

Q&A: Designing with New Sensing Technology
LOCATION: United States
PAGE: 46

SECURITY: Embedded Authentication
LOCATION: United States
PAGE: 54

ARCHIVE PROJECT: Alarm Data Logger
LOCATION: United States
PAGE: 70

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

JANUARY 2013
ISSUE 270

EMBEDDED APPLICATIONS

**Future Chips:
Can MoS₂ Best Silicon?**

**MCU-Based Altitude
Control**

**Failure Mode &
Criticality Analysis**

**Web-Based Energy
Monitoring**

PLUS
Open-Source Success

DIY Data Logger Design

// Arduino's Applicability

// Data Logger Shield

// System Programming

// And More



www.circuitcellar.com

Now with 32MB Flash and 64MB RAM!

MOD54415 Core Module

32-bit 250 MHz processor
64MB DDR2 RAM
32MB flash
10/100 Mbps Ethernet
44 general purpose I/O
Eight UARTs
Five I2C
Two CAN
3 SPI
1-Wire®

5 pulse width modulators (PWM)
SSI

MicroSD flash card

8 analog to digital converters (ADC)
Two digital to analog converters (DAC)

NANO54415 Core Module

32-bit 250 MHz processor
64MB DDR2 RAM
8MB flash
10/100 Mbps Ethernet
30 general purpose I/O
Eight UARTs
Four I2C
Two CAN
3 SPI
1-Wire®

8 pulse width modulators (PWM)
SSI

MicroSD flash card ready

6 analog to digital converters (ADC)
Two digital to analog converters (DAC)



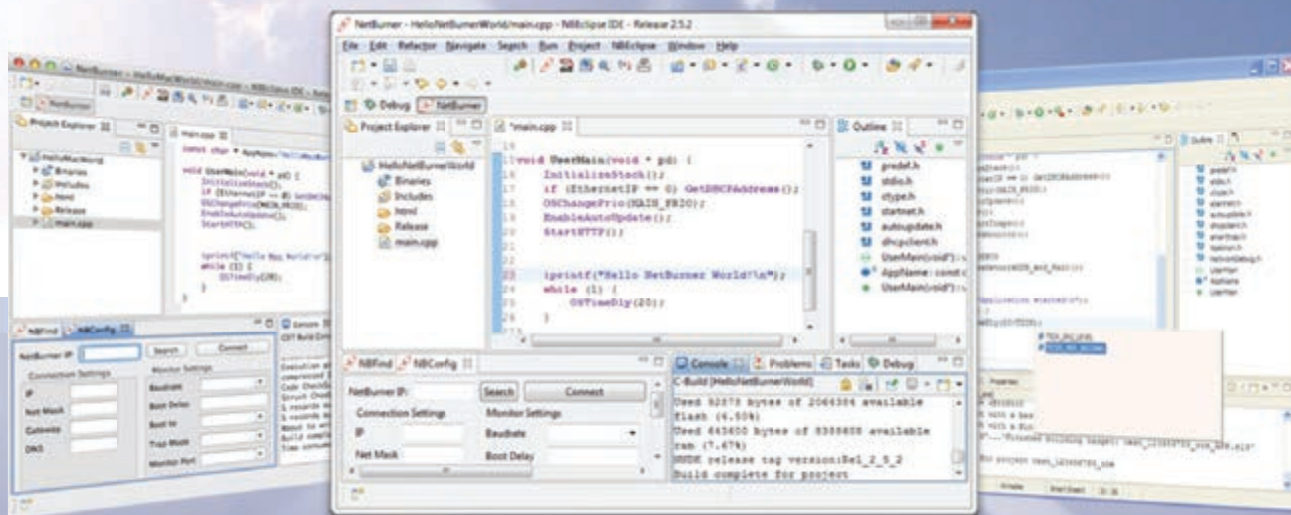
NANO54415

\$69⁰⁰
Qty. 100



MOD54415

\$89⁰⁰
Qty. 100



Quickly create and deploy applications from your Mac or Windows PC

Low cost NetBurner development kits are available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kit includes platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, flash file system, Eclipse IDE, debugger, and cables. The kit enables you to communicate with peripherals that use SD/MMC Flash Card (including SDHC), SPI, I²C, or the general purpose digital I/O interface. The NetBurner security suite option includes SSH v1, v2 and SSL support.

Development Kit for MOD54415
Part No. NNDK-MOD54415-KIT
\$99.00 for a limited time

Development Kit for NANO54415
Part No. NNDK-NANO54415-KIT
\$99.00



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





DESIGNSPARK



NEW. FREE. MODELSOURCE

OVER 80,000 FREE SCHEMATIC AND PCB SYMBOLS
IN MORE THAN 20 FORMATS, INCLUDING PADS,
ORCAD, ALTIUM AND CADSTAR.

Discover today at
www.designspark.com

UNIQUE
RESOURCES BY 

FORWARD PROGRESS

As you might have noticed, parts of this issue look a bit different than the publication you're used to reading. You can see a slightly updated layout, some different colors, and a few new sections. We've made these changes to reflect where we are today and where we're taking this magazine in the months to come. It's all about forward progress. Here are the broad strokes:

FRESHENED UP LAYOUT

We're planning an exciting layout redesign for 2013. The layout will be modern, clean, and engaging, but its fonts and colors won't distract you from what you're reading—professional engineering content. Since the new layout is still an issue or two away, we're presenting you with this freshened up issue to mark the transition to 2013. We hope you like the changes.

CLIENT PROFILES

On page 20 you'll find a new section that will appear frequently in the coming months. The purpose of our client profiles is to shine a light on one company per month and bring you an exclusive offer for useful products or services.

TECH THE FUTURE

Last month we ran Steve Ciarcia's final "Priority Interrupt" editorial. This month we're introducing a new section, "Tech the Future." The EE/ECE community is on the verge of major breakthroughs in the fields of microcomputing, wireless communication, robotics, and programming. Each month, we'll use page 80 to present some of the fresh ideas, thought-provoking research projects, and new embedded design-related endeavors from innovators who are working on the groundbreaking technologies of tomorrow.

CC25

You'll soon have *Circuit Cellar's* 25th ("CC25") anniversary issue in your hands or on your PCs or mobile devices. Here are just a few of the exciting topics in the issue: *Circuit Cellar* in 1988, design/programming tips, engineers' thoughts on the future of embedded tech, and much more. It's going to be a classic.

Well, there's certainly a lot of publishing-related innovation going on at our headquarters. And I know you're equally busy at your workbenches. Just be sure to schedule some quiet time this month to read the articles in this issue. Perhaps one of our authors will inspire you to take on your first project of the new year. We feature articles on topics ranging from an MCU-based helicopter controller to open-source hardware to embedded authentication to 'Net-based tools for energy efficiency. Enjoy!

cj@circuitcellar.com



CIRCUIT CELLAR®

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

EDITORIAL CALENDAR

ISSUE	THEME
270 January	Embedded Applications
271 February	Wireless Communications
272 March	Robotics
273 April	Embedded Programming
274 May	Measurement & Sensors
275 June	Communications
276 July	Internet & Connectivity
277 August	Embedded Development
278 September	Data Acquisition
279 October	Signal Processing
280 November	Analog Techniques
281 December	Programmable Logic

Analog Techniques: Projects and components dealing with analog signal acquisition and generation (e.g., EMI/RF reduction, high-speed signal integrity, signal conditioning, A/D and D/A converters, and analog programmable logic)

Communications: Projects that deal with computer networking, human-to-human interaction, human-to-computer interaction, and electronic information sharing (e.g., speech recognition, data transmission, Ethernet, USB, I²C, and SPI)

Data Acquisition: Projects, technologies, and algorithms for real-world data gathering and monitoring (e.g., peripheral interfaces, sensors, sensor networks, signal conditioning, ADCs/DACs, data analysis, and post-processing)

Embedded Applications: Projects that feature embedded controllers and MCU-based system design (e.g., automotive applications, test equipment, simulators, consumer electronics, real-time control, and low-power techniques)

Embedded Development: Tools and techniques used to develop new hardware or software (e.g., prototyping and simulation, emulators, development tools, programming languages, HDL, RTOSes, debugging tools, and useful tips)

Embedded Programming: The software used in embedded applications (e.g., programming languages, RTOSes, file systems, protocols, embedded Linux, and algorithms)

Internet & Connectivity: Applications that deal with connectivity and Internet-enabled systems (e.g., networking chips, protocol stacks, device servers, and physical layer interfaces)

Measurement & Sensors: Projects and technologies that deal with sensors, interfaces, and actuators (e.g., one-wire sensors, MEMS sensors, and sensor interface techniques)

Programmable Logic: Projects that utilize FPGAs, PLDs, and other programmable logic chips (e.g., dynamic reconfiguration, memory, and HDLs)

Robotics: Projects about robot systems, devices capable of repeating motion sequences, and MCU-based motor control designs (e.g., mobile robots, motor drives, proximity sensing, power control, navigation, and accelerometers)

Signal Processing: Projects and technology related to the real-time processing of signals (e.g., DSP chips, signal conditioning, ADCs/DACs, filters, and comparisons of RISC, DSP, VLIW, etc.)

Wireless Communications: Technology and methods for going wireless (e.g., radio modems, Wi-Fi/IEEE 802.11x, Bluetooth, ZigBee/IEEE 802.15.4, cellular, infrared/IrDA, and MCU-based wireless security applications)

UPCOMING IN CIRCUIT CELLAR

FEATURES

Build a Digital Dip Meter, by Stuart Ball

3-D Paint: A Hardware/Software Package for 3-D Drawing, by William Meyers and Guo Jie Chin

Digital Camera Controller (Part 3), by Richard Lord

COLUMNS

Arduino Survival Guide: Digital I/O, by Ed Nisley

QR Coding for Engineers, by Jeff Bachiochi

Fault-Tree Analysis, by George Novacek

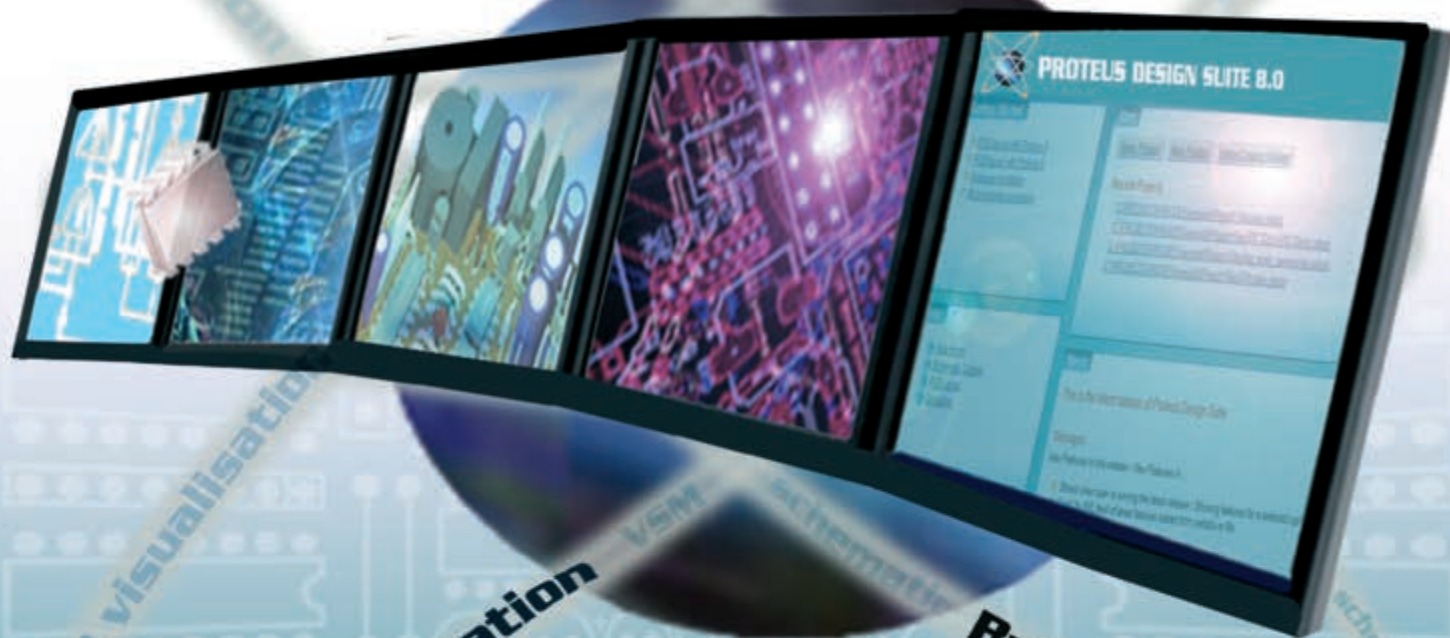
Concurrency in Embedded Systems (Part 5), by Bob Japenga

An Introduction to Standing Waves, by Robert Lacoste

CAD CONNECTED

PCB Layout

Microprocessor Simulation



3D visualisation

Built-in IDE

PROTEUS DESIGN SUITE VERSION 8

Featuring a brand new application framework, common parts database, live netlist and 3D visualisation, a built in debugging environment and a WYSIWYG Bill of Materials module, Proteus 8 is our most integrated and easy to use design system ever. Other features include:

- Hardware Accelerated Performance.
- Unique Thru-View™ Board Transparency.
- Over 35k Schematic & PCB library parts.
- Integrated Shape Based Auto-router.
- Flexible Design Rule Management.
- Polygonal and Split Power Plane Support.
- Board Autoplacement & Gateswap Optimiser.
- Direct CAD/CAM, ODB++, IDF & PDF Output.
- Integrated 3D Viewer with 3DS and DXF export.
- Mixed Mode SPICE Simulation Engine.
- Co-Simulation of PIC, AVR, 8051 and ARM MCUs.
- Direct Technical Support at no additional cost.

labcenter  www.labcenter.com
Electronics

Visit our website or
phone 866 499-8184
for more details

Labcenter Electronics North America, 411 Queen St. Newmarket, Ont. Canada L3Y 2G9.
Email: info@labcenter-electronics.com Tel 905.898.0665 Fax 905.898.0683

02 EDITOR'S LETTER Forward Progress

By C. J. Abate

10 NEW PRODUCTS

18 MEMBER PROFILE

19 TEST YOUR EQ

20 CLIENT PROFILE

22 MCU-Based Altitude Control

An IR Protocol for Helicopter Commands

By Akshay Dhawan and Sergio Biagioni

30 Open-Source Hardware Development

By Josh Davis, Tomas Carvalho e Silva, and John Vaughn

38 DIY Function Generator

By Larry Cicchinelli

46 QUESTIONS & ANSWERS Sensory Innovation

An Interview with Stephan Lubbers

By Nan Price

50 THE CONSUMMATE ENGINEER Failure Mode and Criticality Analysis

By George Novacek

54 EMBEDDED SECURITY Embedded Authentication

By Patrick Schaumont

60 FROM THE BENCH Web-Based Tools for Energy Efficiency

By Jeff Bachiochi

70 FROM THE ARCHIVES Gotcha!

Alarming the Alarm System

By Steve Ciarcia and Jeff Bachiochi

(Circuit Cellar 95, June 1998)

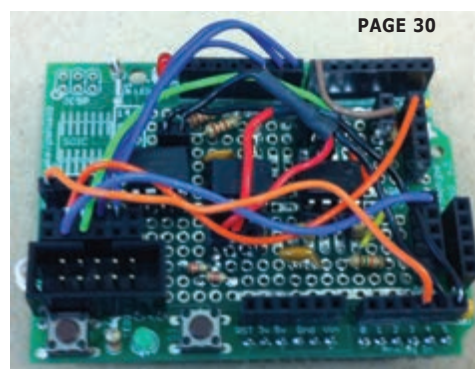
76 CROSSWORD

80 TECH THE FUTURE Can MoS₂ Outperform Silicon?

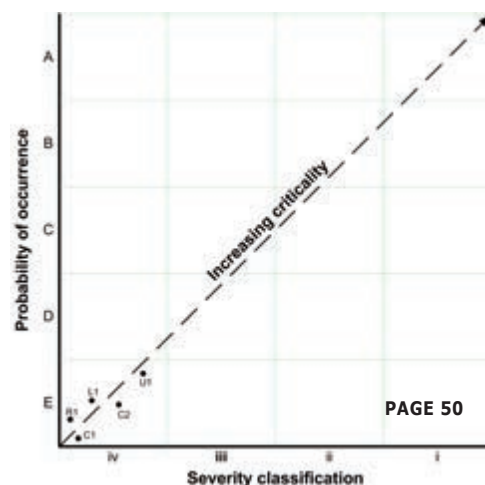
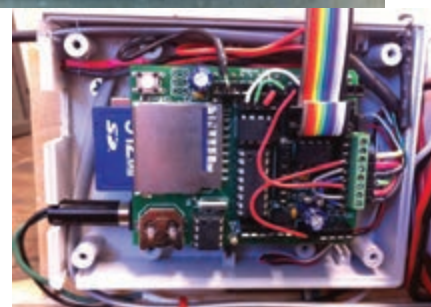
By Saptarshi Das



PAGE 22



PAGE 30



PAGE 50



mouser.com
Distributing semiconductors and electronic
components for design engineers.

Authorized Distributor



ORE.

mouser.com

The widest selection of the newest products.

The Newest Products for Your Newest Designs®



a tti company

THE TEAM

FOUNDER:	Steve Ciarcia	PROJECT EDITORS:	Ken Davidson, David Tweed
EDITOR-IN-CHIEF:	C. J. Abate	PUBLISHER:	Hugo Van haecke
ASSOCIATE EDITOR:	Nan Price	ASSOCIATE PUBLISHER:	Shannon Barraclough
CONTRIBUTING EDITORS:	Jeff Bachiochi, Bob Japenga, Robert Lacoste, George Martin, Ed Nisley, George Novacek, Patrick Schaumont	ART DIRECTOR:	KC Prescott
		CONTROLLER:	Jeff Yanco
		CUSTOMER SERVICE:	Debbie Lavoie
		ADVERTISING COORDINATOR:	Kim Hopkins

THE NETWORK



OUR INTERNATIONAL TEAMS

**United Kingdom**

Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

**Spain**

Eduardo Corral
+34 91 101 93 85
e.corral@elektor.es

**India**

Sunil D. Malekar
+91 9833168815
ts@elektor.in

**USA**

Hugo Van haecke
+1 860 875 2199
h.vanhaecke@elektor.com

**Italy**

Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

**Russia**

Nataliya Melnikova
8 10 7 (965) 395 33 36
nataliya-m-larionova@yandex.ru

**Germany**

Ferdinand te Walvaart
+49 (0)241 88 909-0
f.tewalvaart@elektor.de

**Sweden**

Wisse Hettinga
+31 (0)46 4389428
w.hettinga@elektor.com

**Turkey**

Zeynep Köksal
+90 532 277 48 26
zkoks@beti.com.tr

**France**

Denis Meyer
+31 (0)46 4389435
d.meyer@elektor.fr

**Brazil**

João Martins
+351214131600
joao.martins@editorialbolina.com

**South Africa**

Johan Dijk
+27 78 2330 694 / +31 6 109 31 926
J.Dijk@elektor.com

**Netherlands**

Harry Baggen
+31 (0)46 4389429
h.baggen@elektor.nl

**Portugal**

João Martins
+351214131600
joao.martins@editorialbolina.com

**China**

Cees Baay
+86 (0)21 6445 2811
CeesBaay@gmail.com

Issue 270 January 2013

ISSN 1528-0608

Subscriptions

Circuit Cellar, P.O. Box 462256, Escondido, CA 92046
E-mail: circuitcellar@pcspublink.com
Phone: 800.269.6301, Internet: www.circuitcellar.com
Address Changes/Problems: circuitcellar@pcspublink.com

Postmaster: Send address changes to Circuit Cellar, P.O. Box 462256, Escondido, CA 92046.

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 111 Founders Plaza, Suite 300, East Hartford, CT 06108. Periodical rates paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate USA and possessions \$50, Canada \$65, Foreign/ROW \$75. All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank.

Cover photography by Chris Rakoczy—www.rakoczyphoto.com

US Advertising

Strategic Media Marketing, Inc.
2 Main Street, Gloucester, MA 01930 USA
Phone: 978.281.7708, Fax: 978.281.7706, E-mail: peter@smmarketing.us
Internet: www.circuitcellar.com
Advertising rates and terms available on request.

New Products: New Products, Circuit Cellar, 111 Founders Plaza, Suite 300, East Hartford, CT 06108, E-mail: newproducts@circuitcellar.com

MEMBERSHIP COUNTER

We
now have

265646

members
in

83

countries.

Not a member yet?

Sign up at www.circuitcellar.com

SUPPORTING COMPANIES

All Electronics Corp.	78	Elprotronic, Inc.	52	MCC, Micro Computer Control	77
AP Circuits	57	EMAC, Inc.	57	Microchip Technology, Inc.	25
Apex Expo 2013.	18	Embedded World 2013	21	Microengineering Labs, Inc.	78
Apox Controls, LLC.	79	ESC Design East Summit	27	Mosaic Industries, Inc.	77
ARM.	41	EzPCB.	67	Mouser Electronics, Inc.	5
Artila Technologies, Inc.	19	Front Panel Express	73	NetBurner	C2
Beta Layout, Ltd.	49	FTDI Chip.	C3	Newark element14	11
BusBoard Prototype Systems.	78	Grid Connect, Inc.	73	Pololu Corp.	53
Butterfly Network, Inc.	67	HannoWare	79	Reach Technology, Inc.	79
Circuit Cellar 25 th Anniversary USB	33	Humandata, Ltd.	15	Saelig Co., Inc.	42
Cleverscope.	49	Imagineering, Inc.	C4	Scidyne.	77
Comfile Technology	13	Ironwood Electronics	52	Technologic Systems	8, 9
Custom Computer Services	77	Jeffrey Kerr, LLC.	43	Techsol Engineering, Inc.	79
DesignSpark	1	Kozio, Inc.	17	Tern, Inc.	77
Earth Computer Technologies	20	Labcenter Electronics	3	Triangle Research International, Inc.	78
Elektor	59	Lakeview Reasearch	43		
Elektor	68,69	Maxbotix, Inc.	79		

Not a supporting company yet?

Contact Peter Wostrel (peter@smmarketing.us, Phone 978.281.7708, Fax 978.281.7706)
to reserve your own space for the next issue of our member's magazine.

Head Office

Circuit Cellar, Inc. 111 Founders Plaza, Suite 300, East Hartford, CT 06108
Phone: 860.875.2199

Copyright Notice

Entire contents copyright © 2012 by Circuit Cellar, Inc. All rights reserved.
Circuit Cellar is a registered trademark of Circuit Cellar, Inc. Reproduction
of this publication in whole or in part without written consent from Circuit
Cellar Inc. is prohibited.

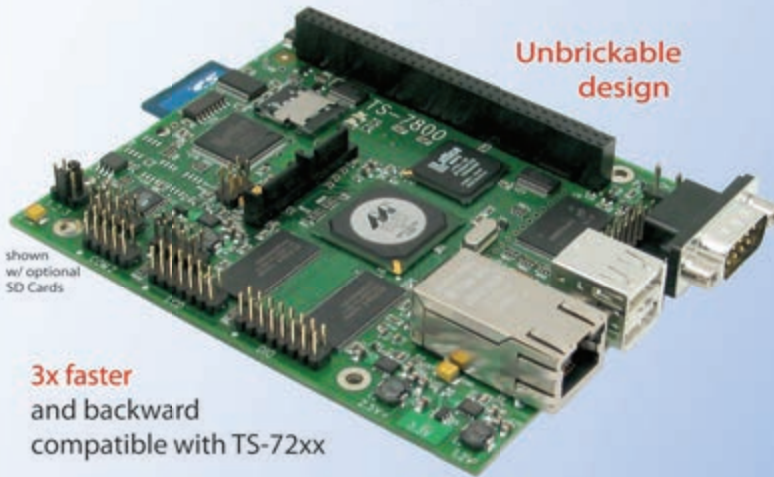
Disclaimer

Circuit Cellar® makes no warranties and assumes no responsibility or
liability of any kind for errors in these programs or schematics or for the
consequences of any such errors. Furthermore, because of possible
variation in the quality and condition of materials and workmanship of
reader-assembled projects, Circuit Cellar® disclaims any responsibility for
the safe and proper function of reader-assembled projects based upon or
from plans, descriptions, or information published by Circuit Cellar®.

The information provided by Circuit Cellar® is for educational purposes.
Circuit Cellar® makes no claims or warrants that readers have a right
to build things based upon these ideas under patent or other relevant
intellectual property law in their jurisdiction, or that readers have a
right to construct or operate any of the devices described herein under
the relevant patent or other intellectual property law of the reader's
jurisdiction. The reader assumes any risk of infringement liability for
constructing or operating such devices.

Embedded Systems

High-End Performance with Embedded Ruggedness



shown
w/ optional
SD Cards

Unbrickable
design

3x faster
and backward
compatible with TS-72xx

TS-7800 500MHz ARM9

- 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
- 10 serial ports
- 5 ADC (10-bit)
- Sleep mode uses 200 microamps
- Boots Linux 2.6 in 0.7 seconds
- Linux 2.6 and Debian by default
- 2 SD sockets
- 110 GPIO
- 2 SATA ports

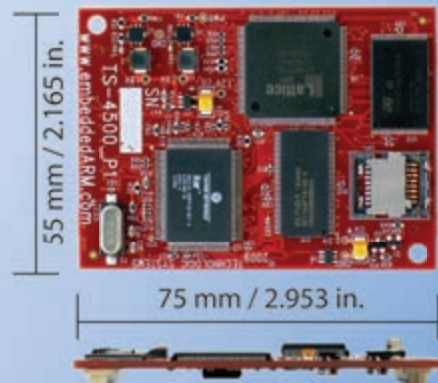
\$229
qty 100

\$269
qty 1

TS-SOCKET Macrocontrollers Jump Start Your Embedded System Design

TS-SOCKET Macrocontrollers are CPU core modules that securely connect to a baseboard using the TS-SOCKET connector standard. COTS baseboards are available or design a baseboard for a custom solution with drastically reduced design time and complexity. Start your embedded system around a TS-SOCKET Macrocontroller to reduce your overall project risk and accelerate time to market. Current TS-SOCKET products include:

- TS-4200: Atmel ARM9 with super low power
- TS-4300: 600MHz ARM9 and 25K LUT FPGA
- TS-4500: Cavium ARM9 at very low cost
- TS-4700: 800MHz Marvell ARM with video
- TS-4800: 800MHz Freescale iMX515 with video
- Several COTS baseboards for evaluation & development



- Dual 100-pin connectors
- Secure connection w/ mounting holes
- Common pin-out interface
- Low profile w/ 6mm spacing

series
starts at
\$92
qty 100

\$139
qty 1

- 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

Touch Panel Computers 800MHz with Video Acceleration

- Resistive touchscreen, LED backlit display
- Gasketed construction
- Tough powder coated finish
- Fanless operation from -20°C to +70°C
- 800MHz ARM CPU
- 256MB RAM, 256MB SLC XNAND Drive
- MicroSD slot
- 5K LUT programmable FPGA
- Dual Ethernet, USB ports
- CAN, RS-232 ports, RS-485
- Mono speaker on PCB, stereo audio jack
- SPI, DIO



Fully enclosed TPC
available Q3

NEW!



series
starts at

\$415
qty 100

\$479
qty 1



picture of TS-8820-BOX

NEW!

series
starts at

\$199
qty 100

\$229
qty 1

Technologic Systems now offers three powerful computers targeting industrial process control. Implement an intelligent automation system at low cost with a minimal number of components.

Industrial Controllers Powerful, Rugged, Affordable

- 250MHz (ARM9) or 800MHz (ARM9 or Cortex-A8) CPU
- Fast startup (under 3 seconds)
- Fanless operation from -20°C to +70°C
- User-programmable opencore FPGA
- Program in Ladder Logic or C
- Debian Linux
- Modbus support
- PoE capable 10/100 Ethernet, USB 2.0 Host Ports
- Industrial screw-down connectors
- Opto-Isolated DIO, Digital Counters, Quadrature
- Up to 46 DIO with PWM
- Opto-Isolation available for RS-232, RS-485 and CAN
- DIN mount option



We use our stuff.

Visit our TS-7800 powered website at

www.embeddedARM.com



SDR DEMONSTRATOR FOR WIRELESS DATA APPLICATIONS

The **DE9941** is a credit card-sized demonstration board for a complete linear modulation-based, software-defined radio (SDR) for wireless data. The board features a small, low-cost design with minimal components/values.

The **CMX998**, **CMX994**, and **CMX7164** are highly integrated RF and baseband devices. The CMX998 is a Cartesian loop transmitter IC providing power amplifier linearization for non-constant envelope/linear systems. The CMX994 is a direct conversion receiver, providing direct conversion from RF down to I/Q baseband. Based on CML's FirmASIC technology, the CMX7164 multi-mode wireless data modem IC offers multiple modulation schemes that can be uploaded into the device via Function Image.

You can use the DE9941 to demonstrate transmit and receive performance with 4/16/64 QAM linear modulation and constant-envelope modulation schemes (e.g., 2/4-level FSK and GMSK), which are also available on the CMX7164 multi-mode wireless data modem. Combined with CML's PE0002 interface board, a full transceiver can be demonstrated using a Function Image and control scripts.

The DE9941 board includes a 1-W power amplifier and RF performance designed to be compliant with EN 302 561 and EN 300 113. Scripts are available on the CML website to enable a quick-start evaluation of the board and components.

Contact CML for pricing.

CML Microsystems, Plc
www.cmlmicro.com



DIGITAL-DIMMING SSL LED DRIVERS WITH UP TO 25-W OUTPUT POWER

The **iW3616** (12 W) and the **iW3617** (25 W) LED driver ICs are based on iWatt's third-generation digital AC/DC solid-state lighting (SSL) LED driver platform. The drivers are designed for 120 V/230 VAC offline LED lighting bulbs and fixtures.

