduino while I was working on my existing project. Here's how it all started. Many of us "geeks" dream of having a personal assistant robot (i.e., a butler robot) to help out with everyday chores—and, of course, fetch that cold beer from the fridge. All jokes aside, such a design is my long-term goal. Why long-term? Well, for robotics, you need to be knowledgeable in three different fields: mechanics, electronics, and programming. As a "regular Joe" with only a little background in these fields, I needed time to be able to build my butler robot. After a bit of practice with Lego Mindstorms, I gave it a shot and built my first butler prototype using an Evolution Robotics ER1 robot kit. The kit came with motors, a controller, and a frame for installing a laptop and hooking up a webcam (for detecting objects and people). I customized the kit by adding an arm (a weak one) and I named it Eric. But I had a hard time programming the robot, because I had no idea how to program a PC and the native interface was limited. Therefore, I decided to build a cheaper, smaller version so I could improve my programming skills. The idea was to develop most of the code to run on the modules that I could later install on a larger robot. This is how the MiniEric robot was born.

I wanted the MiniEric to meet a few requirements. One, it had to be inexpensive. Two, it had to be easy to build (with mostly hand tools). Three, it had to be a smaller replica of a larger robot. And finally, I wanted to incorporate most of the functions a bigger robot would have. I managed to keep costs low for most of the parts. (I did, however, spend money on sensors that I plan to incorporate in the big robot. I wanted to test them out first.)

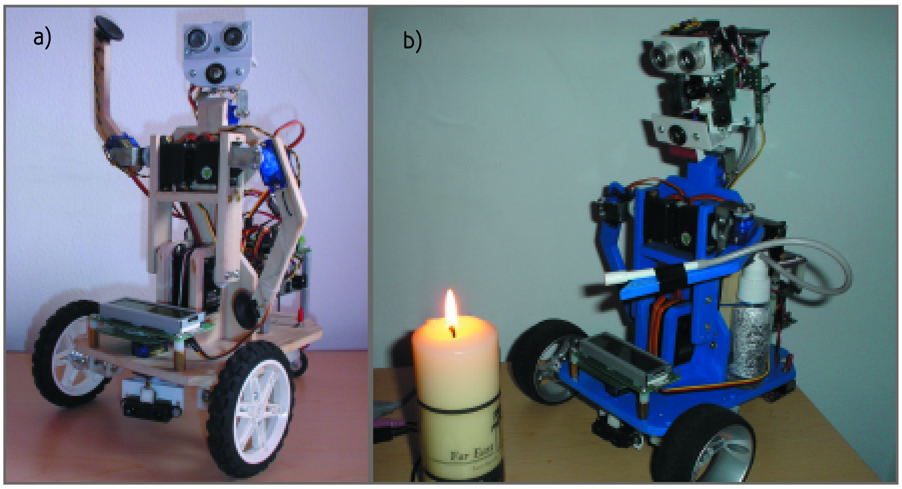


Photo 1a—This is the finished robot b—Here the robot is about to put out a flame.

I started the process by designing with Google SketchUp, which is free, easy-to-use, 3-D CAD software that's quickly becoming more and more popular. To give it the general look of the Johnny 5 robot in the movie *Short Circuit* (Sony TriStar, 1986), I designed it to have a bendable torso that's separate from its body. This enables the robot to lift objects from the floor and place them on a shelf or small table, but without the added weight of the head servos and sensors in the torso. The arms can move independently, and they can act as a claw for grasping objects.

I printed the designed parts, traced them, and cut them out from a piece of 0.25† poplar board I bought at a hardware store and assembled everything using small wood screws. I ordered the servos from HobbyKing.com for less than $7 apiece, along with some wheels I never used and the tail wheel. I made the servo brackets from an automotive CV boot clamp I had on hand. The sensor brackets were made from a piece of a leftover L-shaped PVC bar I got from a shower frame. Finally, the wheels were from an old Lego Mindstorms RIS kit.

The robot has eight servos: one for bending at the waist, two for the arms, two for the shoulders, two for the panning/tilting the head, and one for a sensor scanner mounted under the deck. At first, I used two more servos, modified for continuous rotation for driving, but they were too noisy for my taste and didn't have encoders. I replaced them with a pair of Faulhaber geared motors with built-in encoders. These motors are great, but rather weak for this robot, so I might change them in the future. As for the sensors, the robot has a Parallax Ping))) ultrasonic range sensor mounted on the head, along with a quite expensive I^{2}C Devantech eight-pixel thermal array sensor that is capable of detecting body heat and a candle flame. A Sharp IR range sensor is mounted on the scanning servo, low to the ground, so the robot can "see" and center itself on the objects it wants to pick up, or do other things like follow a wall. Push buttons were mounted on the tip of the robot's arms to be able to feel when an object was gripped, but they were too stiff and the servos didn't have enough power to push them, so I decided to replace them with Interlink Electronics 0.5" circular force-sensitive resistors (FSRs) instead. An infrared remote control receiver sensor is also mounted on the head. It enables me to use a universal remote control—tuned for a Sony TV—to control the robot remotely. In the future, I'll install a battery sensor and some other specific sensors will be used on the I^{2}C interface.