The driver ICs build on iWatt's 15-W iW3614, a Flickerless LED driver. The Flickerless technology features an intelligent dimmer interface with digital dimming algorithms that automatically detect and adapt to the dimmer type for smooth, flicker-free dimming.

Compared to the iW3614, the iW3616 and the iW3617 increase the output power to 25 W, lower the bill of materials (BOM) cost by 10% to 20%, and expand the dimmer compatibility to offer compatibility with installed dimmers (e.g., residential TRIAC dimmers and sophisticated digital dimmers).

The driver ICs are capable of greater than 0.95 power factor, less than 15% low total harmonic distortion (THD), and greater than 85% efficiency. The iW3616 and the iW3617 exceed global energy and LED lighting standard requirements, including the European Union IEC61000-3-2, NEMA SSL 6 dimming, and Zhaga hot-plug interchangeability standards.

Contact iWatt for pricing.

iWatt, Inc.
www.iwatt.com



NEW PRODUCTS



ARM

Honeywell

Sensing and Control



Expertise Applied | Answers Delivered

molex



Tektronix



COMPLETE ENGINEERING SOLUTIONS

Start here.

"The navigation and
ordering process are
easy to work. Thanks."

– Richard, Newark element14 customer

At Newark element14, all your engineering needs come together in one source—vast product range from world-class brands, fast online search, seamless purchasing tools, resources and services, one-on-one support, and a community of experts. Here, you'll find simpler, smarter and faster ways to do business.



Newark

element14

HOW MAY WE HELP YOU TODAY?



COMMUNITY: element14.com

WEBSITE: newark.com

PHONE: 1.800.463.9275

LEARN MORE: newark.com/together



POWER MANAGEMENT ICs FOR PORTABLE APPLICATIONS

The **NCP6924** and the **NCP6914** are power-management integrated circuits (PMICs) for battery-powered systems, such as smartphones, tablets, digital cameras, GPSes and other portable electronics. The PMICs are designed to optimize system efficiency and save battery life.

The NCP6924 integrates two high-efficiency 800-mA, 3-MHz step-down DC-DC converters and four low dropout (LDO) voltage regulators for a 105- μ A total quiescent current. The NCP6914 integrates a single high-efficiency, 800-mA, 3-MHz step-down DC-DC converter and four LDO regulators to deliver a 72- μ A low-quiescent current. Featuring five and six voltage rails, respectively, both of these devices are well sized to supply power to mixed-signal modules (e.g., cameras) or to complement power distribution to an application processor (AP) under minimum supervision.



The PMICs' programming flexibility is provided through a 400-kHz/3.4-MHz I²C interface, which controls power-up sequencing, enables and disables output power, and controls individual active output discharges. The DC-DC converters' dynamic voltage scaling enables the system to adjust core or I/O voltages to load profiles in situations such as a system entering Sleep mode. In addition, to reduce overall power loss, the on-board LDO regulators can be directly supplied by one of the integrated DC-DC converters.

The NCP6924 and the NCP6914 are offered in 2.45 mm \times 2.05 mm and 1.76 mm \times 2.05 mm, respectively. They both feature wafer-level chip scale packages (WLCSs) with a 0.4-mm pitch. The NCP6924 costs **\$1.85** per unit. The NCP6914 costs **\$1.35** per unit, both in 10,000-unit quantities.

ON Semiconductor
www.onsemi.com

SINGLE-SUPPLY, 12-BIT, 300-ksps SERIAL-OUTPUT ADC

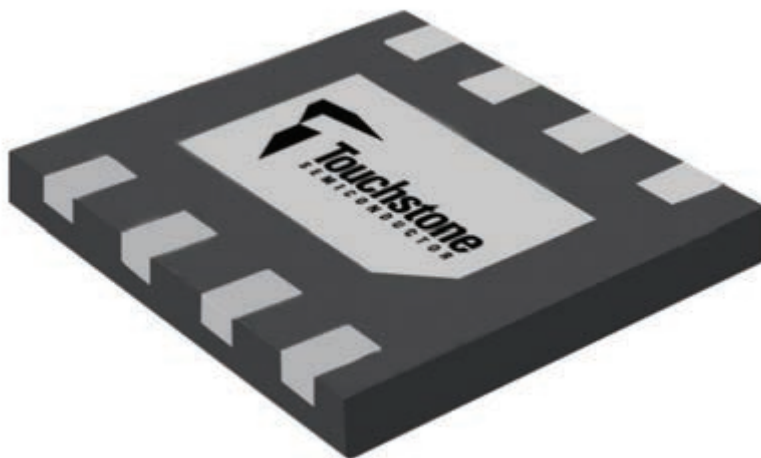
The **TS7003** is a single-supply, single-channel, 12-bit ADC that is low power, easy to use, and has a small footprint. The ADC is well suited for PCB-space-conscious, low-power, remote-sensor, and data-acquisition applications, such as process control and factory automation, data and low-frequency signal acquisition, portable data logging, pen digitizers and tablet computers, medical instrumentation, and battery-powered instruments.

Operating from 2.7-to-3.6-V supplies, the TS7003 consumes less than 2.8 mW when converting at 300 ksps. The TS7003's three-wire serial interface directly connects to any microcontroller or interface-compatible computing device. The ADC doesn't require separate, external logic.

An external serial-interface clock provides you with full control over the TS7003's conversion process and its output shift register operation.

The converter features a pin-compatible, single-channel, higher-speed upgrade to Maxim Integrated Products's MAX1286 ADC. The TS7003 also features internal 10-MHz track-and-hold, a high-speed serial digital interface, and a 0.2- μ A low-supply current in shutdown. It is fully specified over the -40°C-to-85°C temperature range and is available in a low-profile, eight-pin 3 mm \times 3 mm TDFN package with an exposed backside paddle.

The TS7003 costs **\$0.95** each in 1,000-piece quantities.



Touchstone Semiconductor, Inc.
www.touchstonesemi.com

NEW PRODUCTS



Windows CE 6.0 Touch Controller

CUWIN



The CUWIN is a series of Windows CE touch controllers that are more cost-effective than a PC, but with more features than an HMI touch screen. Create sophisticated applications with C++ or any .Net language.

533MHz ARM CPU
128MB SDRAM & Flash
SD Card Support
Ethernet, RS-232/485
USB, Audio Out
Windows Embedded CE 6.0

The CUPC is a series of industrial touch panels with all the features of a modern PC for the most feature-rich user experience.

ATOM N270 1.6GHz CPU
2GB RAM
320GB HDD
SD Card Support
Color Display (1024 x 768)
RS-232, Ethernet
USB, Audio Out
Windows XP/XP Pro

CUPC



Industrial Touch Panel PC with ATOM Processor



C-Programmable Modular Industrial Controller

MOACON

The MOACON is a modular, C programmable, ARM-based automation controller designed for industrial environments.

Choose from a diverse, feature-rich selection of modules including:
Digital I/O
Analog I/O
RS-232/485
Motor Control
Ethernet
High-speed Counter & PWM

BASIC with LADDER LOGIC CONTROLLER



only \$29

New

Integrated CUBLOC Controller and I/O Board

CB210

The CB210 is an inexpensive, integrated CUBLOC controller and I/O board programmable in both BASIC and Ladder Logic.

I/O Ports x20 10-bit A/D x6
PWM x3 RS-232 x1
80KB Program Memory 3K Data Memory

The CUSB is a series of compact, CUBLOC-integrated, industrial I/O boards programmable in both BASIC and Ladder Logic.

CUBLOC CB280 Core Module
80KB Program Memory
3KB Data Memory
DC 24V Inputs x9~16
A/D Inputs x2~6
Relay Outputs x6~16
PWM x2~6
High-speed Counters x0~2
RS-232/485 x1~2

CUSB

Integrated Industrial Controllers



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

MCUs PROVIDE SIMULTANEOUS INVERTER CONTROL FOR UP TO THREE MOTORS

The **RX63T** microcontroller family now has higher pin counts, larger memory versions, and expanded functions, including USB 2.0, 10-bit DAC, and faster ADC with a 0.5- μ s minimum conversion time. The microcontrollers increase the number of channels of timers and serial I/Fs. The new RX63T family also inherits the features of the RX62T and existing RX63T. The microcontrollers help reduce system costs and power consumption in applications including industrial motor control, air conditioners, washing machines, and solar-inverter solutions.

The RX63T platform is built around the high-performance 32-bit RX CPU core featuring a 100-MHz operating frequency with zero-wait-state flash to achieve a 165-Dhrystone MIPS (DMIPS) performance and 312 CoreMark score. All RX63T and RX62T microcontrollers include DSP functionality.

The microcontroller's key features include a multi-function timer pulse unit 3 (MTU3), which is specialized for motor control. The timer can drive up to two three-phase motors and a 16-bit general-purpose PWM timer (GPT) capable of driving a three-phase motor. The GPT can independently control four channels of single-phase inverter. The timer uses the same clock as the CPU at 100 MHz, enabling it to achieve a range of PWM output waveform width measurements with resolutions as small as 10 ns.

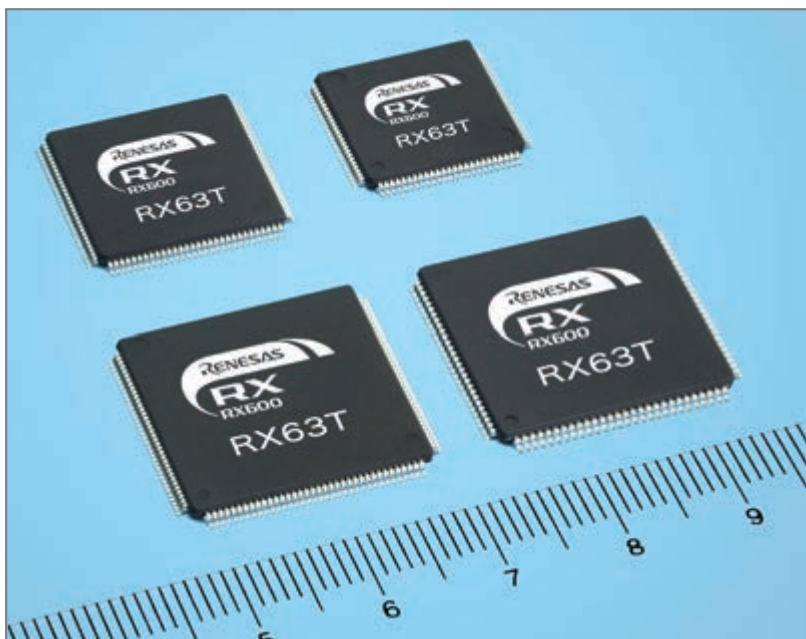
The RX63Ts also reduces the cost of motor-vector control applications and increases their usability. The microcontrollers feature two 12-bit ADC units that can capture analog input values with a 1- μ s minimum conversion time. Because they are capable of simultaneously sampling three input channels, the 12-bit ADCs can be easily used for sensorless vector-control methods (e.g., three-shunt or single-shunt current detection). Continuous A/D conversion is supported by the double data registers installed in the 12-bit A/D module.

The microcontroller family incorporates features to help meet safety standards (e.g., IEC60730), such as an independent watchdog with its own on-chip clock source (IWDT). The RX63T family also supports USB 2.0 host/device/OTG, CAN connectivity and a single-precision floating-point unit (FPU).

The microcontrollers provide a scalable memory solution from 256-to-512-KB flash with up to 48 KB of embedded SRAM. They also include an independent 32 KB of data flash memory with a background operation (BGO) function that enables data to be written while a program is executing. The embedded flash memory is based on Renesas's metal oxide nitride oxide silicon (MONOS) technology, which can be accessed without wait-state insertion. This enables a 1.65-DMIPS/MHz maximum performance level at any CPU frequency, without any limitation from flash technology.

Pricing for the RX63T family of microcontrollers starts at **\$4.10** per unit in 10,000-unit quantities.

Renesas Electronics Corp.
www.renesas.com



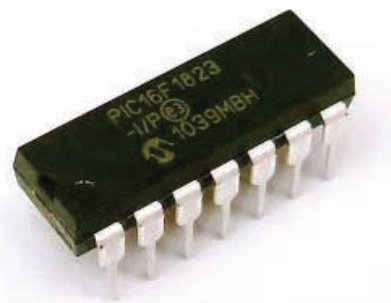
PS/2 KEYBOARD-TO-ASCII CONVERTER CHIP

The **KB1** is a PS/2 keyboard-to-ASCII converter chip for embedded microcontroller applications. The easy-to-use chip is based on the Microchip Technology PIC16F1823 extended mid-range processor.

The chip's flexible design offers a pin-selectable choice of UART, SPI, or I²C interface modes. The KB1 has a seven-character buffer and generates interrupts in SPI and I²C modes. The KB1's data manual is available online and includes example PIC Assembly language routines to communicate with the KB1 in SPI and I²C modes. The converter chip is available as a 14-pin dual in-line package (DIP).

A single KB1 costs **\$6.25**.

Lucid Technologies
www.lucidtechnologies.info



NEW PRODUCTS

SAVING COST=TIME with readily available FPGA boards

- Basic and simple features, single power supply operation
- Quality and reliability is provided by years of sales
- Same board size and connector layout – ACM/XCM series
- All stocked items are ready to be shipped immediately
- Over 100 varieties of FPGA/CPLD boards are available
- Customizing speed grade and/or any features are possible
- Free download technical documents before purchasing
- High quality and highly reliable FPGA/CPLD boards from Japan
- Almost all products are RoHS compliance

ALTERA FPGA Board

Cyclone IV E F780 FPGA board

ACM-204 series

Cyclone IV E **SDRAM**

EP4CE30F29C8N
EP4CE40F29C8N
EP4CE115F29C8N
Credit card size (86 x 54 mm)

RoHS compliant



Arria II GX F572 FPGA board

ACM-025 series

Arria II GX **DDR2**

EP2AGX45DF25C6N
EP2AGX65DF25C6N
EP2AGX95DF25C6N
EP2AGX125DF25C6N
Credit card size (86 x 54 mm)

RoHS compliant



CycloneIV GX F484 FPGA board

ACM-024 series

Cyclone IV GX **DDR2** **SIF40**

EP4CGX50CF23C8N
EP4CGX75CF23C8N
EP4CGX110CF23C8N
EP4CGX150CF23C7N
Credit card size (86 x 54 mm)

RoHS compliant



MAX II T144 CPLD board

ACM-302-1270

MAX II

EPM1270T144C5N
Compact size (54 x 53 mm)

RoHS compliant



XILINX FPGA Board

Spartan-6 FGG484 FPGA board

XCM-018/018Z series

Spartan-6 **MRAM** **DDR2**

XC6SLX45-2FGG484C
XC6SLX75-2FGG484C
XC6SLX100-2FGG484C
XC6SLX150-2FGG484C
Credit card size (86 x 54 mm)

RoHS compliant



Virtex-5 FFG676 FPGA board

XCM-109 series

Virtex-5 **SDRAM**

XC5VLX30-1FFG676C
XC5VLX50-1FFG676C
XC5VLX85-1FFG676C
XC5VLX110-1FFG676C
Compact size (43 x 54 mm)

RoHS compliant



Spartan-6 FGG676 FPGA board

XCM-206 series

Spartan-6 **MRAM** **DDR2**

XC6SLX100-2FGG676C
XC6SLX150-2FGG676C
Credit card size (86 x 54 mm)

RoHS compliant



XCM-206Z series

Spartan-6

XC6SLX75-2FGG676C
XC6SLX100-2FGG676C
XC6SLX150-2FGG676C
Credit card size (86 x 54 mm)

RoHS compliant



Spartan-3A FTG256 FPGA board

XCM-305 series

Spartan-3A **MRAM**

XC3S700A-4FTG256C
XC3S1400A-4FTG256C
Compact size (54 x 53 mm)

RoHS compliant



FPGA/CPLD Stamp Module PLCC68 Series

Easy and Quickly Mountable Module

FPGA Module IC socket mountable

- 50 I/Os (External clock inputs are available)
- 3.3V single power supply operation (Voltage converters for auxiliary power supply are built-in)
- Separated supply-inputs: Core, I/O drivers
- JTAG signal
- All PLCC68 series have common pin assignment
- Very small size (25.3 x 25.3 [mm])
- RoHS compliance
- MADE IN JAPAN



XILINX PLCC68 Series

Spartan-6 PLCC68 FPGA Module

XP68-03

Spartan-6 **PLCC 68**

XC6SLX45-2CSG324C
3.3V single power supply operation
On-board oscillator, 50MHz
RoHS compliant



Spartan-3AN PLCC68 FPGA Module

XP68-02

Spartan-3AN **PLCC 68**

XC3S200AN-4FTG256C
FPGA internal configuration ROM
Two User LEDs
On-board oscillator, 50MHz
RoHS compliant



ALTERA PLCC68 Series

Cyclone III PLCC68 FPGA Module

AP68-04

Cyclone III **PLCC 68**

EP3C25U256C8N
3.3V single power supply operation
On-board oscillator, 50MHz
RoHS compliant



Cyclone III PLCC68 FPGA Module

AP68-03

Cyclone III **PLCC 68**

EP3C10U256C8N
4Mbit Configuration Device
Two User LEDs
One User Switch(Slide)
On-board oscillator, 50MHz
RoHS compliant



Universal Board (Type2)

ZKB-106

- One for general power (3.3V 3A max) and the two variable outputs for Vccio (0.8V to 3.3, 3A max)
- For ACM/XCM-2 series FPGA boards
- Power Switch and LED
- Power input: DC5V/2.1 [mm] Jack/ Terminal Block (option)
- Board size : 156x184 [mm]
- 4 Layers PCB, Thru-hole



See all our products, A/D D/A conversion board, boards with USB chip from FTDI and accessories at :

www.hdl.co.jp/CC/

FULL-FEATURED TR-069 AUTO CONFIGURATION SERVER

tGem is a remote single management system for multi-vendor CPE deployment, which benefits from complete TR-069 support. tGem meets all relevant specifications to manage all services beyond triple play on home gateways, set-top boxes, VoIP phones, or femtocell devices.

All the data in the tGem occurs in real time. tGem can provide a historical overview of the data and the latest values from the dashboard that summarizes the population status for different factors (e.g., software upgrades, newly activated devices, alarms, etc.) to the status reports and monitoring available on each device.

You can use the dashboard to see the overall health of deployments and catch major problems as soon as they occur. The GUI and northbound interfaces are used to set, update, and access all data.

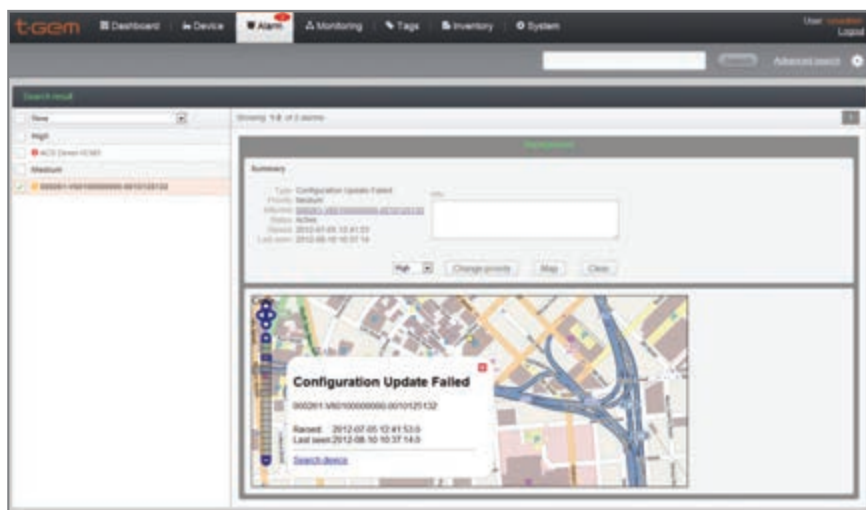
tGem can provide customized reports on devices' key performance indicators in real time or from the cache. This enables you to instantly know a device's last status, even if it has gone dark.

tGem comes equipped with two powerful modules. Service providers can use the Troubleshooting module to shorten subscriber call duration by presenting all the necessary data to the call center staff. It is unnecessary to connect to any command line or remote device GUI, all the information pertinent to the call (e.g., device history, KPI values, real-time status report, and diagnostics) are downloaded from the CPE.

The Alarms and Monitoring module can be used to monitor any parameter on one, several, or all devices over time. It

can enable alarms to trigger when certain levels (e.g., high or low) are reached. All parameters can be monitored, used to proactively detect issues, and detect a device's history when troubleshooting. Monitoring devices show all its power when coupled with the Alarms functionality. Besides triggering, when certain thresholds are attained for designated parameters, alarms can trigger on a number of events defined in tGem (e.g., failed software update, dropped connection, etc.). Each alarm can trigger an e-mail or a SNMP trap and propagate upward, ensuring no issues go unnoticed.

Contact Tilgin for pricing.



Tilgin
www.tilgin.com

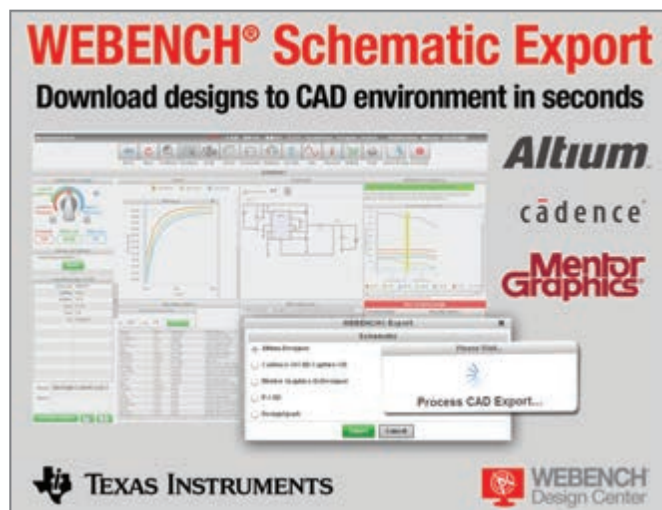
ONLINE TOOL FOR EXPORTING LED LIGHTING DESIGNS

WEBENCH Schematic Export was developed by Texas Instruments (TI) in cooperation with the makers of computer-aided design (CAD) development platforms. The online tool enables you to instantly export dynamically created power and LED lighting designs to platforms such as Cadence Design Systems's OrCAD/Capture CIS, Mentor Graphics's DxDesigner, and Altium formats (e.g., Altium Designer).

You can optimize and export a power-supply design created with TI's WEBENCH Power or LED Designer or a complete power-supply system created with WEBENCH Power or LED Architect via WEBENCH Schematic Export to one of five CAD development platforms. You can use WEBENCH Designer and Architect to stimulate design performance prior to export. WEBENCH Schematic Export is supported with schematic symbols for more than 30,000 components and is available in eight languages, so you can compare complete system designs.

To start a cost-free design, visit TI's WEBENCH design environment at www.ti.com/webench-pr.

Texas Instruments, Inc.
www.ti.com



NEW PRODUCTS

LOW-POWER RF MODULES

The **LMX-ISM-242** series of low-power, 2.4-GHz radios are available in dual in-line package (DIP) and surface-mount device (SMD) configurations. A long-range version, the LMX-ISM-242-LR, is also available.

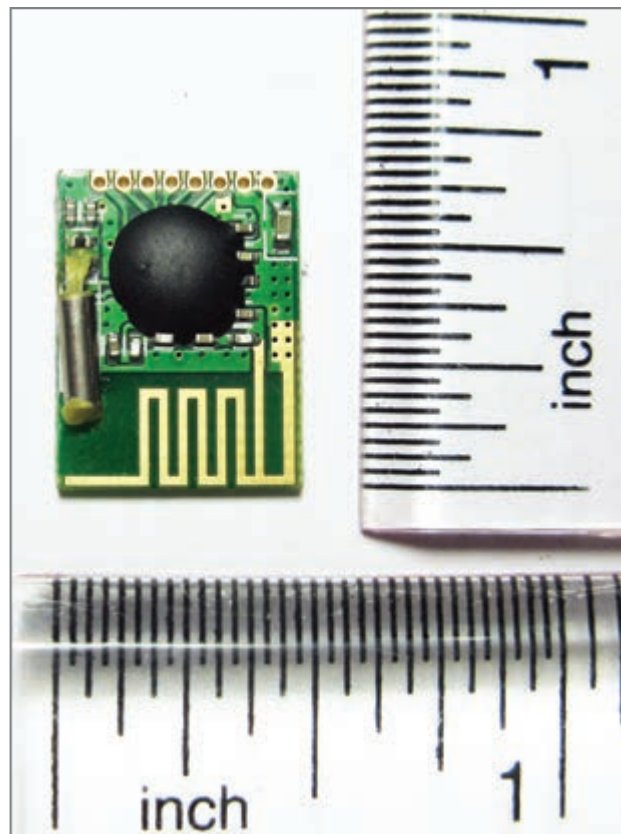
The radios operate in the 2.4-to-2.4835-GHz segment of the license-free industrial, scientific, and medical (ISM) radio bands. All LMX-ISM-242 family devices feature programmable output power and data rate. A low 1.9-to-3.6-V voltage rate enables the data radios to easily fit into many embedded applications, including scientific telemetry, building security, remote monitoring, remote control, home automation, and consumer electronics. Configuration information and payload data are passed to and from LMX-ISM-242 radios via an industry standard 8-MHz four-wire SPI.

The LMX-ISM-242 line is supported by the LMX-ISM-242 development kit, which includes hardware and firmware support for all LMX-ISM-242 devices. The LMX-ISM-242 development kit includes the LMX-ISM-242 driver and demo code as well as a real-time clock with an alarm to enable you to simulate a low-power node that reports at a scheduled time.

Lemos also offers a programmable LMX-ISM-242 data radio module that consists of either a standard LMX-ISM-242 or LMX-ISM-242-LR data radio managed by a powerful microcontroller.

Contact Lemos for pricing.

Lemos International, Inc.
www.lemosint.com



SIZE-CRITICAL HUMIDITY AND TEMPERATURE SENSOR

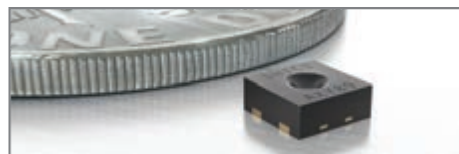
The tiny **SHTC1** humidity and temperature sensor is designed for mobile devices where size is a critical factor. The sensor measures 2 mm × 2 mm × 0.8 mm. The SHTC1 is based on Sensirion's CMOSens technology, which enables the sensor and the signal-processing electronics to be combined on a single silicon chip to achieve small device size.

The SHTC1 measures relative humidity over a 0%-to-100% RH range with a ±3% RH typical accuracy. The sensor's temperature measuring range is -30°C to 100°C with a typical accuracy of ±0.3°C. The

sensor is fully calibrated, features a digital I²C interface, is suitable for reflow soldering, and is compatible with standard industrial mass production processes for electronic modules.

Contact Sensirion for pricing.

Sensirion AG
www.sensirion.com



PRODUCTS

Is Your Hair on Fire?

Kozio's Verification and Test OS (VTOS™)
eliminates the risk of a fire before it starts!



Kozio is the industry's first embedded hardware test software that is user configurable for your design.

- Complete hardware test environment
- Quickly isolate design errors
- Test for manufacturability
- Up & running on your board in 30 minutes

Contact Kozio BEFORE you feel the heat!

<http://www.kozio.com/techpapers> or call us at 303-776-1356

Thomas Struzik



Member Name: Thomas Struzik

Location: Houston, TX

Education: BSEE, Purdue University

Occupation: Software architect

Member Status: He has been a subscriber since day one. "I've got Issue 1 sitting in a box somewhere," he said. Thomas adds that he was a *BYTE* magazine subscriber before *Circuit Cellar*.

Technical Interests: Thomas enjoys automation through embedded technology, robotics, low-level programming, and electronic music generation/enhancement.

Most Recent Embedded Tech-Related Purchase: He recently bought a C WAV USBee SX Digital Test Pod and an Atmel AVR Dragon.

Current and Recent Projects: Thomas is working on designing an isolated USB power supply for his car.

Thoughts on the Future of Embedded Technology: Ever-increasing complexity is becoming a stumbling block for the "average" user. "Few people even realize the technology embedded in everyday items," he said. "How many people know that brand-new LCD TV they've got is actually running Linux under the covers? Fortunately, there seems to be a resurgence of 'need to know how stuff works' with the whole DIY/maker culture. But even that is still a small island compared to the population in general."



CONFERENCE & EXHIBITION
February 19-21, 2013
San Diego Convention Center

www.IPCAPEXEXPO.org

INFORMATION that INSPIRES INNOVATION

IPC APEX EXPO is "THE SHOW" ... to attend valuable courses and see new equipment that will help you and your company see new possibilities, improve your efficiencies and cut costs.

Mario Dion
Manufacturing Engineer Specialist
Trilliant Networks

Join **thousands of your colleagues** from more than **50 countries** and more than **400 exhibitors** at **IPC APEX EXPO 2013**. Learn about **new technologies and processes**, see **new products**, work on **industry standards**, network with **industry experts** and participate in the **world's premier technical conference** for electronics manufacturing. **Register today!**

**design | printed boards | electronics assembly | test
and printed electronics**



Scan for your chance to win
a **three-night hotel stay** or an
MVP registration.

CONTRIBUTED BY
DAVID TWEED

Problem 1—Given a microprocessor that has hardware support for just one level of priority for interrupts, is it possible to implement multiple priorities in software? If so, what are the prerequisites that are required?

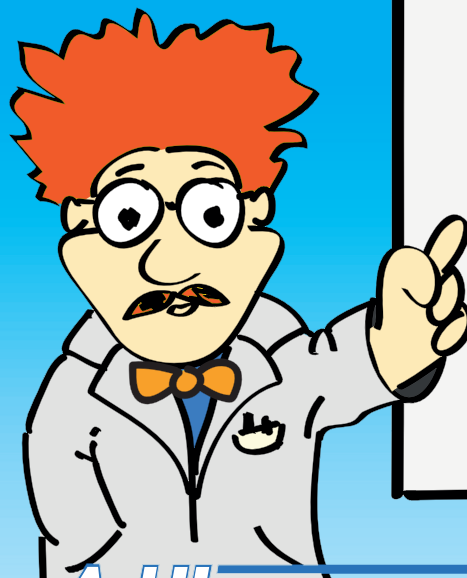
Problem 2—What is the basic scheme for implementing software interrupt priorities?

Problem 3—What are some of the problems associated with software interrupt priorities?

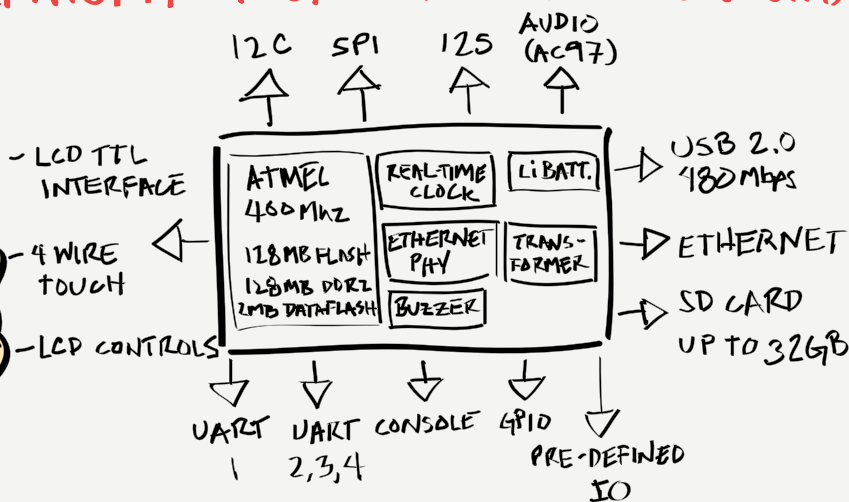
Problem 4—Some processors don't have explicit control over the global interrupt mask; re-enabling it is implicit in the execution of a "return from interrupt" instruction. Is there a workaround?

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

Professor Artila's Tips on System on Module



ANATOMY OF SYSTEM ON MODULE (SOM)



Artila
www.artila.com

www.aaxeon.com // sales@aaxeon.com // +1 (714) 671-9996

aaxeon
Authorized Distributor

Imagineering, Inc.

Client: Imagineering, Inc. (www.PCBnet.com)

Location: 2425 Touhy Avenue
Elk Grove Village, IL 60007

Contact: Khurram Dhanji, KDhanji@PCBnet.com

Embedded Products/Services: Multilayer rigid and flex printed circuit boards, high-mix mixed-volume printed circuit board assembly

Product Information: Join the quoting revolution happening now at PCBnet.com. No longer do you have to wait days for full turnkey quotes. Imagineering's online systems take just minutes to give you quotes for PCBs, components, and assembly all in one place and all online!

Exclusive Offer: Place a full turnkey order with Imagineering for PCBs, components, and assembly labor and receive 25% off your bare boards and 10% off your labor! Imagineering also has daily specials online as well as a new customer promotion. PCBs for \$10 per piece! See all the company's offers at PCBnet.com.

Place a Full Turnkey Order
(PCB's, Components & Labor)

Receive
25% on bare boards
10% on labor

For Daily Specials & Promotions, visit us
@
PCBnet.com



Circuit Cellar prides itself on presenting readers with information about innovative companies, organizations, products, and services relating to embedded technologies. This space is where Circuit Cellar enables clients to present readers useful information, special deals, and more.

We geeked out, so you don't have to.

The ezLCD-304

a 4.3" Smart, Touch LCD



\$159
QUANTITY ONE
PRICE

- Super-easy to program
- Prepackaged with fonts, images, macros and widgets
- Extensive documentation library
- Works with any micro-controller
- Smart, powerful & AFFORDABLE!
- Optional panel-mount bezel

www.EarthLCD.com/ezLCD-304

949.248.2333 • 3184-J Airway Drive • Costa Mesa, CA 92626

©2012 EarthLCD® is wholly owned by Earth Computer Technologies, Inc.

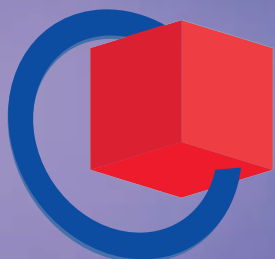
**Earth
LCD**
We Make LCDs Work!™



Nürnberg, Germany

we are the future

February 26 – 28, 2013



embedded world 2013

Exhibition & Conference

... it's a smarter world

Register now for your free entrance ticket:

embedded-world.de

Programmed for success.

With some 900 exhibitors, embedded world is the top pioneering exhibition for embedded technologies worldwide. Make a note of the date now!

Exhibition organizer

NürnbergMesse GmbH
Tel +49 (0) 9 11.86 06-49 12
visitorservice@nuernbergmesse.de

Conference organizer

WEKA FACHMEDIEN GmbH
Tel +49 (0) 89.255 56-13 49
info@embedded-world.eu

Media partners

MEDIZIN & elektronik
Fachmedium für Medizintechnik und Medizintechnik

elektroniknet.de

computer-automation.de

energie-und-technik.de

MEDIZIN-UND-elektronik.DE

DESIGN & ELEKTRONIK
KNOW-HOW FÜR ENTWICKLER

Markt & Technik
Die unabhängige Wochenzeitung für Elektronik

Elektronik
Fachmedium für Industrielle Anwender und Entwickler

Elektronik automotive
Fachmedium für embedded Automotive-Technik

ENERGIE & TECHNIK
Zeitschrift für Energieeffizienz

Computer & AUTOMATION
Fachmedium für Automatisierungstechnik

NÜRNBERG MESSE

MCU-Based Altitude Control

An IR Protocol for Helicopter Commands

This article describes a microcontroller-based altitude controller for a model helicopter. The system uses an IR protocol that sends commands to the helicopter, a phototransistor network that measures altitude, a boom setup that constrains helicopter movement, and a spiking neural network that monitors sensory inputs and output throttle values.

To wirelessly control a model helicopter's altitude and hovering capabilities, we designed a device that uses infrared communication and a simple genetic algorithm implementation. This project culminated from "ECE 4760: Designing with Microcontrollers," which is a one-semester design course taught by Cornell University professor Bruce Land. The project combined our interests in control systems and machine learning and demonstrated our acquired knowledge of microcontroller, hardware, and software design.

Using wireless control, we wanted the helicopter to quickly rise to a given altitude and steadily hover at that position. Another goal was to have the helicopter perfect this task in as few trials as possible. We initially intended to fly the helicopter without any constrictions, but we found it difficult to develop control for all degrees of freedom within a short amount of time and with limited resources. Additionally, the helicopter became unbalanced after we attached an

IR LED for distance measurement. Therefore, we attached the helicopter to a boom and restricted flight to one degree of freedom (DOF). The IR LED attached to the helicopter constantly emitted a signal and was sensed by a phototransistor network placed on the ground under the boom. We used a

neural network to calculate the helicopter's throttle based on distance measured by the phototransistors. We used a genetic algorithm to evolve the network until we achieved a quick rise and steady hover.

SYSTEM DESIGN

The learning process was implemented as a series of 10-s "runs." During each run, the helicopter started on the ground, rose at a speed determined by the neural network and sent by an IR LED, safely dropped, and landed on the ground. The particular neural network used for the run was evaluated at the end of a run. If the run was considered good, the network parameters were stored in the system for further evolution and a different network was uploaded for the

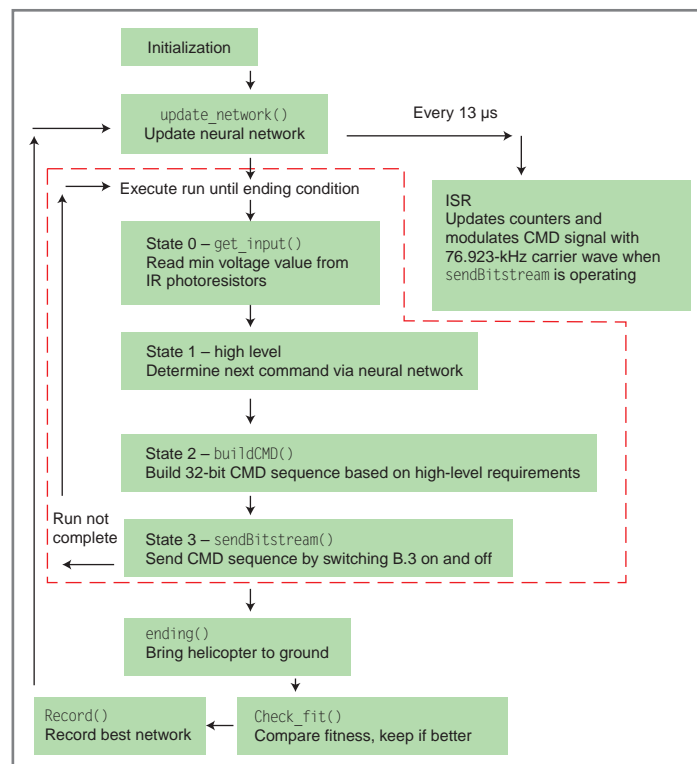


Figure 1—This is the program's high-level functionality. The red box outlines a single "run." The ISR occurs every 13 μ s, even in the middle of a run.

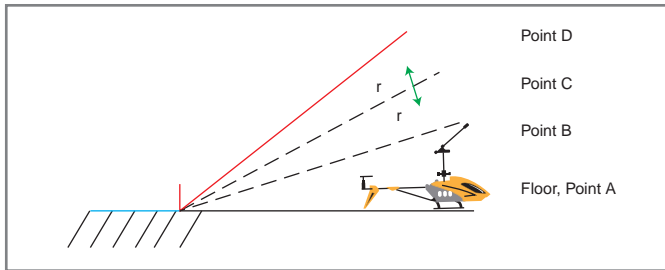


Figure 2—The experimental setup is shown as a series of points. Starting at Point A, the experiment's goal is to fly the helicopter to Point C in the shortest time and to achieve a steady hover around Point C of distance Δr .

next run. Runs were executed until we were satisfied with the helicopter's speed and hover. Figure 1 shows the run structure breakdown.

The microcontroller learned how to fly the helicopter by experimenting with different neural network configurations in an evolutionary manner. Figure 2 shows the experiment's setup. Because of its balance of lightness and rigidity, we used balsa wood to attach the helicopter to a wooden boom. Ground position is referred to as Point A. The goal was to fly the helicopter to Point C in the shortest amount of time and to have the helicopter steadily hover. Point B is a midpoint and Point D is the maximum height the helicopter can achieve before the microcontroller shut down the current run. The maximum point is important because the helicopter loses stability past this point and shoots backward.

A run began with the helicopter at Point A. Every 120 μs , the helicopter's height was calculated through phototransistor readings and the microcontroller emitted a new throttle value. Throttle values were calculated based on neural-network spiking. This will be further described in the software section. The particular network was assigned a fitness value based on how well it accomplished the goal. The network's fitness value started at 0 for a particular run and accumulated every 120 μs depending on the helicopter's height. The helicopter accumulated higher fitness values as it got closer to Point C (i.e., the target point). The concept was to reward the network for quickly flying the helicopter to Point C and for keeping the helicopter closely hovering around Point C. Since every run is capped at 10 s, highest fitness values went to networks that quickly and steadily flew the helicopter to Point C. The exact fitness function used for our experiments is:

$$\text{Fitness} = \begin{cases} +0 \text{ Point A to Point B} \\ +1 \text{ Point B to Point } \frac{3B}{2} \\ +2 \frac{3B}{2} \text{ to C} - 2\Delta r \\ +10 C \pm 2\Delta r \\ +20 C \pm \Delta r \\ -5 > C + 2\Delta r \\ 0 + \text{end Point D} \end{cases}$$

Since the microcontroller couldn't directly read the helicopter's height, we used a look-up table to calculate the height based on phototransistor readings. The IR LED attached to the

helicopter constantly emitted a signal that was picked up by a phototransistor network located directly underneath the boom. When the LED was directly on top of the phototransistor, the voltage was close to 0 V. As the helicopter travelled farther away from the ground, the voltage increased up to a maximum of V_{CC} or 5 V for our circuit. By manually bringing the helicopter to points between Point A and Point D, voltages could be associated with heights.

One neural network was evaluated per run. After 10 s, the microcontroller initiated an ending sequence that brought the helicopter down from any position by gradually decreasing the throttle. Once the helicopter reached Point A, the microcontroller compared the current network's fitness to a six-network population. If the current network has a higher fitness than any network in the population, it replaced the weakest network. For the next run, a network was randomly chosen and randomly mutated in several areas that will be described in further detail in the software section.

HARDWARE

We used a Syma S107R5 helicopter attached to a boom made of balsa wood (see Photo 1). The boom was hinged to a small piece of wood. A thin needle was inserted through the hinge and the balsa wood to keep the boom in place. The wood was manually clamped down, but a mechanical clamp could be used to easily clamp it down. This system's purpose was to reduce the DOF to 1 degree. Due to time and financial constraints, we could not control the helicopter in 6 DOF. Using a boom enabled us to focus on altitude control and develop the learning algorithm.

We attached an IR LED to the helicopter to attain height information. The helicopter came equipped with a blue/red LED attached near the nose. We disassembled the frame and disconnected the LED. Then we connected a Lite-On Technology LTE-4208 IR LED in series with a current-limiting resistor. The on-board 4.2-V battery would provide too much current if directly connected across the LED. We connected a 100- Ω resistor. This would ideally result in 42 mA through the IR LED, which would provide great range. However, the helicopter would not register commands with this configuration because the LED and the on-board phototransistor (which reads remote-control commands) overloaded the battery. We increased the resistor to 330 Ω , which resulted in 12.7 mA through the IR LED. This reduced the range but corrected the



Photo 1—The helicopter is attached to a boom and placed on top of a whiteboard containing a phototransistor network. The command module is a small breadboard behind the wings. The microcontroller is offscreen.

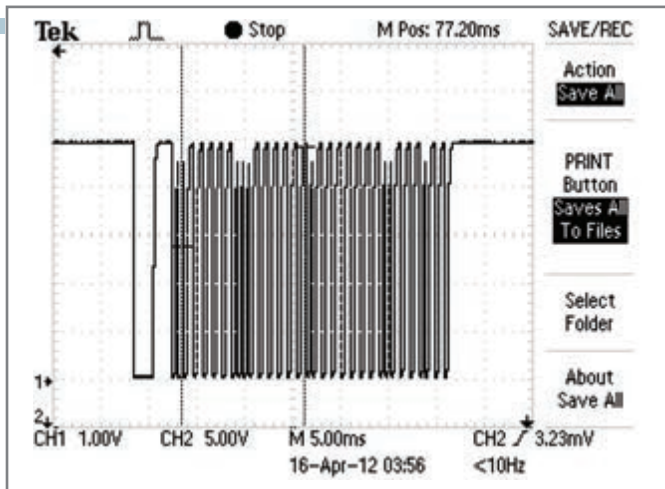


Photo 2—A single IR data packet contains two long header bits and 32 bits representing flight parameters. The shorter peaks are Off bits and the higher peaks are On bits.

helicopter operation. The IR LED's maximum readable distance, as read by a Lite-On Technology LTR-4206E phototransistor, was 25 cm. The minimum distance was 2 cm. We soldered the LED and resistor to the on-board battery's positive and negative leads.

To control the helicopter flight through the microcontroller, we reverse engineered the command protocol from the factory-supplied remote control with help from the *Couch Sprout* blog. We set up a single LTR-4206E to read the command from the remote control. We found commands were organized in packets spaced 120 ms apart. Each packet contained a header, 4 bytes of information, and a stop bit. **Photo 2** shows a single packet on an oscilloscope screen. Moving forward, “low” refers to a low voltage, and “high” refers to a high voltage. The header consists of 2-ms low and 2-ms high. Next, each bit can take one of two values. A 1 is represented by 300- μ s low and 700- μ s high. A 0 is represented by 300- μ s low and 300- μ s high. A byte is composed of 8 bits. By changing one command at a time from the remote, we correlated specific commands to bytes. In order from first to last, the bytes represent yaw, pitch, throttle, and yaw correction. Full yaw to the right is represented by 0, while 127 represents full yaw to the left. Full pitch up is represented by 0, while 127 represents full pitch down. Throttle takes values from 0 to 127. The yaw correction takes the same values as yaw and applies extra yaw. At the end of 4 bytes, a 300- μ s stop bit is issued.

Our test phototransistor could not detect a carrier wave that was issued before every bit. Every bit's low portion oscillates from 0 to 5 V at 76.973 kHz. If this carrier is not inserted, the helicopter's phototransistor and circuitry do not register the command and do not operate.

Our IR command includes three LTR-4208 IR LEDs in series with a 100- Ω , current-limiting resistor. We used three LEDs to increase signal intensity, angle, and range (see **Figure 3**). To send information at the same rate as the helicopter remote, we used an Atmel ATmega644 microcontroller's fast PWM feature, which enabled us to

vary the voltage applied to the circuit from 0 to V_{CC} resulting in fast IR pulses. We used timer-driven interrupts to send IR pulses in packets the helicopter could register. The on-board phototransistor and electronics enable the helicopter to decode the packets and translate them into flight parameters.

Our phototransistor network consisted of eight LTR-4206Es arranged on a whiteboard in a 2×4 rectangular fashion. Each phototransistor only had a 20° viewing angle. We were unable to attain better phototransistors due to time and financial constraints. To ensure the phototransistors could detect the IR from the helicopter at all times, the network was modified based on adjusted heights. Adjustment included phototransistor angling and network size. Each phototransistor was connected to a single ADC pin on the microcontroller that can measure a voltage from 0 to 5 V in 0.039-V increments. In addition, a 20-k Ω current-limiting resistor was placed in series with each phototransistor (see **Figure 3**).

EVOLUTIONARY ALGORITHM

According to Y. H. Said in her article, “On Genetic Algorithms and Their Applications,” an evolutionary algorithm (EA) is a global optimization search that operates on the principles of natural selection.^[1] The algorithm's goal is to find a solution that maximizes a problem's given fitness function. The EA starts by instantiating several solutions or chromosomes. Each of these is evaluated by a fitness function based on how well it solves a particular problem. Chromosomes are randomly mutated and retested. If they perform better than any chromosome in the current population, they replace the worst chromosome. After a round of mutations and checks, the best chromosomes reproduce and move on to the next generation. The EA performs several iterations until the search converges onto a solution and fitness values barely increase across generations.

We used an EA for this project to develop the best helicopter flight. We used an implementation first presented by D. Floreano, et al in their article, “Evolutionary Bits'n'Spikes.”^[2] This

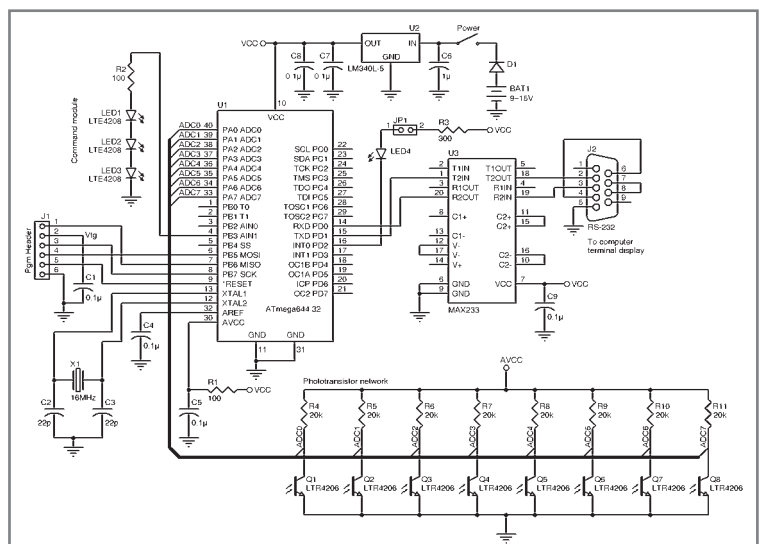


Figure 3—The IR command includes three Lite-On Technology LTR-4208 IR LEDs in series with a 100- Ω , current-limiting resistor. We used three LEDs to increase signal intensity, angle, and range. We placed a 20-k Ω , current-limiting resistor in series with each phototransistor.

We've got you covered!



8 | 16 | 32 bit
PIC® MICROCONTROLLERS



Feeling locked in?

Forced architecture and other compromises should not constrain your design. With Microchip's 8-, 16- and 32-bit solutions, tools and compilers, you won't have to.

The choice is easy when you think about it.

Visit Microchip.com/pic today.

DIVERSITY WITHOUT COMPROMISE

MICROCHIP.COM/PIC



algorithm uses a spiking neural network to take altitude information as sensory input and output throttle as motor output. Neurons can be simply described as single compartments with associated membrane voltages. Certain neurons are responsible for sensing input and relaying this information through a network of connections to terminal neurons that can determine motor output. The relay process occurs through “spikes” or extremely fast voltage swings, which occur when a neuron’s membrane potential passes a threshold. The terminal neuron’s membrane potentials are affected by presynaptic neurons (i.e., those that connect to the terminal neuron). The potential can increase if the presynaptic neuron is excitatory or decrease if the presynaptic neuron is inhibitory. The connections between presynaptic and terminal neurons can be configured to enable the motor output to react to sensory input as desired. This is the basic principle behind learning motor activity in biological systems.

Our neural network includes three major components. First, a layer of sensory neurons spikes is based on altitude information. Second, these spikes are relayed through a network of connections. Third, the connections synapse onto a motor neurons layer. Some of the motor neurons also have connections to other motor neurons. Motor neurons spike based on the spiking through the network of connections. Output spikes define the helicopter’s throttle output. **Figure 4** shows an example of this system.

DIGITAL IMPLEMENTATION

You can use an 8-bit microcontroller to efficiently code the spiking neural network. A single neuron can be represented by a membrane potential, threshold, sign, spike output, and connections. A few attributes describe a neuron’s behavior over time or cycles. A neuron is either spiking (represented by a 1) or not spiking (represented by a 0). Once it has spiked, a neuron cannot spike for one cycle. A neuron can synapse onto other neurons. A neuron’s membrane potential can increase or decrease depending on spiking contributions from input neurons. The change

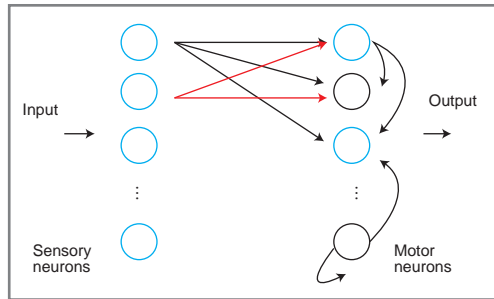


Figure 4—Here is a simple spiking neural network. Input is provided to the sensory neurons, which have connections to motor neurons. Motor neurons also have various connections and signs as shown by blue (excitatory) and black (inhibitory). Output is extracted from motor neuron spikes.

direction is determined by the input neurons’ sign. For example, if a neuron has three excitatory inputs and three inhibitory inputs and all inputs spike, then that neuron’s membrane potential will increase by 1. A neuron can spike if its membrane potential rises above a threshold, which we set at $5 \pm$ a random number between -2 and 2 . The random number ensures the system does not fall into locked oscillations. Finally, a leakage of 1 is always subtracted from a neuron on every cycle. This models the leakage biological phenomenon.

Our project utilizes eight sensory and eight motor neurons. The ADC converts all voltages from the phototransistor to digital values and finds the minimum. It is expected that this voltage most accurately reflects the helicopter height. The others might not be in the helicopter’s field of view. The minimum voltage is represented by an 8-bit number and each sensory neuron corresponds to one of those bits. Connections between a single-sensory neuron and all motor neurons are represented by a single byte. Each bit is a 1 for a connection or 0 for no connection. Therefore, connections between all sensory neurons and all motor neurons can be represented by 8 bytes. Motor neurons can also connect to each other. Another 8 bytes are used for these connections. Finally, one motor neuron output is used to increase throttle by a set amount and another neuron is used to decrease throttle. The other motor neurons are used for intermediate connections.

When the helicopter is close to the ground, all sensory neurons spike, which results in many motor output spikes. For the proper configuration, this will

increase throttle. When the helicopter reaches peak height, fewer sensory neurons spike, which results in a throttle decrease. The proper network configuration will result in the best flight. However, this configuration depends on the experimental setup and noise factors. The entire neural network can be described in 17 bytes: 1 byte for motor neuron signs (sensory neurons are kept excitatory), 8 bytes for sensory connections, and another 8 bytes for motor connections.

The EA produces a random, initial population of six networks or chromosomes. These networks are tested, mutated, and reproduced through several algorithm iterations. Mutation is done in a random manner. A single bit of the sign byte and both connection bytes is toggled at a time. A single run (as described in the System Design section) tests a single network and produces a fitness value that evaluates that network against the current population. If the network is better than any population network, the superior network replaces the inferior one. Mutations are performed until the user is satisfied with the altitude control.

RANDOM NUMBERS

We used a 32-bit linear feedback shift register as a random-number generator. A 32-bit register was instantiated and bits 27 and 30 were XORed to produce bit 0. Then the register was shifted left, which produced a random number with a repeat time that lasted much longer than any experiment. This configuration produces high-quality uncorrelated random numbers.

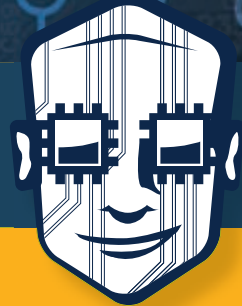
IR PROTOCOL

Time-driven interrupts from the ATmega644 are used to toggle PB3 to the IR command module (as described in the Hardware section). For this project, yaw and pitch are kept constant. Throttle is taken from the spiking neural network’s output. Every 120 μ s, a 32-bit command is built. This command represents the yaw, pitch, throttle, and yaw correction, in that order. Interrupts occur at 100- μ s periods then toggle V_{cc} according to the protocol presented in the Hardware section. Another 76.973-kHz interrupt is used to generate the carrier wave.

DESIGNCON[®] 2013

CONFERENCE January 28-31 | EXPO January 29 & 30
Santa Clara Convention Center | Santa Clara, Ca

>> WHERE CHIPHEADS CONNECT.



Created by engineers for engineers, DesignCon is your one-stop shop to upgrade your knowledge and skills with the latest theoretical design techniques and methodologies while seeing first-hand demonstrations of today's most advanced design tools and technologies.

Join your peers at the largest meeting of chip and board designers in the country.

Covering critical issues around:

- PCB design tools
- RF and Signal Integrity
- FPGA design and Debug
- High-Speed Serial Design
- Verification Tools
- Interconnect Technologies
- Semiconductor Components
- ICs and more

With 100+ tutorials & technical paper sessions | 130+ exhibitors showcasing a wide variety of design tools | DesignTOUR giveaway | Fun networking events
Panel discussions, speed-trainings & product teardowns
Agilent Education Forum | Happy Hours | DesignVision and Best in Test Awards

>> Register today at www.designcon.com!

Early Bird rates end Dec. 7 | Advance rates end Jan. 18

KEYNOTE SPEAKERS



Bill Swift
Vice President of Engineering,
Cisco System, Inc.



Jonah Alben
Senior Vice President GPU
Engineering, NVIDIA



Mike Santorini
Business & Technology Fellow,
National Instruments

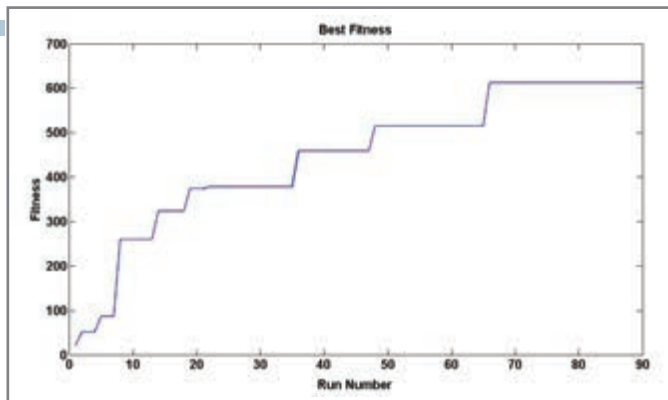


Figure 5—This graph shows the population's best fitness across each run. Individuals quickly evolved in the experiment's beginning and slowed down near the end.

TAKING FLIGHT

The system quickly evolved high-fitness chromosomes in the first 20 runs (see Figure 5). It took more runs to discover better chromosomes as the system became adept at flying the helicopter according to the goal. Many chromosomes resulted in the helicopter flying too high. These were avoided later in the experiment as they resulted in 0 fitness. Chromosomes that didn't raise the helicopter above ground resulted in 0 fitness and were also avoided. The average population fitness sampled every five runs (see Figure 6). A high slope was exhibited in the beginning and it tapered off near the end of the experiment.

The phototransistor network was susceptible to slight changes in movement because the phototransistors only had a 20° viewing angle. The helicopter frequently rotated or shifted a small amount at higher altitudes. This caused the IR LED to leave the phototransistor's field of view. The system interpreted this as a bad run, which resulted in a false negative for a chromosome that may have had a high fitness. The system also generated several false positives for chromosomes that rose to low altitudes but evaded the phototransistors' viewing angle just enough to trick the sensors into thinking the helicopter was at the right altitude. These chromosomes generated high fitness values for bad runs and severely hampered the network's evolutionary capability. However, enough mutations and runs ensured that false negatives and positives did not impede the system's end goal.