The most important part of the robot is its brain. I had several Atmel microcontrollers lying around: an ATmega32 development board, an AVR Butterfly, a MegaBitty board, and a custom robot board (called Ro-Bot-X) that I had made a few years back based on an ATmega8. Only the last board was built with what I needed, so I started to look around for alternatives that would meet two requirements. One was to use an AVR microcontroller so I could program it in Bascom-AVR. (I was a basic programmer, like most beginners.) The other was to have lots of three-pin header connectors for easier servo and sensor connectivity. This is how I found Arduino. Funny thing: it didn't meet any of my conditions, so I almost skipped over it. But after I read more about it, I was pleased to see an open-source hardware and software platform. However, I was at first a bit afraid that it would be tough to learn how to program it; but in the end, I was amazed at how simple it actually was. The only problem was the lack of three-pin headers. Fortunately, I learned that this could be solved with a custom-made shield. While I was asking questions on the "Society of Robots Forum," a user asked me if I wanted to be a beta tester for his new robotic board called Roboduino, which was based on the Freeduino variant. I was thrilled and promised him I'll use it to the max. Well, I kept my promise. The MiniEric uses all of the pins from the Roboduino!

My first goal was to have the robot respond to a remote control. I found all the necessary code on the Arduino Forum. I modified it, so I can even teach the robot new servo moves and store them in the EEPROM by just using the remote. Then, with a push of a button, I can command it to move, wave an arm, play a song, pick up or release an object. I added a serial LCD so I can see the sensor readings, and I programmed the robot to follow a wall and detect a candle light.

I found Arduino software was easy to learn and work with. It had libraries for all of the hardware I needed—and I used a lot: eight servos (using SoftwareServo library), one Ping))) sensor, one Sharp sensor, a speaker, a serial LCD (using SoftwareSerial library), an EEPROM library, a thermopile sensor (using the I^{2}C interface, Wire library), a dual DC motor controller (using the hardware serial), and an IR remote sensor (I built a library for it). All of the robot's code takes less than 60% of flash memory space—unless, of course, I leave all of the `serial.print` statements uncommented. I can have a few different programs in there and just select them with the remote when the time comes. And if I ever need more program space, I can upgrade to the ATmega328 microcontroller that the new Arduino Duemilanove has onboard. However, I reached the point where too many things needed to happen at the same time, so I had to split the functionality in several modules. I already have a serial DC motor controller. Next will be a servo controller because the servo timing interferes with other tasks. Perhaps I'll use the Roboduino as a servo controller—using the new servo library based on Timer1 interrupts—and add another Arduino on top as the main brain and connect them via I^{2}C. But then I will need a prototyping shield for each new Arduino board in the system. I decided to take a different route and design my own multi-modular board that I can customize for the different modules I need to use. This is how the "R-Dev-Ino" was born.

The R-Dev-Ino is a Robotic Development Arduino-compatible board with a small prototyping area and a dual-mirrored I^{2}C connector so multiple boards can be stacked straight or staggered. Each board uses an ATmega168 or ATmega328 with an Arduino bootloader, and they are programmed using an external FTDI Basic board I bought from SparkFun.

I've used four boards on my robot, each of them customized for a special purpose. The MotorController module was customized by soldering a dual H-bridge SN754410 and screw connectors for the motors. Also, this module takes care of the quadrature encoders that are built in the motors. The ServoController module was customized by soldering the filtering capacitors and the switching voltage regulator for the servos. The SpeechController module was customized by soldering a RC filter and a small amplifier for the Text-to-Speech function. Also, this module is the interface to the Voice Recognition (VRbot) module and it handles the logic that triggers the actions by voice commands. The Mapper module was customized with a voltage interface for a Nokia Color LCD breakout board. This module takes care of the robot's position on the map and displays it on the color LCD. The first two modules are slave-only, and the last two modules can be either slave or master, as needed. Implementing I^{2}C communication was pretty easy. The Arduino's Wire library that handles the I^{2}C interface is interrupt-based and handles the possible multi-master conflicts with ease. All I needed to do was to set up a list of unique commands (followed by two parameters), so there are no two identical commands in the system—no matter the destination address.

MiniEric went through an intensive rebuilding process that included a paint job, a little height reduction by changing the wheels and a small cut from the vertical structure, a complete head redesign, complete change of on-board electronics, and the addition of an extra servo to trigger the spray. It was almost ready for the Canadian National Robotic Games and a fire-fighting competition. Unfortunately, some low-level functions didn't work properly after the rebuild, and I spent too much time fixing them instead of debugging and improving the navigation. So, I skipped the competition, but I performed some tests to show some of the capabilities, like starting on voice command, driving in a straight line, finding the candle flame, and putting it out by spraying water. You can watch a couple of videos demonstrating that on my blog (http://seriousrobotics.wordpress.com).

I am really happy with the Arduino platform. I have learned a lot since I started to use it, and I've solved all my problems by asking questions on the Arduino forum. The Arduino integrated perfectly in my project. When I outgrow it, I could design a board to meet my needs and still use the Arduino software on it. Because it's simple to use, versatile, and easy to program, I think Arduino is the best platform for both beginners and professionals. ■

Editor's note: For more information about the hardware and programming associated with this project, go to http://seriousrobotics.wordpress.com/.

Titus Gabriel Petrut is a computer technician with a strong interest robotics. He lives in Toronto, Canada. You may contact him at robotxro@yahoo.com.

SOURCES

Arduino board
Arduino | www.arduino.cc

TPA81 Thermopile array
Devantech Ltd | www.robot-electronics.co.uk

ER1 Robot kit
Evolution Robotics, Inc. | www.evolution.com

141:1 6VDC Gearhead motor
Faulhaber | www.faulhaber.com

0.5† Circular force-sensing resistor
Interlink Electronics, Inc. | www.interlinkelectronics.com

PING))) Ultrasonic sensor
Parallax, Inc. | www.parallax.com

VRbot Speech recognition module
RoboTech | www.robotechsrl.com

GP2D120 IR Range Sensor
Sharp Microelectronics of the Americas | www.sharpsma.com

Nokia color LCD breakout board
SparkFun Electronics (distributor) | www.sparkfun.com