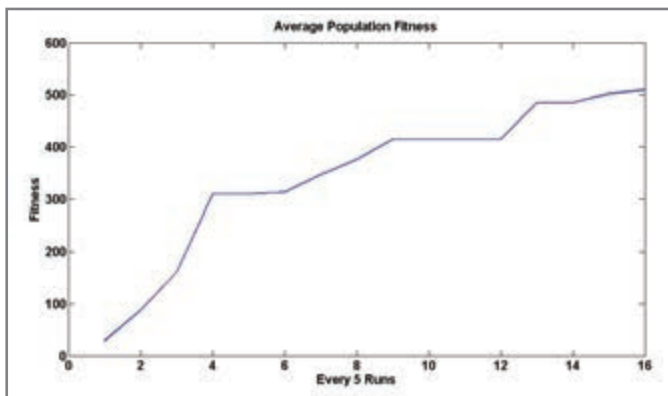


Figure 6—The average population fitness exhibits a high slope near the beginning and a low slope near the end. The average was taken every five runs.

LIFTING OFF

A robust learning and evolving system was developed to implement altitude control from a microcontroller to a toy helicopter. This system involved an IR protocol execution to send commands to the helicopter, a phototransistor network to measure altitude, a boom setup to constrain helicopter movement to 1 DOF, a spiking neural network to take sensory inputs and output throttle values, and an evolutionary search to find the best neural-network configuration. The search converged into a solution that quickly flew the helicopter from the ground to a desired point and kept it at a steady 10-s hover. Although the system exhibited several false positives and negatives throughout the search process, the microcontroller overcame them by testing enough chromosomes to find the proper configuration. Future work should focus on improving experiment aspects, such as using wider angle phototransistors and a more reliable boom. 📌

Authors' acknowledgements: We would like to thank Professor Bruce Land and the Cornell University ECE Department for guidance, ideas, and resources from start to finish of this project.

Akshay Dhawan (and43@cornell.edu) is an MEng student at Cornell University in Electrical and Computer Engineering. He earned his Bachelor of Science at Cornell University. He is specializing in bioinstrumentation. His interests include neurophysiology, neural interfaces, machine learning, and evolutionary algorithms. After earning his MEng degree, Akshay will pursue an MBA at Cornell University.

Sergio Biagioni (sab323@cornell.edu) is an Application Support Engineer at MathWorks. He earned his BS and MEng at Cornell University in Mechanical and Aerospace Engineering. He specializes in aerospace control systems, dynamic modeling, and simulation.

REFERENCES

- [1] Y. H. Said, "On Genetic Algorithms and Their Applications," *Handbook of Statistics, Volume 24: Data Mining and Data Visualization*, North Holland, 2005.
- [2] D. Floreano, N. Schoeni, G. Caprari, and J. Blynel, "Evolutionary Bits'n'Spikes," *Artificial Life VIII: The 8th International Conference on the Simulation and Synthesis of Living Systems*, 2002.

RESOURCES

Agustin, *Couch Sprout*, "Arduino Helicopter Infrared Controller," 2011, www.avergottini.com/2011/05/arduino-helicopter-infrared-controller.html.

New Wave Instruments, "Linear Feedback Shift Registers," www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm.

Syma Helicopters, www.symahelicopters.com.

SOURCES

ATmega644 Microcontroller
Atmel Corp. | www.atmel.com

LTE-4208 LED Emitter and LTR-4206E phototransistor
Lite-On Technology Corp. | <http://optoelectronics.liteon.com>

CONNECT WITH **Circuit Cellar**

For people who are passionate
about hardware and software design,
embedded development, and
computer applications. Awesome
projects, tutorials, industry news,
design challenges, and more!



Connect.

Follow us on Twitter.
Like us on Facebook.

www.circuitcellar.com



@CircuitCellar
@editor_cc



circuitcellarmagazine

Open-Source Hardware Development

Open-source hardware is more popular than ever. This article describes an Arduino-based data logger built to log data about bats' entry/exit behavior. The logger was designed and built using open-source hardware and software, contributed code, and hardware CAD files.

Every software developer knows about the pre-Facebook social networking phenomena of open-source software. This concept of sharing software and intellectual property among a wide audience of contributors and users led to several high-profile collaborations, such as the Apache Software Foundation, Linux distributions, and SourceForge. What about hardware? Replacing the

handful of older standards (remember PC/104?), a contemporary movement toward open-source hardware—which encourages reusing and extending firmware and board designs—is catching on among small system and device vendors. Currently, the most widely known embedded example is the Arduino controller board, which has its own open-source programming language. How can

all this software and hardware goodness be leveraged in a practical project?

Answering a call for project ideas for our Embedded Computing course, a biologist colleague suggested logging data on bat entry/exit behavior. It sounded like just the thing for an embedded application. Beginning as an exercise in selecting sensors and storage for data logging, we ended up using open-source

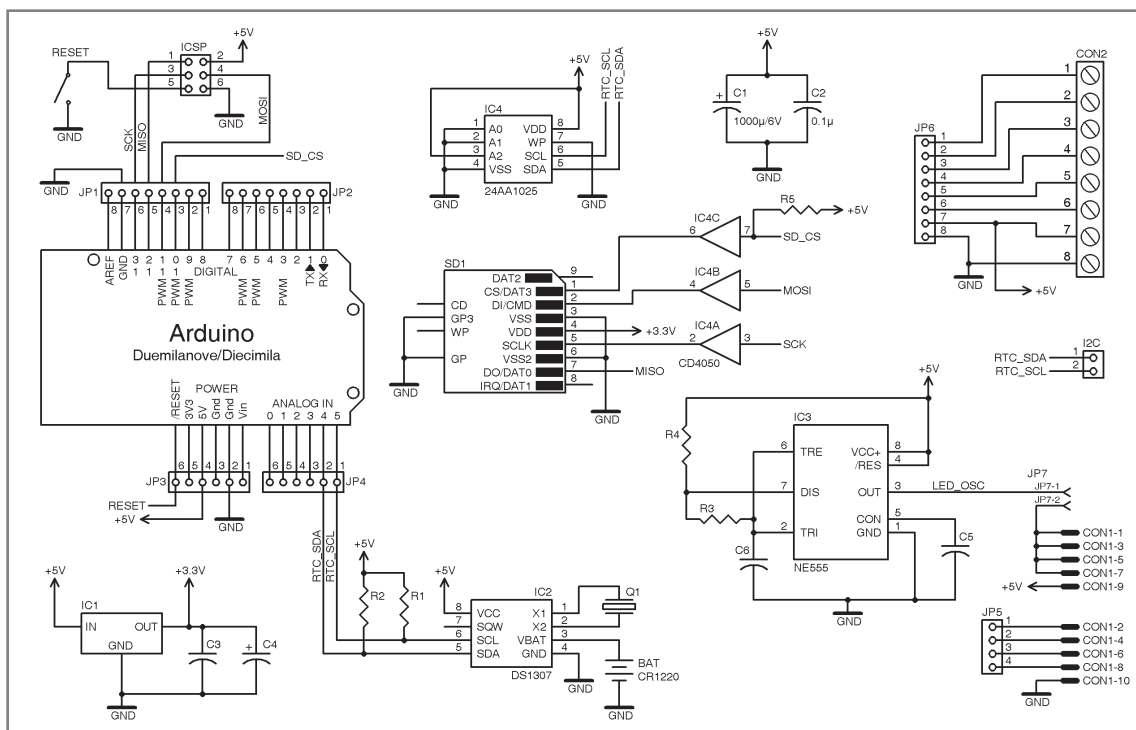


Figure 1—An Arduino microcontroller board is at the heart of the bat house logger shield.

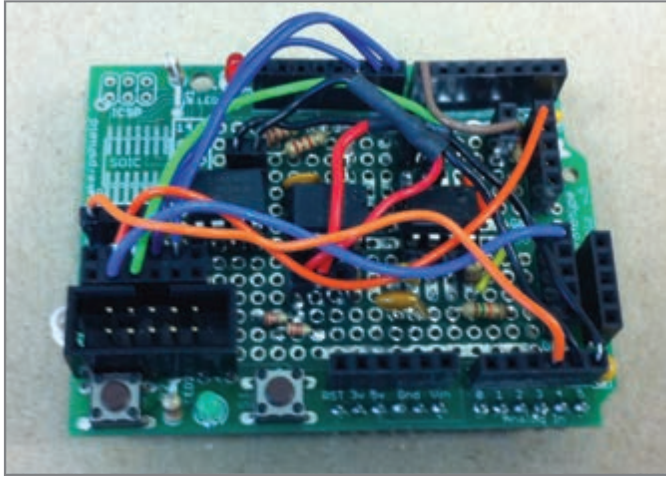


Photo 1—An AdaFruit Proto Shield was used to prototype the bat logger.

hardware and software for all aspects of the project, down to the bat house's physical design. This article describes the hardware and software we used and extended for this project. But first, a few words about bats.

BAT LOGGER SPECIFICATIONS

Our bat expert explained that the bats under consideration are not large and tend to cluster together in small spaces (e.g., bat houses), when not hunting insects at night. The bats happily squeeze into 0.75" roosting slots and are quite good climbers. Our colleague wanted to collect data on the bats' entry/exit activity and check for correlations with interior/exterior temperatures and ambient light levels. As a reward for the bats' cooperation, they received a new sensor-equipped bat house to replace their dilapidated one. With this win-win situation resolved, we proceeded to set our project specifications, keeping in mind other potential logging applications (e.g., bird houses).

Our bat logger specifications included: an Arduino controller board, a real-time clock (RTC) with battery backup, up to four temperature probes, up to four asynchronous entry/exit sensors, an ambient light sensor, a serial EEPROM and/or SD card storage, and a USB serial connection to a simple user interface for configuration and control. We also wanted everything solar powered for unattended/remote operation. In addition, we wanted the nonproprietary hardware and software to be sufficiently accessible and affordable for a high school, college, or DIY-level project, indicating mainly 0.1" pitch, through-hole DIP components.

An out-of-the-box data logger would meet almost all our requirements. But we wanted to explore the complete open-source route, including bootstrapping our own circuit, PCB design, and open-source software. Specific sensor selection was fairly simple, as there are a number of inexpensive devices already supported by Arduino and other open-source libraries. We wanted to use low-cost parts that featured software support and were available from our spare parts boxes. [Figure 1](#) shows the Arduino-based logger schematic.

Choosing Arduino as our controller enabled us to use open-source hardware "shields" available from several vendors. Shields typically fit the slightly eccentric, but now standard,

header-pin spacing for typical Arduino boards and contain specialized circuitry for data loggers, GPS devices, Ethernet chips, motor drivers, audio codecs, and so forth.

Shields are available as bare PCBs, kits with components, or fully assembled and tested. The open-source aspect usually comes from CadSoft's Eagle PCB software and/or image files for shields, often published under a Creative Commons agreement enabling royalty-free use and extensions of the shield design. The licenses' terms differ between vendors, but there is a trend toward such source hardware at smaller DIY and hobbyist shops. Creative Commons's Attribution-ShareAlike license is a typical example that, according to its website, "lets others remix, tweak, and build upon your work even for commercial purposes, as long as they credit you and license their new creations under the identical terms."^[1] This type of license is part of the latest chapter in the extremely long saga of GPL, GNU, and other software licenses dating back to the Unix wars. We won't revisit that story here, but it is good to know your history in this area.

DATA LOGGER SHIELD

We used an AdaFruit Industries prototyping shield kit for our data logger prototype's initial design and construction. This required mainly point-to-point wiring between the IC sockets and discrete components. As you can see in [Photo 1](#), it was not a thing of beauty, but it did the job of testing and debugging the hardware (except for the SD card and circuitry, which took up a bit too much space).

Next, we used an existing AdaFruit data logger shield as the basic design model for our full logger. With downloadable CAD files, we used CadSoft's Eagle light edition to design our shield schematic and PCB with minimal trouble. Numerous device vendors supply helpful device library files for use in the PCB layout process. Many tutorials for using Eagle and other tools are available on YouTube. Since we wanted to continue in the open-source development mode, we had the boards fabricated by BatchPCB, an affiliate of Sparkfun Electronics. Submitted via its automated process, Gerber and drill files describing smaller designs like ours are collaged to form larger boards that are sent to a fabrication house then separated and

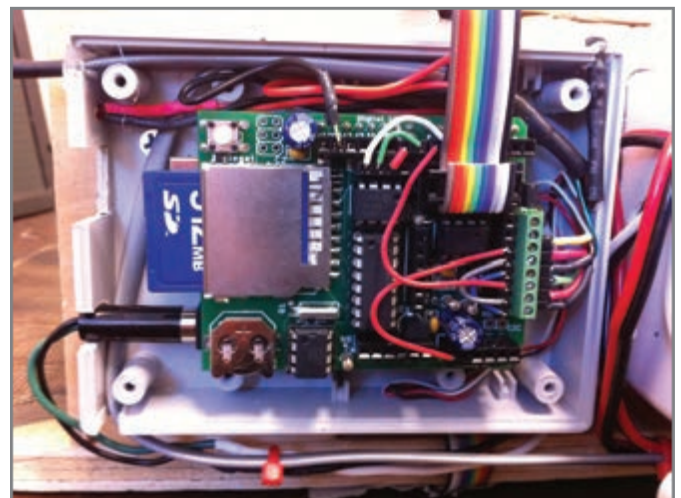


Photo 2—This is the bat house logger shield.

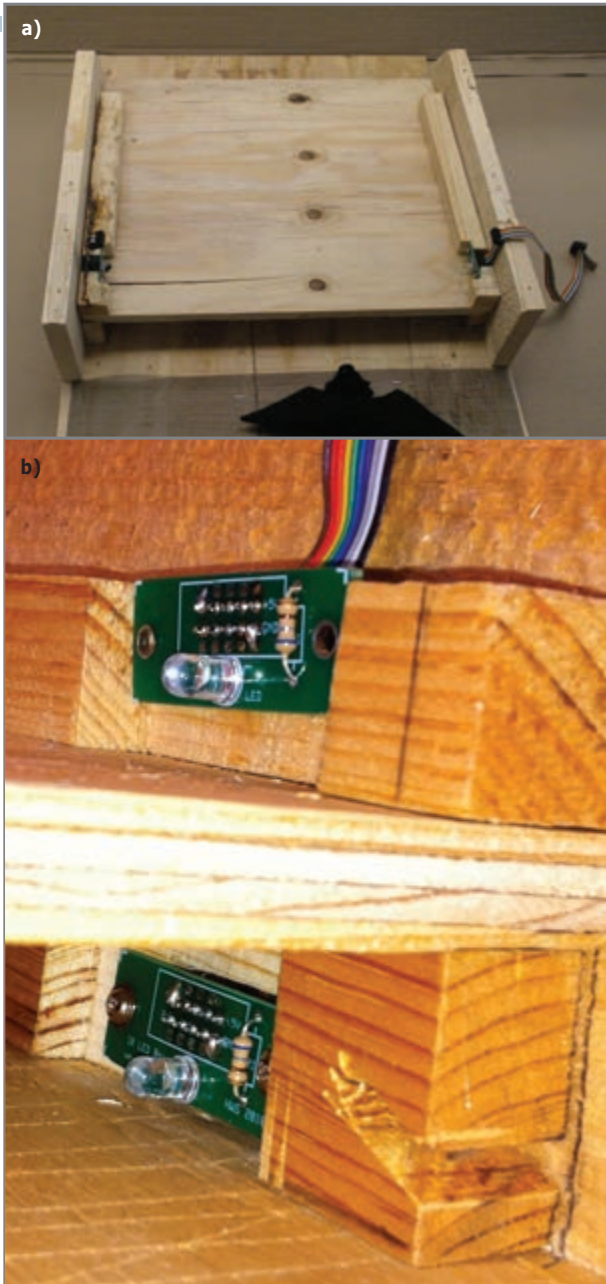


Photo 3a—The break-beam LEDs and sensors are mounted within recesses facing across the bat house's three roosting slots. **b**—Here is a close-up view of IR LED break-beam boards.

shipped to customers to reduce costs. Board designers can also advertise their boards on BatchPCB's website. After resolving some problems with a couple of board design glitches, we had three boards made. Once we stuffed and soldered them, we had working loggers. [Photo 2](#) shows the shield in action. Modifications for use as a bird house logger are straightforward, as is adding additional sensors.

At full operation, the logger consumes about 80 mA from a 12-V supply, which we could have reduced with some better

with three seasons of data logging.

BAT LOGGER CONSTRUCTION

The wooden bat house was constructed by a fellow student and delivered for our sensor fitting. We used surplus four-conductor phone cable to wire Maxim Integrated Products 1-Wire thermometers together. We mounted them through holes inside the top of the house. We kept one outside for ambient readings. We designed some smaller PCBs to hold our infrared (IR) receiver and IR LEDs, which formed our simple break-beam

Listing 1—This is the RTC get date/time function.

```
// Gets the date or time from the ds1307. We put the
// resulting date/time parts into our parameter array
// to pass them back to the calling function.
void getDateTIme(int dateTIme[]){
    byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

    // Reset the register pointer
    Wire.beginTransmission(DS1307_I2C_ADDRESS);
    Wire.send(0);
    Wire.endTransmission();
    Wire.requestFrom(DS1307_I2C_ADDRESS, RTC_PIN);

    // A few of these need masks because certain bits are control bits
    second    = bcdToDec(Wire.receive() & 0x7f);
    minute    = bcdToDec(Wire.receive());
    hour      = bcdToDec(Wire.receive() & 0x3f);
    // Need to change this if 12 hour am/pm
    dayOfWeek = bcdToDec(Wire.receive());
    dayOfMonth = bcdToDec(Wire.receive());
    month     = bcdToDec(Wire.receive());
    year      = bcdToDec(Wire.receive());

    dateTIme[0] = (int) second;
    dateTIme[1] = (int) minute;
    dateTIme[2] = (int) hour;
    dateTIme[3] = (int) dayOfWeek;
    dateTIme[4] = (int) dayOfMonth;
    dateTIme[5] = (int) month;
    dateTIme[6] = (int) year;
} // End of getDateTIme

}
```

sensors. These are connected by inexpensive 10-conductor IDC ribbon cable, which simplifies connections but required some decoupling capacitors to avoid crosstalk. [Photo 3](#) shows how the break-beam LEDs and sensors are mounted within recesses facing across the bat house's three roosting slots.

The software is built on a collection of C++ libraries for the Arduino language. The Arduino IDE/compiler is accessible as Java open-source software and available for Windows, OS X, and Linux. Our RTC's I²C code is part of the Wire. [Listing 1](#) shows intentionally verbose code to extract the RTC response's time and date parts. Arduino provides support for the 1-Wire library as well, enabling simple queries to our three TO-92 form factor Maxim DS18B20 thermometer ICs. Two other useful libraries were the Arduino serial EEPROM I²C library and the Fat-16 library for reading and writing the SD card files. These libraries contain code contributed by a variety of hobbyists, vendors, and anonymous do-gooders. These kinds of contributions power the open-source movement and keep it alive and well.

ONE POWERFUL TOOL FOR YOU

The Entire *Circuit Cellar* Magazine Archive on a
Limited-Edition 25th Anniversary USB drive!



Here's what's included:

- Special anniversary USB drive or our traditional CC GOLD USB drive. *Your choice!*
- PDFs of all *Circuit Cellar* magazine issues in print through date of purchase
- Article code
- Design Challenge projects*
- Two years of *Elektor* magazine issues in PDF format (2010–2011)
- Two years of *audioXpress* magazine issues in PDF format (2010–2011)
- A USB memory upgrade from 16 GB to a whopping 32 GB!
- Free gift! *Circuit Cellar* 25th anniversary hat

Order today at www.cc-webshop.com

* Design Challenge add-ons:

Atmel AVR Design Contest 2006, WIZnet iEthernet Design Contest 2007, Microchip Technology 16-Bit Embedded Control Contest 2007, Texas Instruments DesignStellaris Contest 2010, WIZnet iMCU Design Contest 2010

Listing 2—These are the Arduino setup and our main loop code for handling user I/O and calling the logging methods.

```
//Setup simply gets our Wiring, Serial, inits variables and starts a timer to wait for user input.
void setup() {
  Wire.begin();
  Serial.begin(BAUD);      // Open the serial port:
  initialize();             // Do our variable initializations
  Serial.println("Enter any char for main menu.");
  startTimer(millis());     // Ready to start logging or menu
} // End setup

/* Main loop begins here. On startup we display a prompt and wait a few seconds for user input. If we receive none, go directly into
 * logging mode and log data using the previous EEPROM restored settings. If we get any user input, show the main menu and
 * go into menu mode. We never return to start mode but either are in menu or logging mode from now on. We can switch between these
 * two modes by user input. Note, data is logged in two situations. Regular (synchronous) data is logged on a periodic interval
 * set by the user. Asynchronous data logging happens when IR break beam events happen.
 */

void loop() {
  // Capture any user input and react accordingly
  while (Serial.available() > 0) { // Add any incoming Serial characters to our buffer
    myChar = filter(Serial.read()); // Filter our chars to ignore spaces, convert to lower case, recognize CR if needed
    if (myChar > CR) strBuffer[currIndex++] = myChar; // Good input, add to buffer
    currIndex %= STRING_MAX; // Wrap around for large input, no overruns
    strBuffer[currIndex] = NULL_CHAR; // Manually add string terminator as temporary end of input
    delay(10); // Give some time so we get an entire user input string if no CR is coming
  } // End while still have input to process
  // If we need to wait for a CR to terminate our input, do so by looping again
  #if WAIT_FOR_CR
    if (myChar != CR) return;
  #endif

  // Check which mode we are in and react accordingly
  switch (mode) {
    case START: // Start mode normally occurs only once after reset. Just wait for any user input until timeout.
      if ( (strlen(strBuffer) > 0) && (elapsedTime(SECONDS) < COUNTDOWN) ) { // Got input?
        mode = MENU; // Get ready for menu mode.
        showMainMenu();
        currIndex = flushBuffer(strBuffer);
      } // End if
      else if ( elapsedTime(SECONDS) > COUNTDOWN ) { // No user input, we time out and go directly to logging
        mode = LOGGING;
      } // End if
      break; // End of start mode

    case MENU: // Handles input meant to indicate menu choices with parameters attached
      if ( elapsedTime(TENTHSECONDS)%10 < 5 ) LED_ON;
      else LED_OFF; // Blink to show we are in menu mode
      if (strlen(strBuffer) > 0) { // Have a non-empty string to process
        int command = getCommand(strBuffer[0]); // Get the initial command character, match latter with our command list
        strBuffer[0] = ' '; // Erase our first char so we can now do an atol on the rest
        for (int i = 0; i < strlen(strBuffer)-1; i++) { // Collect remaining user input into our command parameter string
          strBuffer[i] = strBuffer[i+1];
        } // End for copying chars to strBuffer as parameter
        strBuffer[currIndex-1] = NULL_CHAR; // Set new terminator
        if (doCommand(command, strBuffer)) // Carry out the command indicated, may be ill formed, catch in this function call
          Serial.println("Operation successful."); // Indicate success/failure to user
        else
          Serial.println("Error in operation.");
        currIndex = flushBuffer(strBuffer); // Done with input, clear buffer to avoid atol hangovers
      } // End if we read some serial bytes
      break; // End of menu mode

    case LOGGING: // Start logging data here
      if (strlen(strBuffer) > 0) { // First, check to make sure the user is not typing some chars to interrupt logging
        mode = MENU;
        showMainMenu();
        currIndex = flushBuffer(strBuffer); // If the user enters anything, return to menu mode
        break;
      } // End if
      // Now actual logging events begin
      irIndex = updateIR(); // Any IR events recorded?
      if ( irIndex != NOT_FOUND ) { // Check asynchronous events status and log any such event
        logData(ASYNCH_EVENT);
        clearIR(irIndex); // Logged this event, clear it to start anew
      } //
  }
}
```

Listing continued on p. 35.

Listing 2—Continued from p. 34

```
if ( elapsedTime(SECONDS)%10 == 1 ) LED_ON;
else LED_OFF; // Blink once to show we are logging
if (elapsedTime(TENTHSECONDS) > logInterval) { // Log when our time interval is reached
    logData(SYNCH_EVENT); // Do the logging now
    resetTimer(); // Restart our ms timer so we know when to log again
} // End if
break;
default: // Unknown mode. Should not get here without some weird error.
;
} // End switch
} // End loop
// End of Bathhouse file
```

CODE

As shown in Listing 1, Arduino's simplified C++ syntax enables you to declare Wire and OneWire objects, invoke static methods, and use the dot method/variable notation. You can do so without excessive object code overhead, which is critical, given we are targeting a 32-KB flash RAM microcontroller. We did avoid using the latest Arduino (0019) string classes, despite their convenience, as we needed to keep our build well below 28 KB to enable adequate stack space. The Arduino language enforces use of `setup()` and `loop()` functions for initialization and to endlessly read and process data, respectively. We opted for simply polling our IR detectors since we were sensing fairly slow-crawling bats and even slower temperature and light changes. Listing 2 shows our main loop code for handling user I/O and calling the logging methods.

Finally, the user interface consists of single-character commands with an optional parameter received via the Arduino USB serial port. You can use any serial comm program from PuTTY to the built-in Arduino IDE serial window to configure and control the logger. Photo 4 shows the initial menu choices following a reset, which involves either setting EEPROM values restored on reset to ensure consistent restarts or utility commands (e.g., preview of sensor data) so we could be confident of our data integrity before we began logging. In the example shown, the user just requested initial probing for 1-Wire thermometer addresses, of which you see three reporting "present." A 500-MB SD card holds many days of typical bat data and/or we can log to the smaller-capacity serial EEPROM with user-selected logging

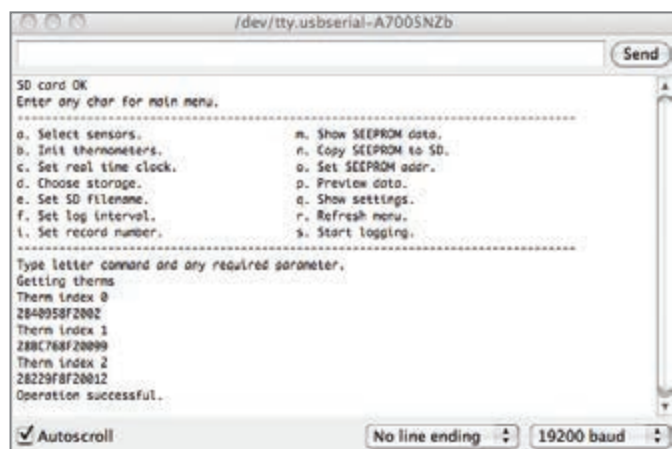


Photo 4—The user configuration/control menu shows that the user just requested initial probing for 1-Wire thermometer addresses.

intervals from 0.1 s to several days. Photo 5 shows the completed bat house with Arduino/shield enclosure, batteries, and a solar charge controller mounted on the side.

In late fall, after collecting some data in the field for seven consecutive days, it was easy to copy CSV files from the SD card to a PC and use a spreadsheet program to plot columns. You can also stream the data out the Arduino USB cable and collect it with any serial comm program. Figure 2 shows a plot of some field data. The horizontal scale is hour:minute and spans a single 4:00 AM to 10:00 PM interval. The specific interval was chosen to illustrate the temperature and ambient light sensors, which are essentially flat during the evening hours when the bats are active. The figure shows superimposed plots (on an arbitrary 100% vertical scale) of seven data series. These include the ambient light sensor, three temperature readings (two on the inside of the roosting area's inside and one on the outside), and three IR break-beam sensors for detecting crawlspace activity.

The inexpensive CdS ambient light sensor is sensitive enough for us to discern the effects of clouds passing overhead



Photo 5—The completed bat house includes an Arduino/shield enclosure, batteries, and a solar charge controller mounted on the side.

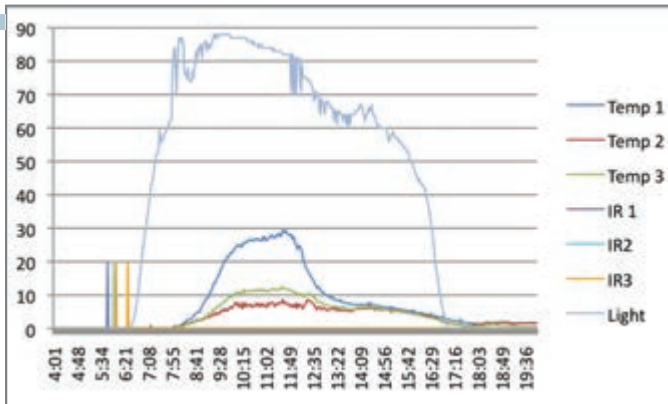


Figure 2—This plot spans a single 4:00 AM to 10:00 PM interval that shows the temperature and ambient light sensors, which are essentially flat when the bats are active.

and shadows cast by nearby trees. Considering our bat house placement, we could see the slanting sunlight's effect on the bat house temperatures. The sunnier sides were clearly hotter. Finally, the IR break-beam sensors are simply binary data spikes occurring at dawn. With the current IR break-beam design, we cannot distinguish between exit and entrance or even two or more bats simultaneously entering or leaving in an echelon along the crawlspaces. For the next design, we may consider something more informative than binary information, but this one does the job. The actual data collected enables fine- or coarse-grained data display depending on the sampling interval, which is set through the software interface.

OPEN-SOURCE BOOTSTRAPPING

The data collected thus far shows the various sensors and software are sufficiently capable for researchers to perform detailed studies on bat activity in relation to environmental factors. However, there is a serious concern about the bat population level to monitor. A devastating outbreak of so-called "white-nose syndrome" is causing the bat numbers in our area to plummet, with an estimated 90% decrease of the little brown bat population. Adding noninvasive video/still cameras, saving images to SD cards to detect infected bats in the houses may be a useful addition to help understand this disease.

This project has led to a natural modification aimed at data collection for small bird houses. These structures require smaller versions of Arduino, many of which are available, and different choices of sensors and data storage. Open-source Arduino libraries are available for most sensors, so leveraging the original code base has proven fairly straightforward with only minor modifications.

Our take-home message is that using open-source hardware and software provides valuable support to rapidly prototype and build your own designs using contributed code and hardware CAD files. In particular, we are grateful to have built from AdaFruit's well-documented logger board and the fatlib SD card library. While this type of open-source bootstrapping may not help with more stringent low-power designs or specialized PCB needs, it is certainly worth considering for many other projects. 📌

Josh Davis (josh.davis15@gmail.com) is a graduate of Hobart College, holds an MEng from Cornell University, and works at Strategic Solutions.

Tomas Carvalho e Silva (tomas.carvalhoesilva@gmail.com) is a graduate of Hobart College from Santarém, Portugal currently living in Washington, DC. He is an analyst at Freddie Mac.

John Vaughn (vaughn@hws.edu) is an Associate Professor at Hobart and William Smith Colleges where he teaches mathematics and computer science.

REFERENCE

[1] Creative Commons, "About the Licenses," <http://creativecommons.org/licenses>.

RESOURCES

BatchPCB, <http://batchpcb.com>.

Sparkfun Electronics, <http://sparkfun.com>.

SOURCES

Adafruit Proto Shield for Arduino Kit

Adafruit Industries | <http://adafruit.com>

Arduino Duemilanove microcontroller board

Arduino | <http://arduino.cc/en>

Eagle PCB software

CadSoft, Inc. | www.cadsoftusa.com

DS1307 Real-time clock and DS18B20 1-Wire digital thermometer

Maxim Integrated Products | www.maximintegrated.com

24AA1025 CMOS Serial EEPROM

Microchip Technology, Inc. | www.microchip.com

TSOP3438 IR Receiver module

Vishay Intertechnology, Inc. | www.vishay.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

MCU-Based Data Logger

by **Brian Beard**

Circuit Cellar 266, 2012

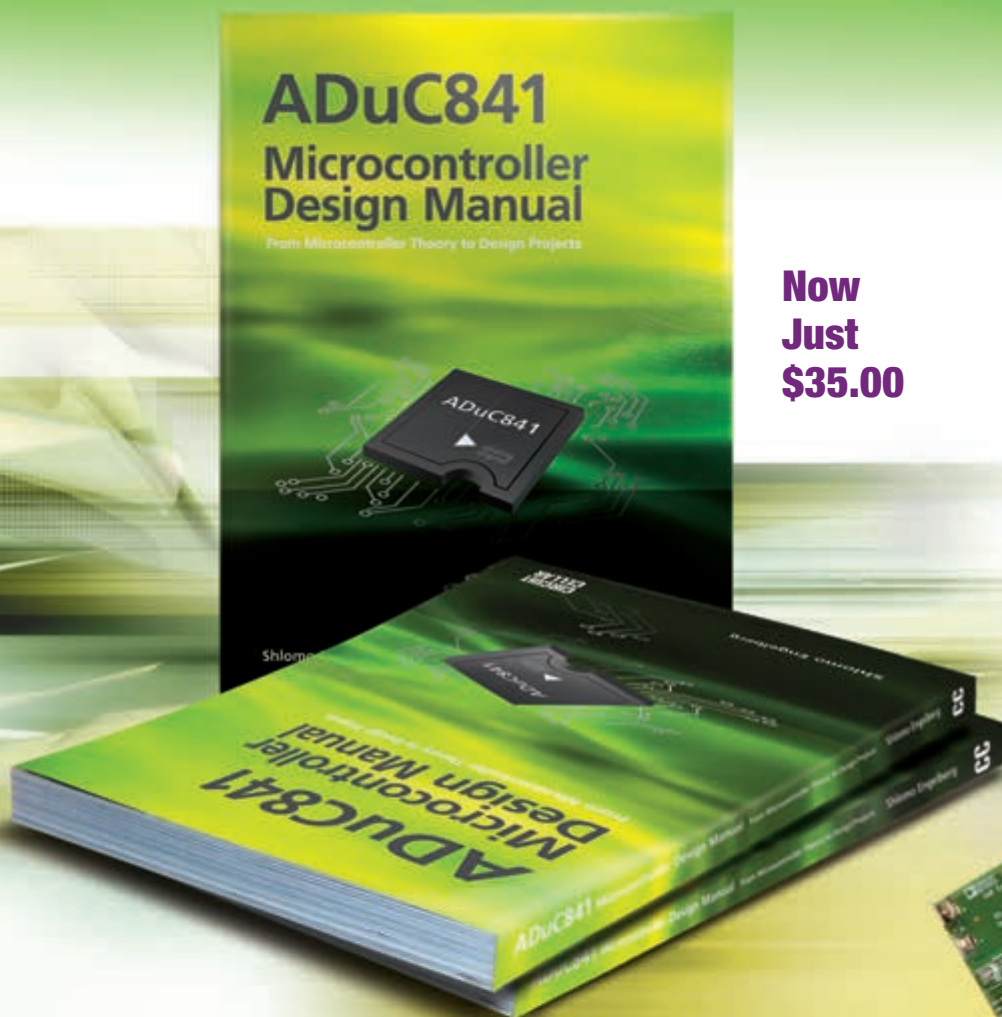
You can plan and construct a flexible environmental data logger. This article details a battery-powered data logger design featuring sensors for temperature, barometric pressure, relative humidity, light, and switches. The logger uses any terminal emulation program for control and data transfer. Topics: Data Logging, Sensor

Go to *Circuit Cellar's* webshop to find this article and more: www.cc-webshop.com

CIRCUIT CELLAR

ADuC841 Microcontroller Design Manual: From Microcontroller Theory to Design Projects

If you've ever wanted to design and program with the ADuC841 microcontroller, or other microcontrollers in the 8051 family, this is the book for you. With introductory and advanced labs, you'll soon master the many ways to use a microcontroller. Perfect for academics!



**Now
Just
\$35.00**

Buy it today!

www.cc-webshop.com

DIY Function Generator

You can build a flexible function generator with multiple simultaneous outputs, direct digital synthesis, and a flexible menu that uses a four-line character LCD, a joystick, and a quadrature encoder. This generator's design includes a microprocessor and custom PCBs.

I developed a flexible function generator based on Digi-Key's Rabbit MiniCore RCM5700, which has pulse-width modulation (PWM) and pulse position modulation (PPM) systems. My system has two PWM outputs (although the RCM5700 has four) and four PPM outputs. All six are simultaneously available. The two PWM and the four PPM outputs will always have the same frequency, but the PWM and PPM systems can be different frequencies. The direct digital synthesis (DDS) is completely independent of the Rabbit, except for its programming.

I wanted my generator to include multiple simultaneous outputs; PWM, PPM, and arbitrary bitstream (ARB) pulses; DDS, including sine and triangle; and a flexible menu system using a four-line character LCD, a joystick, and a quadrature encoder. The system's main hardware components include a microprocessor module and custom PCBs. Figure 1 is a block diagram of the entire system.

PWM

I wanted the pulse generator to be as flexible as possible. With that as the goal, the PWM includes a 95-Hz-to-97.6-kHz programmable frequency (or period) and a programmable duty cycle. To control the pulse width, you can select on count from 1 to 256 or 1,024 (depending on frequency), pulse width in microseconds, duty cycle in percentage, or phase in degrees. The PWM frequency range is based on

the CPU clock frequency, in this case 50 MHz. The formula is normally $F_{\text{PWM}} = F_{\text{CPU}}/2/1,024/N$, where $1 \leq N \leq 256$. Using this formula, the frequency range is 95 Hz to 24.4 kHz.

To get a higher frequency output, the program uses a PWM feature that multiplies the frequency by a factor of four but also reduces the pulse-width resolution by the same factor. This yields an approximately 97.6-kHz high-frequency limit. You can use the menu to set the frequency at which the system enables/disables the $\times 4$ feature. Below this frequency, the pulse width has a one part in 1,024 resolution. Above this frequency, the resolution is one part in 256.

Although the $\times 4$ circuit may be enabled or disabled individually for both outputs, the program does not implement this. If this was to be implemented, one channel's output frequency would be four times that of the other, but only over a specific range of frequencies.

The PWM (and the PPM) are based on integer submultiples of the CPU clock, so their generation is restricted because their frequency resolution varies with the absolute value programmed. Table 1 shows a few examples with different values for N ($\times 4$ circuit disabled). At higher PWM frequencies, there is not much frequency resolution. At lower frequencies, there is much finer frequency resolution. Also, notice that the period resolution is fixed at about 40.9 μs . The program measures the CPU frequency so it

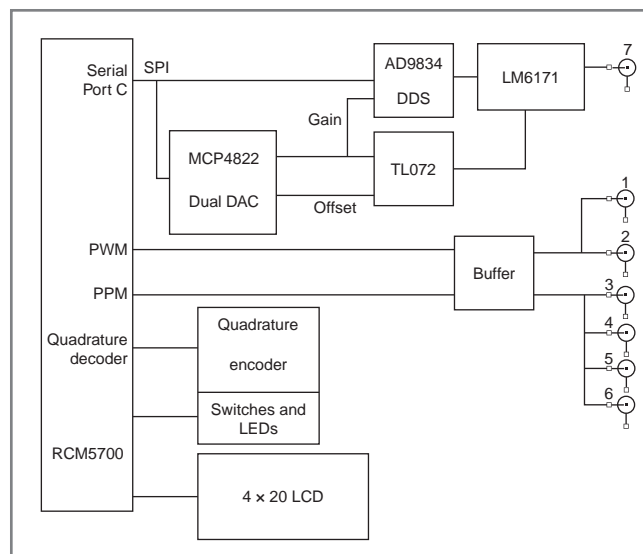


Figure 1—The function generator is based on Digi-Key's Rabbit MiniCore RCM5700. The system features two PWM outputs and four PPM outputs, all of which are simultaneously available.

N	Frequency	Period (μ s)
1	24.4 kHz	40.96
2	12.2 kHz	81.92
50	488 Hz	2,048
51	478.7 Hz	2,089

Table 1—The PWM (and the PPM) are based on integer submultiples of the CPU clock. Here are some examples with different values for N ($\times 4$ circuit disabled).

can make the calculations required to implement and display the actual value for the parameter being modified.

Photo 1 shows the function generator's front panel with the PPM frequency menu. Line 2 of the display shows the frequency as entered via the quadrature encoder and joystick. Line 3 shows the actual frequency. Line 4 shows the actual period. As you can see, the entered frequency is 1,117 Hz, but the actual frequency is 1,162 Hz, which is the closest the system can get with its 50-MHz crystal and the fixed 1,024 divider.

Figure 2 is a block diagram of the Rabbit microprocessor's PWM subsystem. Timer A9 and the up counter are common to all PWM outputs.

The pulse-width resolution in degrees or duty cycle is dependent on the selected frequency. This can easily be seen from the block diagram. The phase angle and duty cycle resolutions will be either one part in 256 or one part in 1,024, depending on whether the output frequency is above or below the $\times 4$ cutoff frequency. Also, when using either of these two parameters to program the pulse width, they will remain constant (within the limits of the resolution) as the frequency/period is varied.

PPM

The feature set for PPM is similar to those of PWM, but with significant additional capabilities. The PPM features a 1.5-Hz-to-12.5-MHz programmable frequency (or period), programmable pulse start position, and programmable pulse stop position.

The resolution for all programmable parameters affecting the pulse width and position (except count) varies with frequency in the same way as the PWM—higher frequencies have less resolution. However, there are a few more ways to independently program each channel's output pulse. You can vary its start position by count (i.e., the number of clock pulses before the output goes high), phase in degrees, or delay in microseconds. Or, you can vary its stop position by count (i.e., number of clock pulses before the output goes low), phase in degrees, delay in microseconds, pulse width in microseconds, or duty cycle in percentage.

Figure 3 is a block diagram of the Rabbit's PPM subsystem. It is also referred to as the Timer C system. As you can see, there is independent control of when the output is set and reset. The program makes all the calculations necessary based on frequency and the start and stop criteria, which can be independently set. The only restriction is that both the start and stop count values must be less than the 16-bit divider value, which is used to program the frequency.

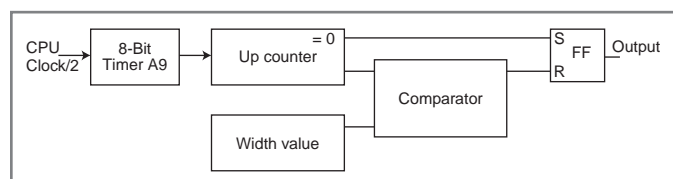


Figure 2—The Rabbit microprocessor's PWM subsystem includes Timer A9 and the up counter, which are common to all PWM outputs.



Photo 1—The function generator's front panel features a PPM frequency menu.

When any of the parameters are varied, other appropriate parameters are also varied. For example, if you have selected a 50% duty cycle and a 5- μ s start delay time, you can vary the frequency as much as desired, but the duty cycle and start time will remain constant as long as the frequency is below 100 kHz. If the frequency is above 100 kHz, the period is less than 10 μ s, so it becomes impossible to have both a 5- μ s delay and a 50% duty cycle. As another example, selecting a 10° start phase and a 100° stop phase, you can change the frequency as much as you want (up to about 694 kHz) and the start and stop phases will remain constant (enabling resolution changes). The reason for the upper limit is because 10° resolution is needed, which means the selected frequency must have at least 36 clock cycles driving the 16-bit counter. If the 16-bit counter is set to 36, the output frequency will be 694 kHz (i.e., CPU frequency/2/36).

ARB

The ARB function uses the SPI system to generate a bitstream that is dependent on user-entered values. The ARB signal is on the same Rabbit I/O pin, as is PPM0. The output (PPM or ARB) is determined by which feature the user has most recently selected. The other PPM signals are not affected by using the ARB. To enable this feature, I had to find an output on the microprocessor that supports both a PPM channel and an SPI output. The ARB has only three programmable features: frequency, bit pattern, and number of times to repeat the pattern (continuous is selected by entering a negative number).

The frequency is controlled by programming the SPI

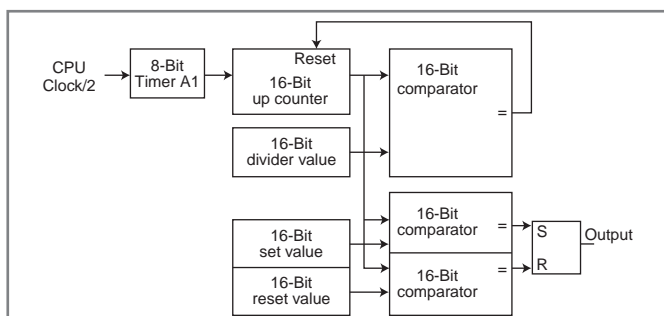


Figure 3—The Rabbit's PPM subsystem is known as the Timer C system.

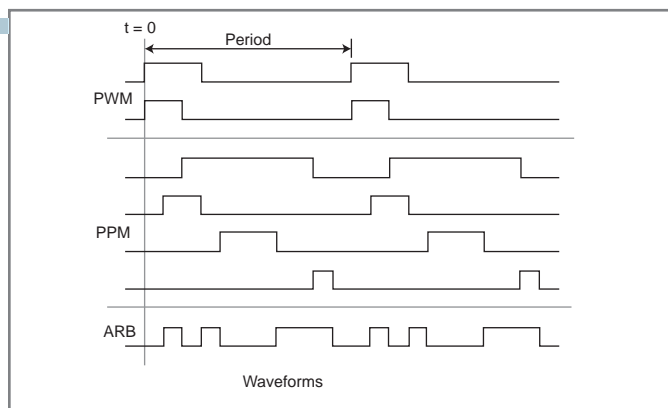


Figure 4—Sample waveforms available from the PWM, PPM, and ARB systems are all shown with the same period. However, each output type can have a different period/frequency value.

channel's bit rate. The bit pattern is obtained from a file (C:\ARBdata.txt), which is loaded into the program image at compile time. However, there is a sample pattern of 0xAA, 0xF0, which can be selected to see what an ARB signal might look like. The ARB pattern can be edited using the menu, but it cannot be saved back to the PC. The buffered ARB clock is also available on the circuit board, but I did not bring it out to a connector on the unit I built. Comments in the source file fully describe the ARBdata.txt format.

WAVEFORMS

Figure 4 is a diagram of sample waveforms available from the PWM, PPM, and ARB systems. Even though the figure shows all the waveforms having the same period, each different output type can have a different period/frequency value.

DDS

I implemented a DDS system that uses an Analog Devices AD9834, which is the same integrated circuit (IC) I wrote about in "Stand-Alone DDS Unit" (*Nuts & Volts*, November 2006). The major enhancement I implemented is that this circuit has both amplitude and offset capability. There is also a high-speed buffer amplifier with a gain of two, which increases the output amplitude and protects the DDS IC. The software sets the frequency range to 1 Hz to 20 MHz and the resolution to 1 Hz. The frequency range and the resolution can be modified by editing the source code.

Figure 5 is a simplified schematic of the circuit used to control the amplitude and offset. The component references are not the same as in the schematic. They have been renumbered here to make the description easier to follow. I used a spreadsheet (Gain_Offset.ods) to verify the calculations.

IC1 is a Microchip Technology MCP4822, which is a dual-channel, 12-bit DAC controlled via SPI. IC2 is a STMicroelectronics TS912, which is a rail-to-rail CMOS dual op-amp. Only one section is used to generate the appropriate offset voltage for IC4. The DDS IC (circuit not shown) is an Analog Devices AD9834, which is a 75-MHz DDS IC also controlled by SPI. It is capable of sine, triangle, and square-wave outputs. IC4 is a Texas Instruments LM6171, a high-speed op-amp, which is used to amplify and buffer the AD9834 and to insert the offset voltage.

As noted on the schematic, the AD9834's peak-to-peak output

voltage is controlled by DAC0. The AD9834 output current is:

$$I_{PP} = 18 \times \frac{(V_{REF} - V_{DAC})}{R_{SET}}$$

V_{REF} is internally generated by the AD9834 and is approximately 1.2 V. I made R_{SET} 7.2 k Ω to make the calculations easier. That makes the formula:

$$I_{PP} = 18 \times \frac{(1.2 - V_{DAC})}{7,200}$$

Therefore the output voltage across the 200- Ω resistor is:

$$V_{PP} = 200 \times 18 \times \frac{(1.2 - V_{DAC})}{7,200} = \frac{(1.2 - V_{DAC})}{2}$$

(Circuit details are available in Analog Devices's "Amplitude Control Circuit for AD9834 Waveform Generator (DDS)" circuit note.) The AD9834's output is such that the signal's "bottom" is always ground. Therefore, the average output voltage is $V_{AVG} = V_{PP}/2 = (1.2 - V_{DAC})/4$.

The LM6171's configuration is such that it multiplies the AD9834 voltage by three—assuming the output at IC2.1 is AC ground. The TS912's output voltage is:

$$V_1 = V_{DAC1} - \left(\frac{R_2}{R_1} \right) \times (5 - V_{DAC1}) - \left(\frac{R_2}{R_3} \right) \times (V_{DAC0} - V_{DAC1})$$

The LM6171's output is:

$$V_2 = V_{AVG} - \left(\frac{R_5}{R_4} \right) \times (V_1 - V_{AVG})$$

I used the following criteria to calculate the resistors: (1) The LM6171 gain for the DDS signal (V_{AVG}) is 3. This criterion is somewhat arbitrary. The main reasons for this selection were that a gain of 3 enables linear operation up to about 20 MHz. Higher gains would lower the output frequency range. Also, a lower gain causes a lower offset voltage range. (2) The average output voltage should be independent of V_{DAC0} (gain DAC). (3) The average output voltage should be symmetrical about 0 (equal positive and negative offsets).

Using (1), $R_5/R_4 = 2$. Their absolute values (theoretically) do not matter. However, from a practical perspective, they should be relatively small. I chose $R_4 = 200 \Omega$ and $R_5 = 400 \Omega$. Lower values would force the op-amps to supply more current and high

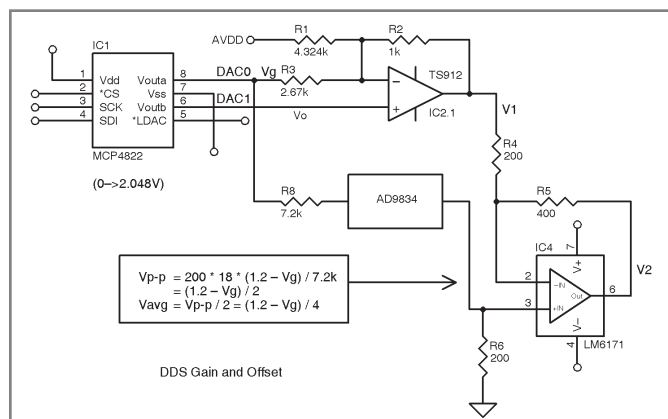


Figure 5—A schematic of the DDS gain and offset Analog Devices AD9834 circuit is shown.

FGen Main Menu+
DDS+
Channel+
Channel#
Frequency
Phase
Amplitude
Offset
Waveform
Calibrate+
Amplitude
Offset
PWM+
x4 Frequency
Frequency
Channel+
Channel#
On Count
Phase Width
Pulse Width
Duty Cycle
PPM+
Frequency
Channel+
Channel#
Start+
Start Count
Start Phase
Start Delay
Stop+
Stop Count
Stop Phase
Stop Delay
Pulse Width
Duty Cycle
ARB+
Frequency
Cycles
Byte Count
Data
State
Read ARB File
Sample
Settings+
Show Settings
Save Settings
Read Settings
Set Default
CPU Freq. Corr

Figure 6—The menu system uses a 4 × 20 LCD, a four-position joystick with a center push switch, and a quadrature encoder. The menu and LCD can be independently used for other projects. This is a current menu example.

values can adversely affect the frequency response due to R-C time constants.

Using (2):

$$R3 = 8 \times \frac{R2}{3}$$

I selected $R2 = 1 \text{ k}\Omega$, which yields a value of $2.667 \text{ k}\Omega$ for $R3$.

Using (3):

$$R1 = 16 \times \frac{R2}{3.7}$$

This makes $R1 = 4.32 \text{ k}\Omega$. Note that the offset voltage's accuracy will only be as accurate as the 5-V supply and the resistor ratios. With these values, the offset range is approximately $\pm 3.2 \text{ V}$. I usually use 1% SMD resistors since they cost the same as 5% parts.

There is a low-pass filter between the DDS IC and the buffer. It eliminates some of the 50-MHz component from the

output signal. The remaining 50-MHz signal is about 1 mV_{pp} . Most of this could be eliminated by inserting another filter between the buffer and the output connector. I did not find this little bit a problem with how I am using the system. I had attempted a more aggressive filter arrangement with no improvement over the final design. There are two capacitor locations that are not used as noted on the schematic.

THE MENU SYSTEM

The menu system is very flexible. It is a hierarchical menu that features up to 100 total items. Each "parent" can have up to 10 "children" and there are up to five nesting levels. Each value can be changed by editing the appropriate `#define` declaration in the menu library file. Even though the code was written for this project—the Rabbit 5000 processor and a parallel interfaced LCD—the code should be portable to easily use any processor and character display. The software is written to use libraries so the main menu code should not have to be changed. Most of the effort would be in

creating a library for the low-level LCD functions. I used Assembly language for some of the most frequently executed routines. The remainder of the program is in C.

There are functions that enable you to enter and modify the following data types: integer (16-bit signed), long integer (32-bit signed), floating point, selection from a list of text items, text value, and IP address. A text item list example is the days of the week. The text value can be used to enter visible or invisible text. The invisible feature is useful for entering passwords. The IP address is useful for entering Internet addresses (not used in this program).

One "nice" feature of the menu system is that any numeric parameter being modified can be instantly implemented by the hardware. This enables you to immediately see the effect of the new values as you are making them.

The menu system is designed to use a 4 × 20 LCD, a four-position joystick with a center push switch, and a quadrature encoder also with a center push switch. The menu and LCD functions are

mbed

mbed NXP LPC11U24 Microcontroller

Rapid prototyping for USB Devices, Battery Powered designs and 32-bit ARM® Cortex™-M0 applications

<http://mbed.org>

BEST SCOPE SELECTION & lowest prices!

WORLD'S SMALLEST

World's smallest MSOI
This DIP-sized 200kHz
2-ch scope includes a
spectrum analyzer and
Arbitrary Waveform Gen.
Measures only 1 x 1.6 inches in size!



XPROTOLAB \$49



IPHONE SCOPE

5MHz mixed signal
scope adapter for the
iPhone, iPad and iPod Touch!
The FREE IMSO-104 app is available for
download from the Apple App Store.

IMSO-104 \$297.99

30MHZ SCOPE

Remarkable low
cost 30MHz, 2-ch
250MS/s sample
rate oscilloscope.
8-in color TFT-LCD
and AutoScale function. Includes
FREE carry case and 3 year warranty!



SDS5032E \$299



60MHZ SCOPE

60MHz 2-ch scope
with 500MSa/s rate
and huge 10MSa memory!
8" color TFT-LCD and FREE carry case!

SDS6062 \$349

100MHZ SCOPE

High-end 100MHz 2-ch
1GSa/s benchscope
with 1MSa memory
and USB port • FREE
scope carry case. New super low price!



DS1102E \$399



100MHZ MSO

2-ch 100MSa/s
scope • 8-ch logic
analyzer. USB 2.0 and
4M samples storage per channel with
advanced triggering & math functions.

CS328A \$1359

HANDHELD 20MHZ

Fast & accurate handheld
20MHz 1-ch oscilloscope.
• 100 M/S sample rate
• 3.5 in. color TFT-LCD
• 6 hour battery life
FREE rugged, impact-resistant case!



HDS1021M \$269.95

WWW.SAELIG.COM



Saelig
unique electronics



in separate libraries so they can be independently used for other projects. The menu functions have been written so each major function has both a blocking and non-blocking entry point.

Figure 6 shows the current menu. Those entries with "+" suffixes are parent entries. They do not cause a function to execute. The menu system does not force this syntax. I implemented it to make it more obvious so the user can determine "parent" entries when navigating the menu. A menu system limitation is that any menu entry that has "children" cannot execute a function.

During the program development, I was able to easily change the menu item order by simply "cutting" an entry and "pasting" it into the desired location. There are several more parameters for each entry that are not shown here (e.g., what function to execute), but are well documented in the menu library comments.

To navigate the menu, first use the joystick to move down (move down one entry), then up (move up one entry), then left (move up one menu level), then right or push in (select this item). Once you have navigated to the "child" you want to modify, you can use the joystick alone or in combination with the quadrature encoder.

The joystick operates on numeric values as follows: left (move the cursor one position to the left), right (move the cursor one position to the right), up (increment the digit, rolls over from 9 to 0), down (decrement the digit, rolls over from 0 to 9), and push in (select this value and exit this "child").

The quadrature encoder operates on numeric values as follows: rotate clockwise (add the power of 10 of the current cursor position to the value), rotate counter clockwise (subtract the power of 10 of the current cursor position from the value), and push in (select this value and exit this "child"). Note: the "power of 10" becomes a "power of 16" when modifying the ARB data.

Here is an example of using the encoder: If you place the cursor on the hundreds digit, rotating the encoder one position will add/subtract 100 to/from the value. When you rotate clockwise one position and the digit is 9, the digit will become 0, but the next more significant digit will increment.

HARDWARE

As stated earlier, the system is based on a Rabbit RCM5700. It will also work with an RCM6700, but it will yield higher maximum and minimum frequencies for the PWM, PPM, and ARB. The RCM6700 contains a method to bring the frequencies lower, but I have not implemented it.

My first system implementation used the baseboard that is available for the RCM5700 to keep construction difficulties to a minimum. My final design uses three custom PCBs and includes a main board with a socket for the RCM5700; a Hitachi HD44780 LCD connector using either an 8-bit or 4-bit interface; a multiplexed and buffered digital I/O for the LEDs and joystick; buffered pulse outputs, voltage regulators, and a connector for a DDS "daughterboard;" a DDS with amplitude and offset circuits; and switches and LEDs.

SOFTWARE

I used Digi International's Dynamic C IDE to develop the software. It is the development environment for the RCM5700 used for this project and is available as a free download from Digi International's website. Dynamic C V10 is completely ANSI 99-compliant, except for bit fields that are not required for this project.

Many of the functions use costates, which is a non-ANSI feature. This is a relatively easy way to create software state machines. If you want to port the code to another CPU, you will have to remove or replace the costate functionality. This feature is primarily used in the main program and the menu library. Using costates enabled me to write non-blocking functions so other processes can be executed while waiting for operator input. This is not important in this application, but I have used this same menu library in other applications where using non-blocking functions was required.

Most, if not all, of your modifications should be in the -Fgen_IO_defs.C file. It contains a number of hardware-related definitions that enable the modification of specific features (e.g., the serial port to use for SPI, which bits to use for various control purposes, etc.). Having these definitions enables you to modify the hardware without having to alter the program.

There is a single function in the file that is for all microprocessor I/O's custom initialization. If the I/O definitions are changed, there is a possibility that this function will have to be modified.

The -FunctionGenerator.C file is the system's main program. It contains the menu structure and all the functions specific to the menu selections.

The -Menu.Lib file includes the functions specific to handling the menu. There is nothing in this library that is specific to the function generator.

The -HD44780.Lib file includes the functions that handle the LCD. This is mostly written in C, but there are a few cases where Assembly language is used to minimize execution time. One of these cases is the function that writes byte commands or data to the LCD. This is the function that is most executed, so I wanted it to be as fast as possible. This file also contains the functions required for the quadrature decoder, joystick, and LEDs. I wanted to keep all the functions required to handle the hardware used by the menu system in a single file.

The -MCP4822.Lib file contains the few functions required to control the MCP4822 Dual DAC, while the -AD9834.Lib file contains the functions to control the AD9834 DDS IC.

The -Fser.Lib file contains the functions for general-purpose, serial I/O operations and is capable of both synchronous (SPI) and asynchronous serial I/O. The MCP4822 and AD9834 are both controlled via SPI. This file is also used to generate the ARB bitstream. Most of the API functions are written in C. However, the interrupt service routines (ISRs) are written in Assembly to minimize execution time.

GENERAL CONSTRUCTION

I used BatchPCB (for the first time) for this project and was quite pleased at the boards' quality. The only drawback is that delivery can be a little slow. There are three job ID numbers you will need to place an order, one for each board. However, you only need to pay a single set-up fee if all the boards are ordered at the same time. Check my website—*K3PTO Published Articles* (www.qsl.net/k3pto)—for the job ID numbers and to see if there have been any updates to the boards or any of the files.

I used Novarm's DipTrace PCB design

software to draw the schematics and layout the PCBs. All the design files are available from *Circuit Cellar's* FTP site and from my website.

I like using connectors for all off-board connections. They make it relatively easy to assemble and mount the boards. I use connectors that cost about \$0.20 per connection when purchased in a 100-pin quantity. You don't have to use these connectors. You can simply solder wires into the pin locations. There are approximately 80 of these connectors. I used a silver marking pen to write the header number and mark the "pin 1" edge on each cable connector. Before obtaining a crimping tool, I used a pair of needle nose pliers to crimp the wires to the pins.

The chassis connectors I selected for the outputs are simply what I wanted to use (see Photo 1). There is nothing "magical" about them. I like RCA-type connectors because they are inexpensive, common, and easy to use. The order of the outputs shown in Photo 1 is: PWM1, PWM2, PPM1/ARB, PPM2, PPM3, and DDS.

The RCM5700's connector has very closely spaced pads, as do several of the surface-mount ICs. You should use fine solder (10 mils if possible) and a fine-tip soldering iron. Some solder wick and a solder sucker will also be useful. I use a lighted magnifying lens for most of my fine soldering work. I also use an eye loupe with 10× magnification to check some of the closest-spaced pads.

In general, I solder the lower-profile components first. The connectors are last since they are generally the highest profile.

The function generator has four schematic files: FunGen_Main, FunGen_DDS, FunGen_Switches, and FunGen_Chassis (the interconnections among the boards). Each is available as a DipTrace file and as a JPG file.

MAIN BOARD CONSTRUCTION

The power supply requires a 20-VAC center-tap (10-0-10) wall wart. There is a provision on the main board for applying DC voltages if you have external sources for 3.3 to 5 V and -5 V. If you do not need or want to use the DDS's offset feature, you do not need the -5-V regulator. If that is the case, you can use a 7-to-12-VDC input of instead of an AC or a 10-VAC wall wart. You will also have to connect 79L05

Access USB Devices from your Embedded Systems



USB Embedded Hosts The Developer's Guide Jan Axelsson

\$29.95

LVR.COM

From the author of *USB Complete*

Connect With Design Engineers From Around The Globe.

Reserve advertising
space in **Circuit
Cellar and CC
News Notes** today!

Strategic Media Marketing
978.281.7708
peter@smmarketing.us
www.smmarketing.us

PIC-SERVO MOTION CONTROL

MOTION CONTROLLERS FOR
BRUSH, BRUSHLESS AND
STEPPER MOTORS.

- controller chips
- controller boards

www.picservo.com

JEFFREY KERR, LLC

pins 1 and 3 where it would have been mounted.

The six pulse outputs are driven by a Texas Instruments 74LVC8T245 transceiver. This device has separate power supply pins for its input and output sides. This enables you to have 3.3 V on one side and 5 V on the other. The output-side supply pins are connected to H101, which also has connections to the 3.3- and 5-V supplies. The pins are arranged such that a simple jumper plug can be inserted to select either one of the two supplies. Also, the output side supply pins are connected to 3.3 V via a diode to ensure they are not left open.

H103 and H203 must be mounted on the backside. H103 should be a right-angle header. I suggest you also mount H201 and H204 on the backside. R103 should be mounted on the backside so you can more easily adjust the LCD contrast. I found that any voltage in the 0.5-to-1-V range yields good contrast. I had to change my design twice due to the LCD's lack of availability. The board now has a jumper (JP101), which enables you to select either 3.3 or 5 V for the LCD. Either will work with the RCM5700, as long as you do not attempt to read any data from it. Attempting to read data from a 5-V LCD will damage the RCM5700.

IC203 should be mounted flat to the board with a 4-40 nut and bolt so the copper ground plane can be used as a heatsink. The main board was mounted to the front panel via two 6-32 × 3" bolts with 2.5" of nylon standoffs on each.

LCD

I had to enlarge the LCD's mounting holes a little bit to accommodate 2-56 bolts. This particular LCD has two eight-pin connectors instead of the usual single 16-pin connector. Looking at the back of the LCD, pin 1 is the one closest to resistor RF near the PCB's top-right corner. That connector has pins 1-8. Pin 9 is also at the top of the display at the PCB's opposite end. I also marked the LCD board and cable connectors so I could easily tell which cable connector to plug into which PCB connector. I used two 0.125" nylon standoffs (0.25" would work just as well) and 2-56 hardware to mount the LCD to the front panel.

SWITCHES & LEDS

I had to improvise to put knobs on the joystick and quadrature encoder switches. I cut the shaft on the quadrature encoder to about 0.25". For the joystick, I used a hollow nylon standoff I had in my junk box to extend it a little and to make it suitable for the knob I used. I used 0.25" standoffs and 6-32 hardware to mount the switch board to the front panel. As shown in Photo 1, I drilled the holes for the mounting bolts a little bit too large and had to use some washers to fix the problem. The LEDs and switches are mounted to the front of the board and H401 on the back.

DDS

H301 must be mounted on the bottom of the board so it mates to H202 on the main board. The DDS board will then be parallel to and over the main board. There is a mounting hole to tie the boards together with 4-40 hardware and a 0.5" spacer. The DDS IC has closely spaced leads (0.65 mm or 0.025"), so you will need some magnification to solder it and check the connections. I found the most common problem is shorts between leads.

I used a through-hole 50-MHz crystal oscillator. However, to simplify the layout, I used two of the pins as if they were surface mount. You will have to bend them to solder them to the pads on the board's bottom.

I recently used one of the features (the DDS) to characterize audio filter responses. One of the problems that occurred during this system's development was "feature creep." I had to choose a set of features and stop when they were accomplished, otherwise I would still be modifying the code. During the software writing and debugging, I became more knowledgeable about how both the PWM and PPM systems in the Rabbit operate. Calculating the resistor values in the DDS gain and offset circuit was also an interesting exercise. I now have a versatile piece of test equipment I can use for both audio and digital systems. If I need it to do something more, I can add a feature to the menu and write the required function or modify an existing one. 📧

Larry Cicchinelli (k3ptoo@gmail.com) holds a BSEE from the Drexel Institute of Technology and an MSES from Pennsylvania State University. He was technical support manager at ZWorld, Rabbit Semiconductor, then Digi International (Rabbit Brand) from 2000 to 2012. From 1967 to 2000, Larry worked for Ford Motor Company. He has been licensed as K3PTO since 1961.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2013/270.

RESOURCES

Analog Devices, Inc., "Amplitude Control Circuit for AD9834 Waveform Generator (DDS)," Circuit Note CN-0156, 2010, www.analog.com/static/imported-files/circuit_notes/CN0156.pdf.

BatchPCB, <http://batchpcb.com>.

L. Cicchinelli, "Stand-Alone DDS Unit," *Nuts & Volts*, November 2006.

K3PTO Published Articles, 2011, www.qsl.net/k3ptoo.

SOURCES

AD9834 DDS

Analog Devices, Inc. | www.analog.com

Dynamic C IDE

Digi International, Inc. | www.digi.com

Rabbit MiniCore RCM5700 Serial-to-Ethernet module

Digi-Key Corp. | www.digikey.com

MCP4822 DAC

Microchip Technology, Inc. | www.microchip.com

DipTrace PCB design software

Novarm, Ltd. | www.diptrace.com

TS912 CMOS Dual op-amp

STMicroelectronics | www.st.com

74LVC8T245 Transceiver and LM6171 op-amp

Texas Instruments, Inc. | www.ti.com

Microprocessor Design Using Verilog HDL

With the right tools, such as this **new book**,
designing a microprocessor can be easy.

Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one **comprehensive guide to processor design in the real world.**

Yours for just
\$45.00

Monte demonstrates how Verilog hardware description language (HDL) enables you to **depict, simulate, and synthesize an electronic design** so you can **reduce your workload and increase productivity.**

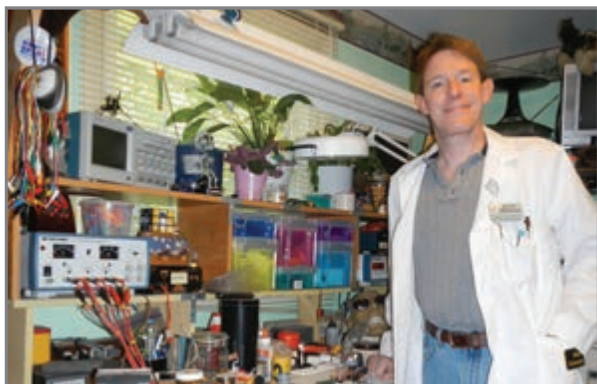
Microprocessor Design Using Verilog HDL will provide you with information about:

- Verilog HDL Review
- Verilog Coding Style
- Design Work
- Microarchitecture
- Writing in Verilog
- Debugging, Verification, and Testing
- Post Simulation and more!

www.cc-webshop.com



QUESTIONS & ANSWERS



Sensory Innovation

An Interview with Stephan Lubbers

Stephan Lubbers enjoys sensing technology. He is a creative engineer and inventor whose designs often build on his need to monitor data and figure out how things work. Steve and I recently discussed some of his designs, his contest-entry process, his thoughts on the future of embedded technology, and what's currently happening on his workbench.—Nan Price, Associate Editor

NAN: Where are you located?

STEVE: I live in Dayton, OH.

NAN: Where did you go to school and what did you study?

STEVE: My formal education is a BS in Computer Science from Wright State University, Fairborn, OH. Outside of schools, I've taught myself many things ranging from radio electronics to achieving an extra class amateur radio license, to assorted computer languages, to FPGA programming—all from just sitting down and saying, "Let's learn this."

NAN: Tell us about your current occupation.

STEVE: I am employed as a Senior Software Engineer at Beijing West Industries, where I develop embedded systems that go under the hood of high-end automobiles. (BWI is the owner of what was once General Motors's Suspension and Brakes components company.) If your "Service Vehicle Soon" light comes on, I may have written the code behind it.

NAN: Tell us about your technical interests.

STEVE: My technical interests fall into two categories. I like to build systems around new sensing technologies and I build systems to support ham radio.

I never really thought about specific technical interests until I was asked

this question. Looking at the Circuit Cellar contests I've entered and exploring my parts closet, I discovered that I have an abundance of sensors and sensor systems. When a new sensing device comes out, I often get one, play with it, and then look around for something to do with it. That usually results in an invention of some kind. I've analyzed the motion of rodeo bulls and dogs with microelectromechanical (MEMS) accelerometers, tracked eyeball movements with optical sensors, and computed automobile speeds using both GPS and microwave electronics. I don't know if it is cause or effect, but I was always amazed by the "tricorder" on *Star Trek*. Do I like sensors because of Scotty and Mr. Spock? Or did I watch *Star Trek* because of the gadgets? I don't know.

My love of electronics led me to amateur radio at a young age. I wasn't as much interested in talking to other people as I was in exploring the technology that enables people to talk. I had a little success building RF devices but found that I had a real knack for digital systems. I've used that ability to create satellite tracking controllers, antenna switchers, and computer-to-radio interfaces.

NAN: How long have you been reading *Circuit Cellar*?

STEVE: I've subscribed to *Circuit Cellar* since Issue 1. I still believe the tagline that said "Inside the Box Still Counts."

NAN: You've written four articles for *Circuit Cellar*. Some focus on data logging, monitoring, and analysis. For example, your article "Precision Motion-Sensing System Analyzer" (*Circuit Cellar* 192, 2006) is about a microcontroller-based, motion-sensing system for bull riders. What inspired you to create this system?

STEVE: Several things came together to spark the creation of the "Precision Motion-Sensing System Analyzer," a.k.a. the BuckyMeter. I had already begun work on a motion-logging system but had no clear goal in mind. Shortly after the logger started working, Circuit Cellar announced its 2005 design contest. I had a short-term goal of entering the contest with my data logger. But what should I log?

My dad provided the suggestion to strap the logger onto the back of a rodeo bull. My parents had become fascinated by the sport of professional bull riding and thought it would be fun to get behind the scenes by doing this science experiment. One of the questions I had when designing the system was: "What kind of maximum G force can I expect to see?" Nobody had an answer, but the doctors responsible for repairing bull riders thought it was an interesting question. They, too, wanted to know that answer. That question opened a few doors to give us access to some bulls. *EE Times* printed a humorous article about my experience strapping an electronic device on the back of

1,200 lb of angry cow. It was definitely an experience!

The BuckyMeter hardware went through several iterations. In the end, an off-the-shelf Motorola Z-Star evaluation module could be used to instrument the bull with the added bonus of wireless data logging.

The project died out after a trip to instrument competition-grade bulls from American Bucking Bull, Inc. (ABBI). In hindsight, I learned an important lesson about managing customer expectations. I went to Oklahoma on a mission to collect data and try out an engineering prototype. I think the people I met with were expecting to see a polished product. Their impression, after our meeting, was that an electronic scoring aid was too slow and too complicated.

NAN: Another article, "Electronic Data Logging and Analysis: A How-To Guide for Building a Seizure-Monitoring System" (*Circuit Cellar* 214, 2008), describes an Atmel ATmega32-based electronic monitoring system that enables pet owners and vets to monitor epileptic seizure patterns in dogs. How does the microcontroller factor into the design?

STEVE: My seizure monitor was an offshoot of the rodeo bull motion-sensing system. The original processor had way more power than was needed and it was difficult to hand solder the part. With a working baseline from the BuckyMeter, it was easy to pick a different chip to work with. I had some experience with Atmel AVR's from a previous *Circuit Cellar* contest, so I looked at its product line. I had a good estimate for RAM/ROM requirements, and I decided it would be nice to have additional SPI channels to interface with the accelerometers. That led to the selection of the ATmega32. It didn't hurt that another Atmel contest popped up in 2006 when I was in the middle of the design.

I have always wanted to expand my data beyond a single patient to see if my



Steve's KartTracker is based on the Renesas RX62N development kit.

theory held up, so I supplied systems to some other people with epileptic dogs. This required continuous design updates mostly to keep up with outdated parts. Unfortunately, I never got any data back from the systems I gave away. My pet (and science guinea pig) passed away a few years ago, so I don't have a subject to continue with this project.

NAN: At the end of your article, "Doppler Radar Design" (*Circuit Cellar* 243, 2010), you note that upgrades to the project (e.g., an enclosure and a portable power supply) could make the system "an easy-to-use mobile device." Tell us about the design. Did you end up implementing any of those upgrades?

STEVE: Doppler Radar Design has been my most popular project. I get e-mails all the time asking how to reproduce it. As I stated in the article, the RF section is now hard to come by and expensive. Not being an RF engineer, I haven't been able to recommend replacement parts.

The project started when my dad loaned me the microwave electronics to play with. He had wired them up for two-way ham radio communications. I couldn't manage to make any radio contact with anybody but myself, so I started looking for other experiments to perform. In one of the experiments, I learned how to make a motion detector. From that, I decided to try to turn the project into a speed radar.

This project took help from a lot of other people because I really didn't know what I was doing. Some radar

discussions on the Internet outlined the basic design for Doppler speed radar, so I followed the suggestions, essentially a transmitter/receiver pair supplied by my borrowed Gunnplexer and a frequency detector (FFT) to show the Doppler shift of the returned signal. Accounting for the radio frequency in use gives you the speed of the reflected target, which in my case was a car.

When I discovered Ramsey Electronics sells a radar kit for \$100, I decided that my

Doppler radar was really just a science experiment. It was educational for me, but for everyone who contacted me just wanting to have their own radar, the Ramsey option was cheaper, more accurate, and already packaged for portability.

I did get some helpful hints from Alan Rutz at SHF Microwave Parts Company, who suggested something called a dielectric resonator oscillator (DRO) could be used in place of the Gunnplexer I used. The advantage of his approach is that DROs are available and cost about \$20. I have not yet been successful with this upgrade.

NAN: The Renesas Electronics RX62N development board is at the heart of your KartTracker's monitoring system ("KartTracker: A GPS-Based Vehicle Timing & Monitoring System," *Circuit Cellar* 259, 2012). Tell us about the design and how the KartTracker functions.

STEVE: The KartTracker came about one day when the neighborhood NASCAR fans went out racing karts. We wondered how fast we went, so the local engineer (me) set about finding out.

I started with a GPS receiver and a data logger and drove around the track to see what happened. As it turns out, GPS receivers automatically give you your speed! That was too easy, so I started looking for more features.

The next couple of races I watched, I tried to pay attention to more than just the action and saw that teams were very concerned with lap times. Well, I could time my laps, but that didn't seem very

interesting or complicated enough. Then I saw a qualifying session where the TV showed a continuous real-time comparison between two cars. That seemed cool! If I could build that, I could race myself to see if I was doing better or worse.

So, the KartTracker concept was born. A GPS receiver feeds continuous position data into a Renesas RX62N board. The software continuously compares my time at some location against the last time I was there. It's like looking at the lap time, but it updates every couple of seconds so you have continuous feedback.

All the timing data is retained so later we can compare times against each other and brag about who went the fastest. I would like to broadcast the times back to the spectators, but that radio is a project for another day.

NAN: You received an Honorable Mention for your 2010 Texas Instruments DesignStellaris Design Contest entry, "Hands-Free USB Mouse." Tell us about the project and your contest-entry process.

STEVE: My eyePOD hands-free USB Mouse is a head-mounted motion sensor that controls the mouse cursor on a PC. By moving your head, the mouse moves around the screen. You wink your eyes to click the mouse buttons. The goal was to produce a PC interface for someone who couldn't use a typical mouse, with a secondary goal of teaching me about USB. There are some problems in certain lighting conditions, but overall it works pretty well.

After about a dozen contest entries, I have a bit of a process for creating an entry. I hope I don't hurt my future chances by sharing my secrets, but since you asked, three things need to line up for me to start a project (contest or otherwise): I need an idea, I need some technology, and I need motivation.

Author James Rollins says, "Don't ask where the ideas come from." But, if you have to know, his story ideas come from a box. My contest ideas come from a little red notebook. In reality, we don't know where the actual ideas come from, but when we get ideas we put them in the box (or book) and make a withdrawal when we need to use an idea.

Part two is that there needs to be a technology that will support the idea. I couldn't build a rodeo bull monitor until there were cheap accelerometers available. I couldn't build the KartTracker without a GPS. So, keep a list of technologies you like in your box of ideas.

Finally, you need motivation to execute the project. At work, your boss provides the motivation in the form of a paycheck. At home, you might have a dog that needs help or a neighbor who supplies beer for the answer of how fast his kart is. When I put the three pieces together, I have the starting point for a project. Apply your abilities and start building.

The only biggie after that is time management. Somewhere there is a deadline you need to meet. Do consistent work on your project and prioritize what needs to be done. I have a knack for drawing a line through the critical parts of a project to make sure I have something working when the end is near. You can always go back and improve a working project, but if you have too many half-built features, you have nothing to fall back on when time runs out. A good example is the radio link for the



The USB eyePod won Honorable Mention in the 2010 Texas Instruments DesignStellaris Design Contest. The hands-free mouse uses a head-tracking system to move the mouse cursor around the computer screen and detects eye motion to provide the button clicks.

KartTracker. Without GPS and timing software, the project would be nothing. When I had time remaining, I added file I/O and data storage on an SD card. Nice features, but they weren't necessary to demonstrate the project. The radio link fell by the wayside when entry time came up.

Finally, don't forget the book report at the end. The judges need to know what you did, so you need to write about it. Who knows? *Circuit Cellar* might like what you wrote and decide to turn it into an article.

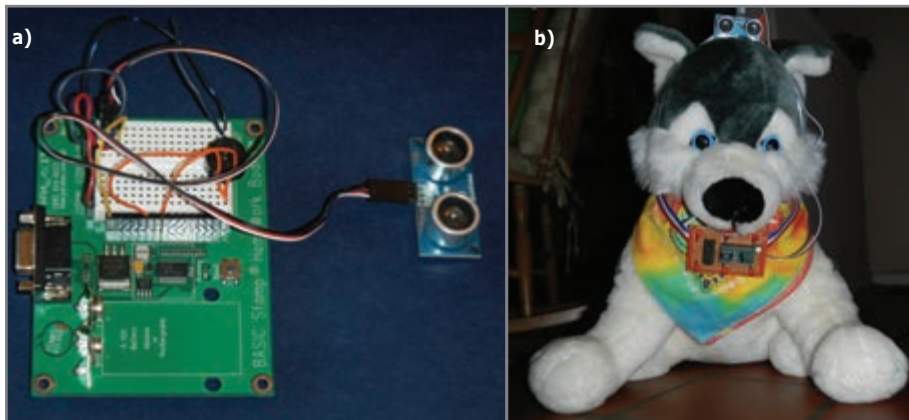
NAN: Have you recently purchased any embedded technology tools to help you with your data logging, monitoring, and analysis projects?

STEVE: My most recent tech purchase was an iPod Touch funded from a recent *Circuit Cellar* publication. Before you say, "That's not embedded," let me explain. I tend to make the user interfaces to my projects simple and to the point. Circuit Cellar contest deadlines don't lend themselves to creating a new fancy interface for each project. Instead, I would offload debugging, control, and extra features to an external system. I started out using RS-232 serial to a PC. For portability and speed, I moved to a PalmPilot with an infrared data access (IrDA) interface. A Bluetooth or Wi-Fi interface seems like a logical progression to me. The iPod Touch has these interfaces and it leaves me with a new gadget to play with.

A more embedded acquisition is the Texas Instruments MetaWatch. If you haven't seen one of these, it's a stylish digital watch that talks to your smartphone. For the more adventurous, the source code is available so you can add your own features. There must be something great that I can do with a wrist-mounted computer, I just haven't had the "ah-ha" moment yet.

NAN: Are you currently working on or planning any embedded-design-related projects?

STEVE: I call my current project the SeeingEye for a dog. The blind have used guide dogs since the 16th century. That's a huge debt man owes his best friend! To help repay that



Steve's SeeingEye prototype uses a Basic Stamp evaluation board and a PING sonar sensor **(a)**. Here is the SeeingEye mounted on a dog **(b)**. The PING sonar sensor hooks on the dog's head and faces forward to where obstacles will be encountered. On the dog's collar is a breadboarded circuit using a Basic Stamp 2. (Since real dogs don't like to pose, Steve says he had to use a stunt double.)

debt, I'm creating a twist on the seeing eye dog by creating a seeing eye for a friend's vision-impaired dog. Using the sensors and technology robots use for collision avoidance, the SeeingEye will detect obstacles in a dog's path. The trick seems to be the user interface to convey the collision avoidance information and training the dog to respond correctly to the stimulus. I figure if microchips in robots can learn to avoid walls, then puppy neurons should be able to do the same thing. I still have more work to do to figure out how to get the sensor to stay in place.

NAN: Do you have any thoughts on the future of embedded technology?

STEVE: As a builder of embedded systems, I am amazed at all of the things we can do with high-speed processors and multiple megabytes of memory. It seems like if we can imagine it, we can build it.

As a user of embedded technologies, it sometimes seems like the engineers are trying to be too clever by stuffing anything they can into the box whether those features are needed or not.

The complexity of some devices has skyrocketed to the point that stability has been affected and users don't know what features they have or how

to use them. We now take for granted a constant stream of software updates to our devices and press reset when it doesn't work as desired.

Einstein is credited with saying, "Everything should be made as simple as possible, but no simpler." I'd like to see the industry adopt Einstein's advice and the "Keep it simple, stupid!" (KISS) principle to help us manage the growing complexities. We'd spend less time serving our devices by trying to make them work and more time being served by our devices as they flawlessly do the work we want done. 📺



THE ORIGINAL SINCE 1994

PCB-POOL

Beta LAYOUT



FREE Stencil

with every prototype order



EAGLE order button

pcb-pool.com/download-button

20% off! on your first order

Call Tyler: 1 707 447 7744
sales@pcb-pool.us



PCB-POOL® is a registered trademark of

Beta

LAYOUT


www.pcb-pool.com

CS328A-XS


100MHz Mixed
Signal Oscilloscope
+ Signal Generator

Capture 200 G Samples

Other scopes capture M Samples. Ours captures days. Find intermittent problems. Then zoom with usec precision.



14 Bits 100 MSPS MSO



www.cleverscope.com



Failure Mode and Criticality Analysis

Failure mode and criticality analysis (FMECA) is an important element of any design. It uncovers potential performance problems and helps you modify your design to prevent or mitigate problems. FMECA also verifies your finished product is safe and working as desired.

Failure mode and criticality analysis (FMECA—pronounced *fem-ika*) provides a bottom-up analysis of component failures and their effects on a system. FMECA starts at the component or functional block level, then feeds up into subsystems, systems, and so forth until an entire system FMECA has been compiled. You can use FMECA to trace how a single screw or a resistor failure or a resistor affects an automobile's operation. Without FMECA, Apollo 13 astronauts would have never made it back to earth. For more details about FMECA, consult the References section at the end of this article.

The best way to begin is with MIL-STD-1629A, the free U.S. military standard. Fundamental FMECA concepts can be found in document AD-A278-508, "Failure Mode, Effects, and Criticality Analysis," which is also available as a free download. It would be a mistake to presume FMECA has no application outside of military or aerospace. The military must be credited with much of the pioneering work we can benefit from today. But, without FMECA, the automotive industry

would be mired in lawsuits, as would all others that manufacture equipment where reliability and safety are important factors (e.g., medical, nuclear, etc.). I believe every engineered product should be subjected to this analysis. The benefits speak for themselves.

CALCULATING RELIABILITY

Most of today's computer programs for calculating reliability such as those from Advanced Logistics Development (ALD), SoHaR, and Parametric Technology, contain a module to generate FMECA (unfortunately, ALD's free MTBF calculator does not include it). However, you can use a spreadsheet or a word processor to easily perform FMECA.

Companies may have different formats for tabulating FMECA data. Table 1 and Table 2 are typical worksheets I've used. The first decision is whether to perform the analysis on a component or a functional block level—this is often dictated by the customer. This decision appears as the Item in the worksheet. Table 1 shows that only one line is

Item	Function	Failure Mode(s)	Failure Effect	λ	Criticality	Compensation
Power supply	Provides all required system voltages	No output or output outside specified limits	System does not work and may be damaged	2.189	E-iv	Power is monitored, a fault results in the system's deenergization
Other block

Table 1—This is a functional block FMECA worksheet. Only one line is needed in a system containing the power supply.

Item	Function	Failure Mode	Failure Effect	λ	Criticality	Compensation
U1	Switching regulator generates system voltages	No output or output outside specified limits	System does not work or may be damaged	1.162	E-iv	Voltages are monitored, a fault results in the shutdown
L1	Choke to suppress conducted emissions	Open	Loss of power	3.80E-006	E-iv	Detected, loss of functionality
		Short	Increased EMI-conducted emissions	3.30E-007	E-iv	Periodic maintenance tests
R1	Resistor to bias the gyrator	Open	Loss of power	3.30E-003	E-iv	Detected, system shutdown
		Value changed	Doesn't matter within certain limits, outside the limits, detected as open or short	2.20E-003	E-iv	Detected as either short or open
		Short	Destruction of several components	5.50E-005	E-iv	Detected, system shutdown
R2

Table 2—Here is an example of component-level FMECA.

needed in a system containing the power supply. Figure 1 shows the schematic diagram.

FMECA

Sometimes FMECA is required at the component level, which can take much more time to prepare. Quite often, though, you end up combining the two approaches. The power supply in Figure 1 contains a switching regulator U1, which is a purchased item. You must treat it as a functional block (see Table 2).

Figure 1 is the power supply's schematic diagram, which the FMECA can treat as a functional block as shown in Table 1, or analyze at the component level, as shown in Table 2. Needless to say, the analyst must thoroughly understand the design.

Column 2 in the FMECA worksheet describes the Item Function, while Column 3 describes the Failure Mode(s). Functional blocks and components can fail in several different modes. For example, resistors can open up, short out, or undergo value changes. Each failure mode can have different consequences, which are described in the Failure Effect column.

The column headed λ (Greek letter lambda) is the failure rate of the component or the block per million hours of

operation calculated during reliability prediction. Resistor R1, for example, would have a failure rate $\lambda = 5.55 \times 10^{-3}$, which is distributed among specific failure modes. Open-circuit and parameter-change failures have almost the same probability, while short circuit accounts for a mere 1% of the total. Based on the failure rates λ , failures can be categorized by their occurrence probability:

- Level A: Frequent
- Level B: Reasonably probable
- Level C: Occasional
- Level D: Remote
- Level E: Extremely unlikely

Here, once again, the analyst's experience and judgment are paramount for assigning the appropriate levels.

Skipping the Criticality column for the moment, the Compensation column shows what to do to circumvent or mitigate the failure. Redundancy, power-up diagnostics, built-in test (BIT), and operator action are all considered. This is an extremely valuable part of the FMECA. It identifies potential performance issues, how to rectify them, what the BIT should

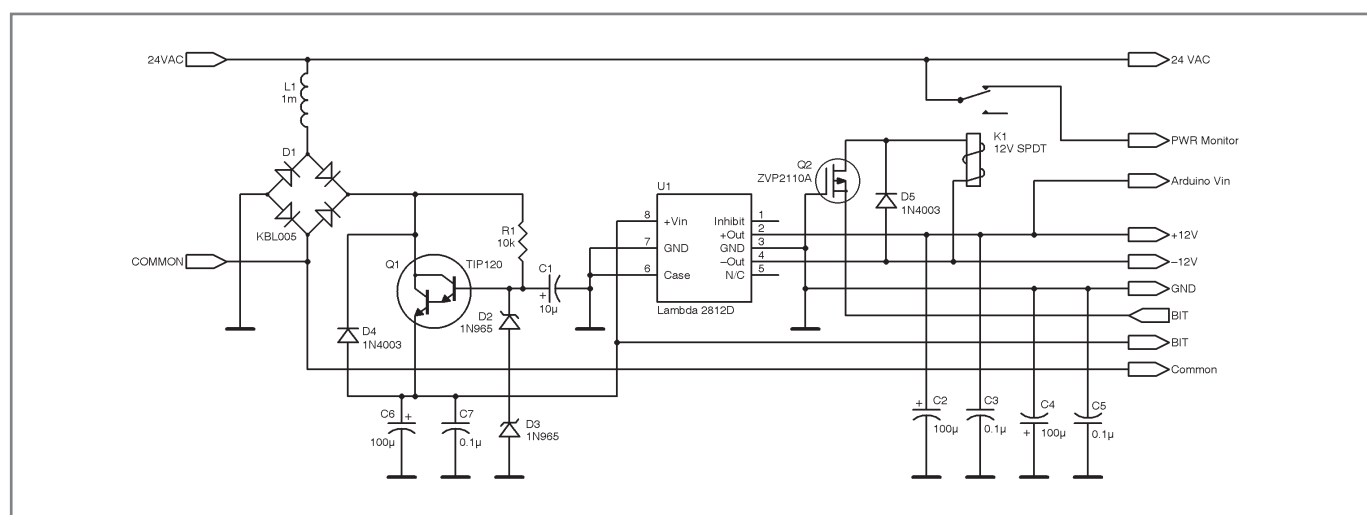


Figure 1—This schematic shows a FMECA-analyzed power supply.

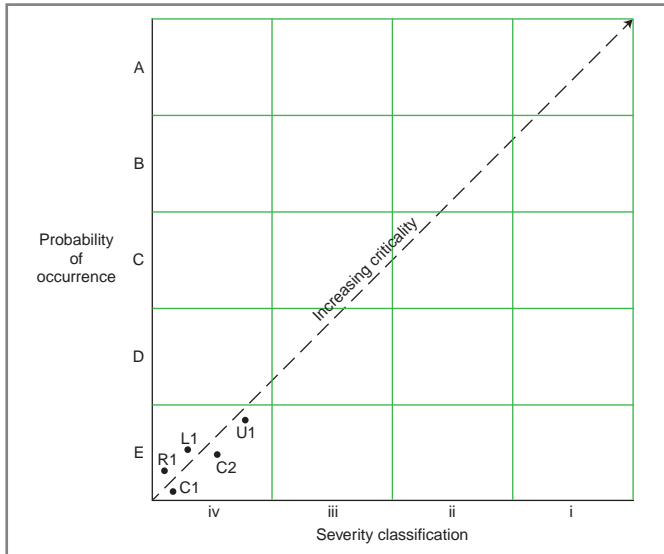


Figure 2—The criticality matrix shows a system's occurrence probability, increasing criticality, and severity classification.

detect, and how the system should react in case of a fault. Many design details are derived from the FMECA. Therefore, it must be performed during the preliminary design stage. It should come as no surprise that the desired equipment functionality is typically achieved with only about one third of the hardware and software, while the remaining two thirds are to a large extent the result of the FMECA to ensure the equipment's predictable, safe behavior.

FAULT LEVELS

For products where criticality is not an issue, this column is omitted, and the FMECA is called FMEA. When the criticality is of concern, four fault severity categories are determined: catastrophic, which causes death or a system loss; critical, which results in severe injury, severe occupational illness, or major system damage; marginal, which causes minor injury, minor occupational illness, or minor system damage; and minor, which is responsible for less than minor injury, occupational illness, or system damage.

I always strive for the E-iv category and consider any other category unacceptable. The criticality matrix helps evaluate the system's criticality (see Figure 2). To generate the criticality matrix, component identifications (e.g., R1, R2, etc.) are plotted where the abscissa represents the criticality category and the ordinate represents the occurrence frequency. The criticality must be determined for each failure mode, its probability of occurrence, its failure rate, its mission duration, and so forth. This can become overwhelming when performed on the component level with hundreds, thousands, or more components. Igor Bazovsky's book, *Reliability Theory and Practice*, is an excellent source for studying the subject.

I hope this article has provided you with a general understanding of what the FMECA is all about and how useful it is as an engineering tool. I only wish all manufacturers used it while designing their products, no matter how seemingly benign. Next month, I'll look at a complementary analysis called fault tree analysis (FTA). 📖

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

One PC and 8 programmers

Elprotronic
 Incorporated
www.elprotronic.com

PLCC Emulation Plug

Upgraded modules to PLCC Socket

- True J-lead emulation
- Low mass for easy assembly
- 20, 28, 32, 44, 52, 68 and 84 pin
- Available with alignment pins
- Volume pricing available
- <50 milliohm contact resistance

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

George Novacek (gnovacek@nexicom.net) is a professional engineer with a degree in Cybernetics and Closed-Loop Control. Now retired, he was most recently president of a multinational manufacturer for embedded control systems for aerospace applications. George wrote 26 feature articles for *Circuit Cellar* between 1999 and 2004.

RESOURCES

Advanced Logistics Development, Free MTBF Calculator, www.aldservice.com/en/reliability-software/free-mtbf-calculator.html.

I. Bazovsky, *Reliability Theory and Practice*, Prentice-Hall, 1961.

W. K. Denson, IIT Research Institute, "Reliability Assessment of Critical Electronic Components," SID17302, 1992, www.theriac.org/informationresources/demosanddownloads/Unlimited%20Distribution/Rel%20Assess%20of%20Critical%20Elect%20Components%20ADA256996.pdf.

Department of Defense, "Procedures for Performing a Failure Mode, Effects, and Criticality Analysis," MIL-STD-1629A, 1980, <http://sre.org/pubs/Mil-Std-1629A.pdf>.

———, Reliability Analysis Center, "Failure Mode, Effects, and Criticality Analysis (FMECA)," AD-A278-508, 1993, www.dtic.mil/dtic/tr/fulltext/u2/a278508.pdf.

Parametric Technology Corp., "PTC Windchill," www.ptc.com/product/windchill.

SoHaR, Inc., Free MTBF Calculator, www.sohar.com/reliability-software/free_mtb.html.

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

Product Reliability

Part 1: Reliability Prediction

by **George Novacek**

Circuit Cellar 268, 2012

Reliability, maintainability, and safety (RM&S) are important activities during product development. This article focuses on the critical task of reliability prediction. Topics: Reliability, Safety

Product Reliability

Part 2: The Meaning of Failure Rate

by **George Novacek**

Circuit Cellar 268, 2012

Now that you know some calculations for determining a product's failure rate, it's time to investigate how the numbers figure in your design. Topics: Reliability, Failure

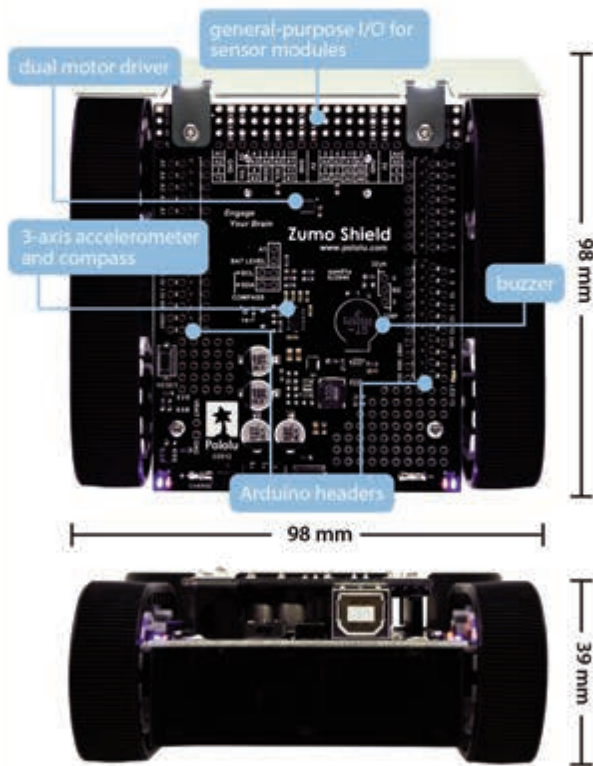
Go to *Circuit Cellar's* webshop to find these articles and more: www.cc-webshop.com

Get a little pushy.



Zumo

Small enough for Mini-Sumo;
flexible enough to make it your own.



Put your Arduino on the right tracks with the new Zumo chassis and Arduino shield.

Pololu
Robotics & Electronics

Learn more at www.pololu.com/zumo



Embedded Authentication

Authentication is the process of credential verification. Embedded systems have many authentication applications, including detecting counterfeit modules and preventing firmware tampering and illegal access. This article examines some basic authentication methods and demonstrates an implementation on a PIC32 microcontroller.

At Black Hat 2012 in Las Vegas, NV, Cody Brocious demonstrated an amazing hack. He showed how to use an Arduino board to unlock the type of hotel room door lock that is installed on approximately 4 million hotel room doors worldwide. A hotel guest typically uses a properly programmed card key to open a lock. The door lock has a data interface that support changing the room access code and other administrative functions. At Black Hat, Brocious demonstrated a significant security hole in this data interface. He used an Arduino board and a 1-Wire communication protocol to read the lock's master key code (which opens all the doors in a hotel) and unlock the hotel room door. No keycard or special code was needed! Because so many installed locks are in use, this simple, low-cost hack presents a significant problem for the vendor of this particular door lock. The door-lock hack is particularly relevant to this article, as it illustrates the importance of authentication, which verifies that someone (or something) is who he says he is. The door locks are insecure because they provide access to the master key code without verifying the requester. Somehow, the lock vendor assumed anyone who can connect a door-lock programmer to the data interface must be someone from the hotel

staff, and therefore must be trusted. But it only takes an adventurous computer engineer to prove otherwise.

In this article, I will discuss the use and implementation of authentication on embedded systems. Figure 1 shows a typical embedded system, including a microcontroller, a few peripheral chips, and an input/output (I/O) interface. The device connects to a host system, which may be remote. The embedded software—the soul of the system—makes up one end of the authentication. Depending on the authentication type needed, the software could verify if the host, the PCB, or even the microcontroller is genuine. The first case corresponds to the hotel room door-lock hack. The lock

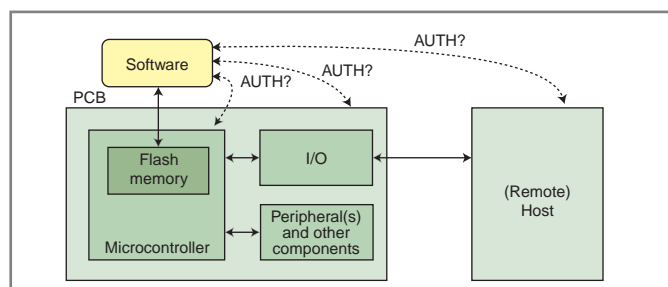


Figure 1—Embedded system authentication enables embedded software to verify its environment is genuine. This can include verification of the credentials of a remote host, the PCB or peripheral chips, or the microcontroller that executes the software. Authentication can also be done in two directions. This would enable the (remote) host to verify the software's credentials, for example, or the microcontroller's credentials.

is the embedded system and the host system is the lock programmer. Before the lock grants access to the lock programmer, it should use authentication to verify whether or not the programmer is genuine. In the second case, the software verifies whether or not the PCB is genuine. This can be done, for example, by authenticating one or more PCB peripherals. Such a test could detect a counterfeit PCB. In the third case, the software authenticates the microcontroller. This is useful to apprehend firmware tampering or firmware piracy. In each case, authentication can be one-way or mutual. In the case of mutual authentication between the embedded system and the host, for example, the embedded system will authenticate the host and the host will authenticate the embedded system. I will discuss both authentication cases.

BASIC AUTHENTICATION

There are two parties involved in an authentication protocol: the challenger and the prover. The challenger tests the prover's credentials. In the most basic authentication, the credentials are implemented as a secret value that is shared between the challenger and the prover. The challenger tests if the prover knows the shared secret's value.

Figure 2 demonstrates how this works. Before the first authentication takes place, the challenger and the prover have exchanged a shared secret, K_{AB} . This secret needs to be agreed on over a secure channel, free from eavesdroppers. The shared secret is the crux of the authentication system. If the secret is stolen, the authentication becomes pointless.

To initiate an authentication round, the prover introduces herself to the challenger through a public identifier (e.g., a name). The challenger then returns a challenge to the prover. The challenge is different for every authentication round. In Figure 2, it is referred to as a number used once (i.e., nonce). The prover creates the response by encrypting the challenge and the public identifier with the shared secret. The challenger can then test the prover's response by performing the same operation and comparing the result with the received response. Obviously, an eavesdropper who doesn't know the shared secret cannot produce the correct response and cannot impersonate the prover. The challenger must ensure the challenge is different for every authentication. Otherwise, an eavesdropper can record successful challenge/response pairs and reuse those later to impersonate the prover.

Figure 2's right side shows a mutual authentication protocol. In this case, Alice and Bob are both challenger and prover. Both of them use the shared secret key to generate a challenge and a response.

The encryption step used by the prover and challenger can be implemented using a suitable symmetric-key

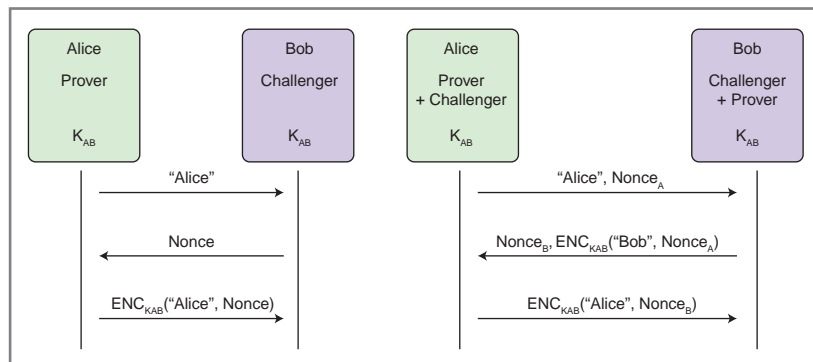


Figure 2—One-way authentication (left) and mutual authentication (right) are shown. Both protocols work on the principle of challenge-response. In a one-way authentication protocol, Alice proves to Bob that she knows a shared secret's value, K_{AB} , by using it to encrypt a challenge (i.e., nonce) provided by Bob. To prevent replay and impersonation, Bob sends Alice a nonce, which Alice encrypts. A mutual authentication protocol is made by interleaving two one-way authentications.

algorithm, such as the Advanced Encryption Standard (AES). A popular choice for the encryption step is to use a hash-based message authentication code (HMAC). I discussed HMAC in my article "One-Time Passwords from Your Watch" (*Circuit Cellar* 262, 2012). I'll briefly recall how it works. An HMAC combines a secret with a variable-length message to produce a fixed-length message authentication code (MAC). A MAC computation requires two hash function nested applications. Each time, the secret is padded onto the hash's input. The resulting MAC's length depends on the hash function type used. For example, when SHA-1 is used as hash, the MAC is 160 bits (i.e., 20 bytes).

Now, let's consider this protocol's implementation. I'll be using a SHA-1-based HMAC as the cryptographic kernel that computes $ENC_{K_{AB}}("Alice", Nonce)$ in Figure 2.

AUTHENTICATION WITH A PIC32

The authentication protocol can now be implemented on an embedded system. I will first describe a solution on a Microchip Technology PIC32 microcontroller. Figure 3 shows the setup. The host and the PIC32 store a common secret to implement the authentication protocol. The PIC32 stores the secret in program flash, rather than in RAM, since the secret needs to be available after power cycling the PIC32. Figure 3 also includes a counter in the program flash. This counter is used to generate the nonce.

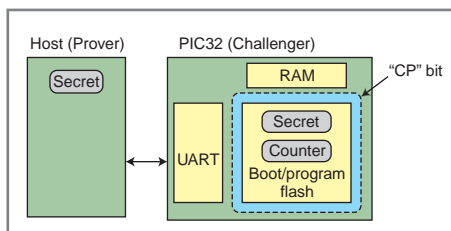


Figure 3—This shows implementation of a basic one-way authentication using a PIC32 as the challenger. The PIC stores the shared secret in nonvolatile memory (i.e., program flash). The PIC also generates a nonce. The nonce is derived from a counter, which is also stored in program flash. Both the counter and the secret need to be protected from external access. The copy-protect bit (CP bit) in the PIC's configuration bits is used to prevent such external access.

Remember, the challenger must send a nonce to the prover to test the prover's knowledge of the shared secret. Every nonce should be used only once. An eavesdropper may collect pairs of working nonces and responses and reuse them later. To ensure nonces are unique, the challenger maintains a counter that is incremented after each challenge/response round.

I used Microchip's MPLAB XC32 C compiler to develop the PIC code. The MPLAB XC32 C supports the allocation of variables into program flash by defining them as initialized `const` variables. For example, a 64-bit secret can be

allocated in program flash as follows:

```
const unsigned char secret [8] =
{0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8};
```

The secret's value can be directly defined in the C program's source code or it can be initialized using the nonvolatile memory access functions provided in the peripheral library. The shared secret's length is a design parameter and 64 bits is at the low side. The recommended length for short-term secrets is 80 bits or higher. For this implementation, I'll stick to 64 bits.

Of course, the devil is in the details. Using `const` variables is good for read-only variables, but to implement a non-volatile counter, it must also be *written* into nonvolatile memory. Flash memory is used to implement PIC32 nonvolatile memory. This type of nonvolatile memory has a particular write behavior. It can be written with "0," but not with "1." Thus, it cannot be incremented when a "0" counter value is stored in flash because it's not possible to turn a bit in flash to "1" once it has been written with a "0." How then, can a flash memory cell ever carry a "1?" When a flash memory is initialized or erased, it is reset to the "1" value.

Flash memory is physically organized in pages. Memory-erase operations work on one flash memory page at a time. In the PIC32, one flash memory page holds 1,024 words. Using this particular read/write memory behavior, I can implement a counter-stored PIC32 flash memory as shown in Listing 1. Line 1 and 2 set configuration bits to enable flash memory runtime programming and to disable external flash memory access. The latter is needed to ensure that the counter and secret value remain private to the PIC32 device, as is needed for an authentication protocol. On Listing 1 lines 4–6, a variable with a one-page flash memory length is created. The `settings` variable is organized as follows: the first two words are used to store the 64-bit shared secret and the third word is used to store a 32-bit counter value. With this convention, the counter value can be accessed in the C program as:

```
unsigned counter = * (int *)
(&(settings[8]));
```

Listing 1—This code shows the PIC32's flash storage and access.

```
1. #pragma config CP = ON // prevent reading of code
2. #pragma config PWP = OFF // allow program flash write
3.
4. __attribute__((aligned(4096)))
5.     const unsigned char settings[4096] =
6.         {0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF};
7.
8. void IncCounter() {
9.     int *cp, v;
10.    unsigned int buf[3];
11.    memcpy(buf, settings, 12);
12.    buf[2] = buf[2] + 1;
13.    NVM_erasePage((void *) settings);
14.    NVM_writeWord((void *) settings, buf[0]);
15.    NVM_writeWord((void *) &(settings[4]), buf[1]);
16.    NVM_writeWord((void *) &(settings[8]), buf[2]);
17. }
```

The `IncCounter` function on Listing 1 lines 8–17 demonstrates how the counter can be incremented at runtime. First, the counter's value is copied in RAM and incremented. Next, the flash page holding the `settings` variable is erased and the

flash is updated with the new value from RAM. Listing 1 lines 13–16 are a critical section of the code. If there's a system power failure before the flash page is updated, the counter value and the secret value could be lost! This risk can

Listing 2—Challenge/response authentication on the PIC32 is shown.

```
18. int main() {
19.    unsigned char id[4];           // id of prover
20.    unsigned char challenge[20];  // challenge to prover
21.    unsigned char expect[20];     // expected response
22.    unsigned char response[20];   // expected response
23.
24.    UARTInit();
25.    LEDInit();
26.    LEDToggle();
27.
28.    while (1) {
29.        putsUART1("Enter ID in format [XXXXXXX]:\n");
30.        getID(id);
31.
32.        sha1_init();
33.        sha1_update(&(settings[8]), 4);
34.        sha1_final(challenge);
35.
36.        hmac(settings, challenge, id, expect);
37.
38.        IncCounter();
39.
40.        putsUART1("Challenge: ");
41.        putChallenge(challenge);
42.        WriteUART1('\n');
43.
44.        putsUART1("Response in format
[XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX]:\n");
45.
46.        getResponse(response);
47.        if (correctResponse(response, expect)) {
48.                                putsUART1("Authenticated\n");
49.                                LEDToggle();
50.        }
51.    }
52. }
```

be mitigated, for example, by allocating the secret and counter in a separate flash page and by using redundant copies of the counter in different flash pages.

CHALLENGE-RESPONSE WITH A CEREBOT MX7CK

Now that I've explained how to use nonvolatile memory to store the critical variables of the authentication protocol, I'll explain how the full protocol can be implemented. Listing 2 shows a sample design that connects to an external host through UART1. This design was written on a Digilent Cerebot MX7ck microcontroller development board. The complete source code for this design is available on *Circuit Cellar's* FTP site. The sample program in Listing 2 enables an external host to toggle an LED status, provided it can successfully authenticate itself to the PIC32.

The protocol starts with the host sending an identification code to the PIC32 (Listing 2 line 30). Any identification value can be used. It is the equivalent of saying, "I'm Bob" or "I'm Alice" as in Figure 2. Next, on Listing 2 lines 32–34, the PIC32 calculates the challenge for the host (i.e., the prover). The challenge is created by using SHA-1 to hash the counter value. The reason for using the counter digest rather than the counter itself, is to hide the counter's actual value from eavesdroppers. Indeed, even though revealing the counter value to an eavesdropper would not invalidate the authentication's result, it would still enable her to learn the number of authentication rounds completed. By hashing the counter value before using it as a challenge, the eavesdropper doesn't learn anything

from the challenge value. The challenge, thus, is a 20-byte SHA-1 digest created from a 32-bit counter.

Next, the PIC32 computes the expected response and the counter value is updated (see Listing 2 lines 36–38). The response is a 20-byte MAC. The challenge is released to the host only after the counter value update. This ensures communication faults do not result in the same challenge appearing multiple times. Finally, the response is compared to the expected value (see Listing 2 lines 46–50) and the LED is toggled when the authentication is successful.

OTHER FORMS OF EMBEDDED AUTHENTICATION

Implementing an HMAC-based challenge-response protocol on a microcontroller is the first step in embedded authentication. The scheme is easy to implement and it doesn't consume a large amount of resources or compute cycles. But, it comes with its own particular pitfalls.

The first pitfall is the need to have absolute trust in the microcontroller storing the secret. Even though the program flash can be protected from external read access, it's not possible to protect it from being scanned through malicious software.

The second pitfall is the use of flash memory to implement a counter. Since the number of flash write operations is limited, doing frequent authentications will wear out the microcontroller's nonvolatile memory.

The third pitfall involves scalability. Suppose you need to manage 4 million door locks, not just a single one. Can you use a different secret for each door lock? Not from a practical

**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



System on Module
New - SoM-3517

- TI ARM Cortex-A8 600 MHz Fanless Processor
- Up to 512 MB of DDR2 SDRAM
- Up to 1GB of NAND Flash
- Up to 2GB of eMMC Flash
- 2 High Speed USB 1.1/2.0 Host ports
- 1 High Speed USB 2.0 OTG port
- 4 Serial Ports, 2 I2C and 2 SPI ports
- Processor Bus Expansion
- 10/100 BaseT/Fast Ethernet
- CAN 2.0 B Controller
- Neon Vector Floating Point Unit
- 24-bit DSTN/TFT LCD Interface
- 2D/3D Accelerated Video w/ Resistive Touch
- Small, 200 pin SODIMM form factor (2.66 x 2.375")



The SoM-3517 uses the same small SODIMM form-factor utilized by other EMAC SoM modules and is the ideal processor engine for your next design. All of the ARM processor core is included on this tiny board including: Flash, Memory, Serial Ports, Ethernet, SPI, I2C, I2S Audio, CAN 2.0B, PWMs, Timer/Counters, A/D, Digital I/O lines, Video, Clock/Calendar, and more. The SoM-3517M additionally provides a math coprocessor, and 2D/3D accelerated video with image scaling/rotation. Like other modules in EMAC's SoM product line, the SoM-3517 is designed to plug into a custom or off-the-shelf carrier board containing all the connectors and any additional I/O components that may be required. The SoM approach provides the flexibility of a fully customized product at a greatly reduced cost. Contact EMAC for pricing & further information.

<http://www.emacinc.com/som/som3517.htm>

Since 1985
OVER
27
YEARS OF
SINGLE BOARD
SOLUTIONS

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL

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

perspective. If you cannot install unique secrets on every system, you will have to reuse them. For example, all door locks manufactured in a given month could use the same secret. Reusing a secret over multiple systems is a liability. If one single system is compromised, all similar ones will go down.

Each of these pitfalls can be addressed at the cost of increased system complexity. To address the first and second pitfall, a dedicated peripheral chip can be used to keep the secret safe from the software. Addressing the last pitfall is a bigger challenge. It requires authentication protocol modification. Each enhancement will be separately considered.

EMBEDDED AUTHENTICATION USING A DEDICATED PERIPHERAL

Figure 4 shows an authentication solution that uses a separate chip. I used an Atmel ATSHA204 authentication chip. Using an I²C interface, a controller can use this solution to implement all steps of the previously discussed challenge-response protocol. The ATSHA204 contains a read-protected nonvolatile memory to store secrets, a random number generator, and a SHA-1 algorithm hardware implementation.

The authentication protocol now works as follows. First, the PIC32 queries the ATSHA204 to generate a challenge, which is forwarded to the prover. The ATSHA204 keeps a local copy of the challenge. The prover then returns the response, which is forwarded by the PIC32 to the ATSHA204. Using the challenge it created earlier and the stored secret, the SHA204 verifies the response. The peripheral chip returns a single logical value to the PIC32: Yes (if the authentication passes) or No (if the authentication fails).

This configuration avoids the need to store secrets in the PIC32 nonvolatile memory, which addresses the earlier concern. Furthermore, the use of a random-number generator simplifies unique challenge generation. The downside of the setup, of course, is that you need an extra chip to support it.

EMBEDDED AUTHENTICATION USING PUBLIC-KEY CRYPTOGRAPHY

The challenge-response protocols used so far bring an important challenge to the designer. They require the distribution and management of a secret key pair for every system you wish to authenticate. Maintaining a global secret among a group of deployed systems is a significant risk. Public-key cryptography is a better solution for the scalability problem.

An example of a challenge/response chip that supports public-key cryptography is Infineon's ORIGA SLE95050 authentication chip. The chip is integrated into a microcontroller using a serial interface, similar to the ATSHA204. Every chip contains a secret key, which never leaves the package. The chip only releases a matching public key, which is not secret. A challenge-response protocol now works as follows: The challenger generates a nonce which is encrypted with the public key. It

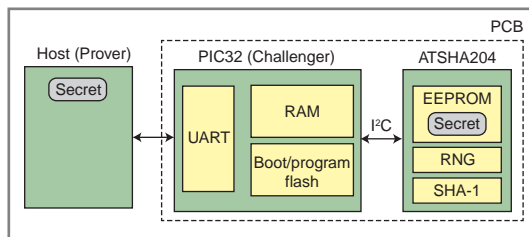


Figure 4—Embedded authentication using a dedicated peripheral is shown. The Atmel ATSHA204 authentication chip contains a nonvolatile memory, a random number generator, and a dedicated implementation of the SHA-1 algorithm. It connects to a microcontroller through an I²C interface. The microcontroller can authenticate a host without having to manage a stored secret.

then forwards the encrypted value to the SLE95050 authentication chip (in the prover role). This chip uses the internally stored secret key to decrypt the nonce and uses it as the response to the challenger. The system scalability lies in the fact that the challenger only needs to handle the public-key values, which are not secret. The secret-key values remain on-chip in the SLE95050.

AUTHENTICATION WITHOUT SECRETS?

In this article, I tried to expose some of the secrets in basic authentication protocols. I have a final observation. All the authentication systems I described require a secret value to embody the system's identity you wish to authenticate. This secret value needs to be stored in a nonvolatile memory and it's a liability for the design's security. A nice alternative that works well on people is using biometrics, which are person-unique features (e.g., fingerprints and iris patterns). In my next article, I will show how a "fingerprint" can be extracted from a field-programmable gate array (FPGA) and used to authenticate the FPGA. ■

Patrick Schaumont is an associate professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. He works with his students on research projects in embedded security, covering hardware, firmware, and software. You may reach him at schaum@vt.edu.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2013/270.

RESOURCES

BlueKrypt, "Cryptographic Key Length Recommendation," 2011, www.keylength.com.

C. Brocius, Trapped Orbit Research, "My Arduino Can Beat up Your Hotel Room Lock," 2012, <http://demoseen.com/bhtalk2.pdf>.

Microchip Technology, Inc, "MPLAB XC32 C/C++ Compiler User's Guide," 2012, www.microchip.com/mplabxc32guide.

———, PIC32 Peripheral Libraries for MPLAB C32 Compiler," 2007, <http://ww1.microchip.com/downloads/en/DeviceDoc/32bitPeripheralLibraryGuide.pdf>.

P. Schaumont, "One-Time Passwords from Your Watch," *Circuit Cellar* 262, 2012.

SOURCES

ATSHA204 Authentication chip

Atmel Corp. | www.atmel.com

Cerebot MX7ck Microcontroller development board

Digilent, Inc. | www.digilent.com

ORIGA SLE95050 Authentication chip

Infineon Technologies | www.infineon.com

PIC32 Microcontroller and MPLAB XC32 C Compiler

Microchip Technology, Inc. | www.microchip.com

Join the Elektor Community

Take out a Gold Membership now!



Your GOLD Membership contains:

- 8 Regular editions of Elektor magazine in print and digital
- 2 Jumbo editions of Elektor magazine in print and digital (January/February and July/August double issues)
- Elektor annual DVD-ROM
- A minimum of 10% DISCOUNT on all products in Elektor.STORE
- Direct access to Elektor.LABS
- Direct access to Elektor.MAGAZINE
- Elektor.POST sent to your email account (incl. 25 extra projects per year)
- An Elektor Binder to store these 25 extra projects (on request)
- Exclusive GOLD Membership card

ALSO AVAILABLE:

The all-paperless GREEN Membership, which delivers all products and services, including Elektor.MAGAZINE, online only.



Take out your Membership now at www.elektor.com/member



Web-Based Tools for Energy Efficiency

A family of Wi-Fi devices enables you to use Web-based tools to control and monitor your electrical power outlets (and the devices they control). This application programming interface (API) can combine devices from many manufacturers into one cohesive application.

Circuit Cellar founder and former editorial director Steve Ciarcia always used to say, "Don't hide it in a patent, give it to the world!" Many companies are now beginning to adopt this mantra.

I circled the date on my calendar: August 20, 2012. This commemorates the day I found the first mass-produced product that enables you to control and monitor anything that comes with an AC plug. In addition to their On and Off buttons, the products let you know exactly how much current they are consuming on command.

Visible Energy, a Silicon Valley start-up, has hatched a family of Wi-Fi-enabled smart electrical power outlets. These applications for Apple's vastly popular "i" product line provide users with Web-based tools essential for energy management. Visible Energy believes in sharing. Application programming interfaces (APIs) are available to communicate with all its hardware, currently the Monostrip, a two-outlet (one switched) hub, and the UFO Power Center, a four-outlet (individually switched) hub (see [Photo 1](#)). First, a quick look at the hardware.

MONOSTRIP & UFO

Photo 1 shows a size comparison between the Monostrip and UFO

devices. The Monostrip model is small and provides two outlets, one passthrough and one controlled (see Photo 1a). The UFO is a fashion statement (see Photo 1b). Its large body is meant to conceal miles of ugly wiring. Beneath its rubber dome are four outlets arranged around a center post. The cavernous interior will hold all the extra feet of



Photo 1—Visible Energy's API includes the Monostrip, a two-outlet (one switched) hub (a) and the UFO Power Center, a four-outlet (individually switched) hub (b).

Function	/x (option)	Command	/?Query (option)	Response	
Device status	0 1 2 3 4 5 6 7	status.xml js		mac = x, utc = x, identity release = x, model = string, name = string, state = online offline	
				repeated for each socket within the device	socket position = x, name = string, state = on off, watts = x, peak = x, role type = neutral master slave, timers total = x, days = -----, time = hh:mm, value = ON OFF, oneshot-timers total = x, utc = x, value = ON OFF
Switch	0 1 2 3 4 5 6 7	switch.xml js	value = on off	mac = x, state = online offline	
				repeated for each socket within the device	socket position = x, state = on off, watts = x, peak = x
Timers	0 1 2 3 4 5 6 7	timer.xml js	time = hh:mm days = ----- value = on off reset erase	mac = x, socket position = x, result = x, timers total = x	
				repeated for each timers total for a socket within the device	days = -----, time = utc, state = on off
One-shot timers	0 1 2 3 4 5 6 7	oneshot.xml js	seconds = utc value = on off reset erase	mac = x, socket position = x, result = x, one-shot timers, total = x	
				repeated for each one-shot timers total for a socket within the device	utc = x, state = on off
Timer overrides	0 1 2 3 4 5 6 7	override.xml js	seconds = utc count = x reset erase	mac = x, result = x, match = socket all, socket = x, type = expires count, value = utc x	
Socket roles	0 1 2 3 4 5 6 7	role.xml js	type = neutral master slave threshold = x	mac = x	
				repeated for each socket within the device	socket position = x, type = neutral master slave

Table 1—These are the functions documented in the Automation API. The option “/x” is used to address an individual device socket; otherwise, you’re talking to all sockets. The option “/?Query” is used to write information/configuration values to the device, otherwise you are just asking for the command’s status.

power cables, keeping your desk (or floor) from looking like the electric company’s power-generation complex. Both devices include Wi-Fi servers that can be added to your home network. The Monostrip and the UFO have 15-A ratings per device. Local socket control is not available, but there is a lot more to these devices than just real-time monitoring and control.

Each (controlled) socket can (autonomously) carry out an extensive automated weekly event schedule based on time of day and one-time events. A unique concept is the master/slave relationship sockets within a UFO device can acquire. Slave sockets can automatically turn on/off based on the master socket’s current (amperage) level (standby versus on). The features will be detailed when you study the API, which I’ll present later.

A CLOUDED SKY

For those who are not inclined to venture into the software realm, you can register your devices for free using Visible Energy’s cloud. From the cloud, you can monitor each device’s name (and each socket) and see a graphic representation of past (and present) energy usage. You can control sockets and set up daily scheduling for each. A great feature of this free service is the ability to receive an e-mail or text message based on any socket’s change in consumption. For instance, if my TV was plugged into a powered socket and the current increased beyond the vampire level (i.e., TV off consumption condition), I could receive a notification that someone had turned it on.

When I clicked my device’s information button, one of the items listed included the device’s present software revision. There is an option to check for revision updates. A new revision

was available, which was automatically handled upon confirmation. In just minutes, I had updated each device.

This service may be all many users will ever need. However, many of us desire a higher power, so to speak. We may feel we don’t want our personal network open to a cloud service. We may be looking for some autonomy. We may have an idea for some specific use. Or, we may just be curious. Whatever our motives, I thank Visible Energy for enabling us to take full advantage of its powerful products.

API

An API is a set of routines, protocols, and tools for building software applications that can communicate with one another. In this case, this occurs between your application and Visible Energy products using Smart Outlet technologies that reside on your local network. Since each device has an embedded Hyper Text Transfer Protocol (HTTP) server, communication follows the HTTP. All functions are HTTP GET operations, returning data in either Extensible Markup Language (XML) language (free format) or JavaScript Object Notation (JSON). While both contain the same information, XML wraps and presents the data a bit differently. If you are viewing the result in a browser, “.xml” (XML) presents a hierarchal view of the data while “.js” (JavaScript) is a simple string that can be difficult to view, but easier for data extraction.

All functions use the form “IP/optional folder/command? optional query string” (i.e., <http://192.168.1.141/config/name.js?name=Living Room>”), which means set the name of the device @192.168.1.141 to “Living Room.” This might return: { “strip”: { “mac”: “071330”, “result”: “0x00”, “name”: “Mono-071330” } }, which, in JSON, shows the object “strip” is

Function	/x (option)	Command	/?Query (option)	Response	
Energy	0 1 2 3 4 5 6 7	energy.xml js	period = hour day week month from = utc to = utc subperiod = xh xd xw xm	mac = x, start = utc, length = seconds	
				repeated for each socket within the device	socket position = x energy = 4 data points (hour) or 24 data points (day) or 7 data points (week) or 31 data points (month) or x data points (h d w m)
Meter	0 1 2 3 4 5 6 7	meter.xml js		mac = x, start = utc, length = seconds	
				repeated for each socket within the device	socket position = x, meter = x
Log	0 1 2 3 4 5 6 7	log.xml js	from = utc to = utc	mac = x, start = utc, records = x, shortest = seconds, longest = seconds	
				repeated for each sample for each socket within the device	sample = socket position = x, W/hr = x

Table 2—These are the energy-over-time functions documented in the Energy API. Once again, the option “/x” is used to address an individual socket within a device; otherwise, you’re talking to all sockets. The option “?Query” is used to write information/configuration values to the device, otherwise you are just asking for the command’s status.

a collection of three key:value pairs where, “mac” equals the “071330” string, “result” equals the “0x00” string, and “name” equals the “Mono-071330” string.

At this time, seven API documents are available including: “Introduction to the HTTP API v2.0,” “Discovery Protocol,” “Network Configuration,” “Device Configuration API,” “Power API,” “Energy API,” and “Automation API.” In this article, I’ll discuss the last two documents. A browser (and your local network configuration parameters) is all that is needed for device setup. Once they are a part of your network, the various other documents discuss the API you could use to do this via an application, but this project simplifies the application tasks necessary for some control and monitoring.

AUTOMATION

The Automation API includes six functions: “Device Status,” “Switch,” “Timers,” “One-Shot Timers,” “Timer Overrides,” and “Socket Roles.” Note: The data within the Power API is included in the “Device Status” information, so it’s excluded from this discussion. [Table 1](#) lists the API functions and parameters. Notice that while the API supports up to eight sockets per device, the Monostrip and the UFO offer one and four (switched) sockets, respectively.

As [Table 1](#) shows, the status command returns the largest amount of data. Most functions operate on a per-socket and per-device basis. By adding the optional “/” (socket number) before a command, the function returns a result from a single socket. Otherwise, results are returned from every socket within the device. The switch function provides access to a socket’s mechanical relay. Each (switched) socket uses a latching relay that requires no holding current. This keeps the device supply current to a minimum no matter what the state.

Timers, one-shot timers, and overrides provide devices with minds of their own. You use the timer function to schedule socket switching based on time of day and day of week. The schedule can be different for every day of the week. The timer function includes scheduling on/off tasks based on real time in hh:mm and the days affected.

One-time events and timer overrides can be scheduled in second increments based on Coordinated Universal Time (UTC). UTC is the primary time standard by which the world regulates

clocks and time. It is based on the number of seconds since January 1, 1970. To use this, you invoke the real-time clock (RTC) function (IP/config/rtc.js), which returns UTC in seconds. You then use this number plus the number of seconds delay you want in the One-Shot or Override command. When the UTC reaches this value, the scheduled operation occurs and the scheduled task disappears.

Socket Roles is a unique function that enables one socket’s current consumption level to determine other sockets’ states. When a socket is configured as master, it also receives a threshold value. Other sockets in the device that are configured as slaves will be turned on when the master’s current consumption rises above the threshold. They are turned off when the level falls below the threshold.

ENERGY

The Energy API includes three functions: Energy, Meter, and Log. [Table 2](#) lists the API functions and parameters. While the API supports up to eight sockets per device, the present products, UFO and Mono, offer four and one (switched) socket, respectively. While immediate energy (and peak) data are part of the Status function described earlier, longer duration data is available from the Energy API.

The Energy function enables you to request energy (i.e., watt-hours) for the last hour (in four 15-min. periods), the last day (in 24 1-h periods), the last week (in seven one-day periods), and the last month (in up to 31 one-day periods). You may also request energy using the from/to/subperiod query to get all data between the two UTC times in subperiod chunks. The subperiod refers to a chunk’s length in seconds, unless the value is followed by m/h/d (minutes/hours/days).

Each device also keeps a running tally of the energy (i.e., watt-hours) used by the device over its lifetime. The Meter command returns this along with when the device was manufactured (i.e., UTC) and the length of seconds it has been in service. You also have access to the entire energy sample record. Unlike the Energy function, this data is held within a serial memory device and will take a bit longer to retrieve, especially as it grows in length. Its format is the same as the Energy function using the from/to query. Each sample returned is in watt/hour for each device socket.

ASSEMBLY LANGUAGE ESSENTIALS

Circuit Cellar's first book, ***Assembly Language Essentials***, is a matter-of-fact guide to Assembly that will introduce you to the most fundamental programming language of a processor.



Author Larry Cicchinelli provides readers with:

- An introduction to Assembly language & its functionality
- Essential terminology pertaining to higher-level programming languages & computer architecture
- Important algorithms that may be built into high-level languages — multiplication, division, and polynomial evaluation
- Overview of Interrupt Service Routines
- Free, downloadable Assembler program ...
and more!

On Sale NOW!

\$47.50

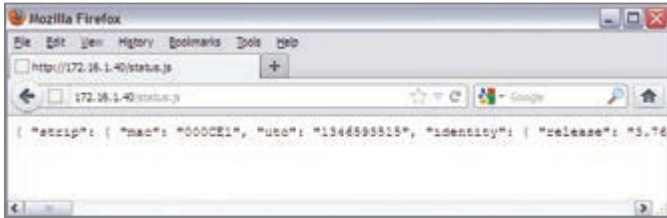


Photo 2—The browser displays a returned .js string without formatting. The string is one continuous line that runs off the screen.

MAKING SENSE OF THE JSON DATA

You can use a browser to directly look at any of these devices by entering its IP address followed by the API function. When you use the .js file tag, you will see something like what is shown in [Photo 2](#). Notice how the data string displayed runs off the screen. It's all there. It's just not formatted by the browser. Try this again with the .xml file tag, and you should see something similar to what is shown in [Photo 3](#). This time, the browser knows how to interpret the XML format's structure. While it's not very exciting, at least you can see all of it and make some sense out of what is presented there. Enter the Visible Energy cloud. Once registered, you can receive a nice graphical display of your devices from any browser or mobile device (the "i" products already have apps available, with Android apps not far behind). However, a browser—whether a desktop, a laptop, or another mobile device—can also be used.

I've used Shoptalk Systems's Run BASIC web programmer for many projects. I will add this project to my server so you can visit it. I use a dynamic DNS provider to direct my published domain name to the correct dynamic IP. I did this so if my IP address changed after I published it, the IP would still be connected to me. My domain name, which is hosted by No-IP.com, is jlbachiochi.myvnc.com (no www). If you want to know more about dynamic DNS, just replace "jlbachiochi" with "www."

It may be a good idea to show some code that picks apart a JSON response to a HTTP Get statement my application will use to gather data from a device. Basically, JSON is a list of objects that are presented as name/value pairs. Each object uses the form { (left brace) "object name" : (colon) <object value> } (right brace). Note: While the object name is always a string, the object value may be a string, a number, an object, an array, true/false, or null. For the most part, you will see a string or number, unless the object is a list of additional objects. To determine this, look at the character following the colon to ascertain what the object value contains. Note: White space (space character) should be neglected. If the significant character following a colon is a " (quote), the following characters are a string value. If it is a digit (0–9), then the following characters are numeric. An open brace indicates a list of objects separated by a , (comma) follows. A [(square bracket) indicates an array of objects separated by a , (comma) follows.

If you execute a IP/status.js command and capture the returned JSON string in response\$, you should be able to make sense of the string using the previously described rules (see [Photo 2](#) and [Photo 3](#)). Starting on the left, you will find an open

brace indicating an object "strip" followed by another open brace. So "strip" will become the beginning of a list of objects. The first object is "strip mac" and the " (quote) after the colon indicates that a string value follows, in this case "strip mac" = "000CE1." The comma indicates there is another object. The next object is "strip utc" = "1346593366." Another comma, another object. However, this object "strip identity" has an open brace following it, so its values will be a list of objects. Let's skip over this list for now by finding the matching close brace. The fourth object found is "strip status." Following its colon, there is another open brace indicating the object "strip status" is again an object list. If you carefully find the closing brace for this list, you will be at the end of the string response\$. At this point, there are two complete items: strip_mac\$="000CE1" and strip_utc\$=1346593366." If you continue with this format, you find out more about "strip identity." This object contains a list of three objects that can be equated to strip_identity_release\$="3.76", strip_identity_model\$="UFO", strip_identity_name\$="UFO-000CE1." The last object, "strip status," has two objects. The first equates to strip_status_state\$="ONLINE," but the second, "strip status socket," is an array of lists (and items in the list are also arrays!).

Before getting into this last data, I want to explain why you must pay attention to the object grouping. While it has not yet been seen, objects could have the same object name. This would become very confusing and lead to a search of response\$ that may return the wrong data unless you pay attention to the

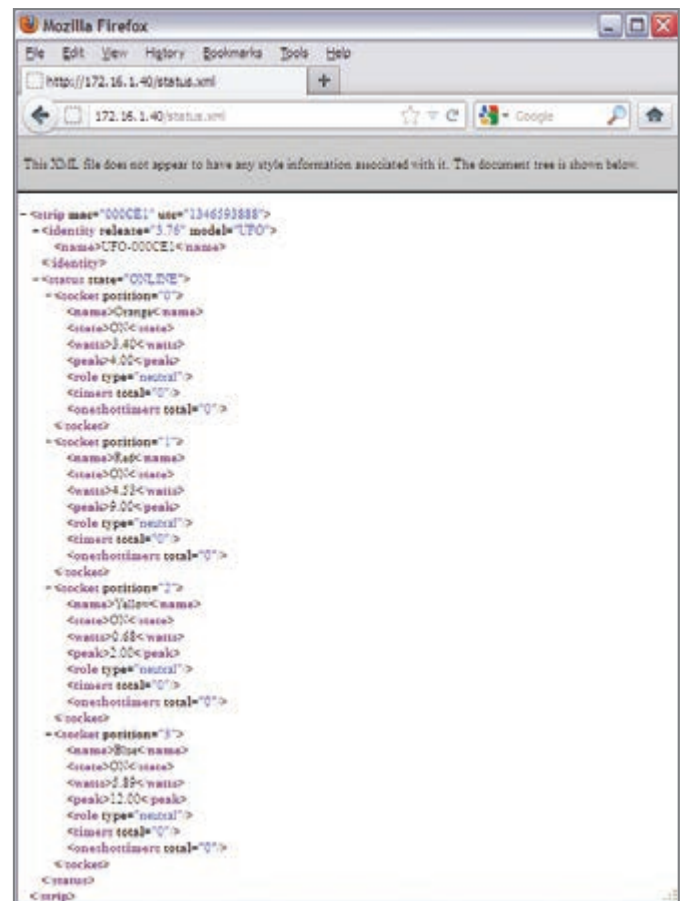


Photo 3—The browser displays a returned .xml string formatted for the screen.

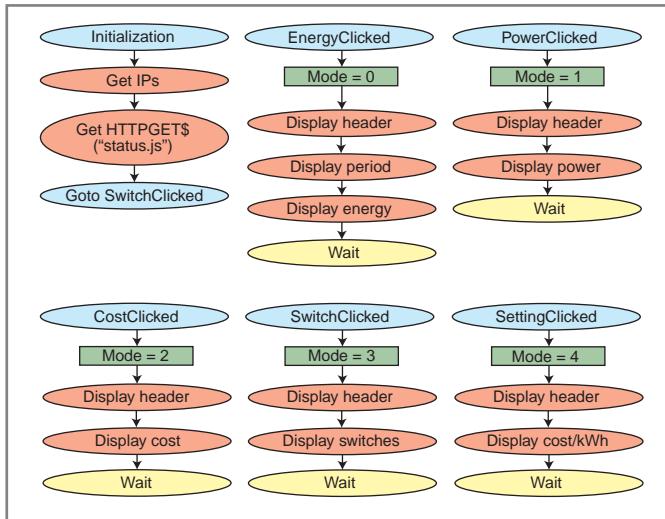


Figure 1—This application uses the HTTPGET\$() statement to retrieve a JSON string containing all the status objects from the device residing at the requested IP address. Based on the selected mode, one of five different screens are displayed. If you request a new device or additional API functions are needed, a new request is generated.

grouping. For instance, the object name "name" can be found in the identity group (as seen) and in the socket group (coming up). I've defined objects by using their complete path so there is no ambiguity.

The documentation shows that a device can have up to eight sockets, so an array can be used to hold identical socket objectlists. The objects found in the socket list are:strip_status_socket_position, strip_status_socket_name, strip_status_socket_state, strip_status_socket_watts, strip_status_socket_peak, strip_status_socket_role (a one-object list), strip_status_socket_total, strip_status_socket_timers (an array of objects, strip_status_socket_oneshottotal, and strip_status_socket_oneshottimers (an array of objects). A socket's array of the first socket might consist of the following objectnames and objectvalues:

```

Socket$(0,0,0)= "9" (number of objects)
Socket$(0,0,1)= ""
Socket$(0,1,0)= "position"
Socket$(0,1,1)= "0"
Socket$(0,2,0)= "name"
Socket$(0,2,1)= "Orange"
Socket$(0,3,0)= "state"
Socket$(0,3,1)= "ON"
Socket$(0,4,0)= "watts"
Socket$(0,4,1)= "3.40"
Socket$(0,5,0)= "peak"
  
```

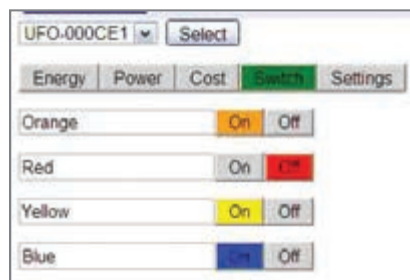


Photo 4—The consistent "header" at the top of all display screens enables you to select devices and display screens. The Switch screen presents you with the present state of every socket with the selected device. It can turn them on/off at will.

```

Socket$(0,5,1)= "4.00"
Socket$(0,6,0)= "role"
Socket$(0,6,1)= "role" (separate list)
Socket$(0,7,0)= "total"
Socket$(0,7,1)= "0"
Socket$(0,8,0)= "timers"
Socket$(0,8,1)= "timers" (separate list)
Socket$(0,9,0)= "oneshotttotal"
Socket$(0,9,1)= "0"
Socket$(0,10,0)= "oneshottimers"
Socket$(0,10,1)= "oneshottimers" (separate list)
  
```

You may have noticed that the <objectname> and <objectvalue> for "role," "timers," and "oneshottimers" are the same. I did this to keep the array sizes under control. These list and array objects can be found elsewhere.

```

role$(0,0)= "1" (number of objects)
role$(0,1)= ""
role$(1,0)= "type"
role$(1,1)= "neutral"
  
```

```

timers$(0,0,0)= "3" (number of objects in list)
timers$(0,0,1)= "0" (number of events from socket$(0,7,1))
(this array will contain any scheduled events for eg.)
timers$(0,1,0)= "days"
timers$(0,1,1)= "smtwtfs"
timers$(0,2,0)= "time"
timers$(0,2,1)= "00:00"
timers$(0,3,0)= "value"
timers$(0,3,1)= "ON"
  
```

```

oneshottimers$(0,0,0)= "2" (number of objects in list)
oneshottimers$(0,0,1)= "0" (number of events from socket$(0,9,1))
(this array will contain any scheduled events for eg.)
oneshottimers$(0,1,0)= "utc"
oneshottimers$(0,1,1)= "1346596966"
oneshottimers$(0,2,0)= "value"
oneshottimers$(0,2,1)= "ON"
  
```

Be careful with the array dimensioning ensuring you leave sufficient room. For instance, the role array will contain two objects when the type is "master." Depending on your event schedule, timers\$ and oneshottimers\$ may contain numerous entries.

SERVER APPLICATION

I looked at some iPhone app screen shots on Visible Energy's website. This project attempts to display similar information to any web browser. Figure 1 is a block diagram of the application written using these APIs. The application begins by reading in a text file containing a list of all the device IPs. This enables the IPs to change without having to hard code them into the application. The application requests each device's status and jumps to the Switch screen. All screens begin with the same header and a number of buttons you can click to change devices (i.e., IPs) or modes (i.e., display screens). Following this header,

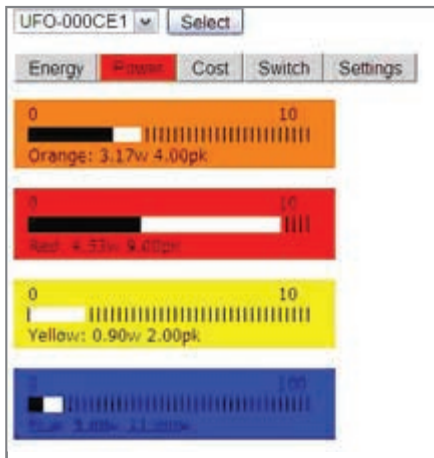


Photo 5—The Power screen presents you with the real-time power usage in watts along with the peak power recorded. On each meter, the peak is shown in white, with the real-time power shown in black. The scale is adjusted to fit the data.

information is provided pertaining to the specific mode.

The Switch screen shows each socket's status within the selected device. Clicking a socket's On/Off buttons sends a request to the device to change the socket's state. The status is again requested and the refreshed screen shows the updated status (see [Photo 4](#)).

The Power screen displays instantaneous power (i.e., real time) usage. A thermometer-style meter shows the instantaneous (in black) and peak (in white) power levels of each selected device's socket. Background colors match a UFO device's physical socket colors: orange (Socket 0), red (Socket 1), yellow (Socket 2), and blue (Socket 3). Each socket's object name is also displayed. In



Photo 7—Once the final cost per kilowatt hour is known, the real-time costs (cents/hour) can be calculated based on this cost multiplied by the instantaneous usage. Background colors are selected based on socket (i.e., position number).

this case, the default names (i.e., socket colors) have not been changed (see [Photo 5](#)).

[Photo 6](#) shows that I am implementing a single setting on the Settings screen; the cost of electricity in cents/kWh as reflected on your utility bill. You can change the amount up to anything less than \$1. Note: Look closely at your bill. You will find it is broken down into two parts: electricity supply service (which you can now choose from whom to buy power) and a delivery service (which will always come from your utility company.) The first is the cost of generating electricity (e.g., solar, wind, oil, gas, hydro, etc.) and the second is the cost of delivering that power (e.g., smart grid infrastructure). On my bill, both are around \$0.08 for a total of \$0.16. I used \$160 (i.e., approximately 1,000 kWh \times \$0.16).

The Cost screen uses the cost/kWh from the Settings screen and multiplies it by each socket's instantaneous power usage to come up with real-time costs (cents/hour) for each socket within a selected device (see [Photo 7](#)).

Finally, the Energy screen displays energy usage over time (see [Photo 8](#)). You can view energy over the last hour, day, week, or month. Each selection must use the energy API to make a data request. On this screen, a second set of buttons enables you to choose the display period for each socket within the selected device. The data retrieved is placed onto a timeline of the period chosen, with a vertical scale based on watt hours used during that period. The hour selection has four 15-min. averages, the day selection has 24 1-h averages, the week selection has seven daily averages, and the month selection has 31 daily averages.

Run BASIC is an object-oriented language. This means code is executed on a trigger from some object (e.g., a button click or a timer). In this application, execution is paused at the end of every display screen (see Figure 1). When you click a button, a jump is made to some code that will perform the required function (e.g., changing screen modes or selecting devices). While I could have designed special icons to make my displays look better, this application is intended to show how to use the APIs available for the Internet devices made

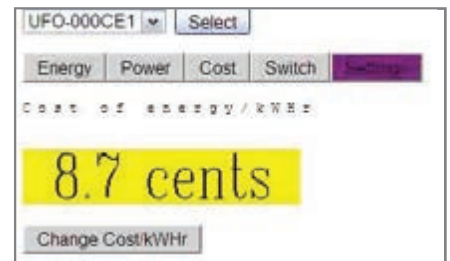


Photo 6—Your electricity bill shows a breakdown of all costs that can be combined to provide you a total cost per kilowatt hour. You may have to adjust this somewhat to cover any flat-fee charges.

by Visible Energy. For more information, visit my server site at jlbachiochi.myvnc.com/ and click on this issue, ftb270.

MY CIRCLED DATE

You don't have to agree with me. You don't have to have the date circled on your calendar. I have. I believe this has a high level of importance, not like the introduction of the first automobile, electric light bulb, and the cell phone, but availability to the masses. I look forward to the day when these devices are part of every switch, outlet, and breaker installed in new buildings.

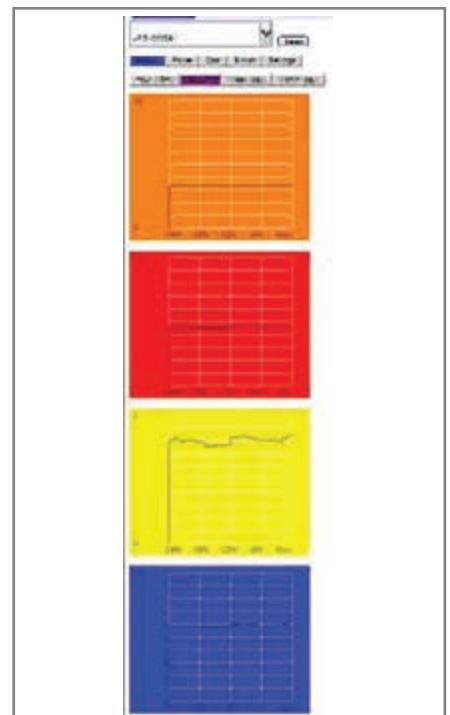


Photo 8—So far, all the data has come from the status.js function. To retrieve past (i.e., recorded) data, use the energy.js function. This can be requested for the last hour, day, week, or month. The displayed graphs help you get a feel for the energy used over time (i.e., the on versus off ratio).

Open APIs such as those offered by Visible Energy will enable us to combine devices from many manufacturers in one cohesive application that will give us complete control over our personal environments (i.e., our homes). Would you like to save money by washing/drying your clothes at night, adjust your heater/air conditioner while you're at work, illuminate your entrance when you come or go in the dark, animate your lights (so it looks like you're at home), or view live video in and around your home? This is the perfect time to begin a new revolution in automation. Thank you, Visible Energy. ☺

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

Jeff Bachiochi's server site, jlbachiochi.myvnc.com.

No-IP.com, www.no-ip.com or www.myvnc.com.

SOURCES

Run BASIC web programmer and Liberty BASIC programming software for Windows
Shoptalk Systems | www.libertybasic.com

MonoStrip and UFO Power Center
Visible Energy, Inc. | www.visibleenergy.com

NEED-TO-KNOW INFO

Knowledge is power. In the computer applications industry, informed engineers and programmers don't just survive, they *thrive* and *excel*. For more need-to-know information about some of the topics covered in this article, the *Circuit Cellar* editorial staff recommends the following content:

Wireless Data Delivery

by Jeff Bachiochi

Circuit Cellar 264, 2012

You can automate a system's data collection process to eliminate the need for manual data entry. This article describes how to use open wireless technology to deliver data wirelessly between a project and an application written to gather data from it. Topics: Open Wireless, Data

Get Your Head Out of the Cloud

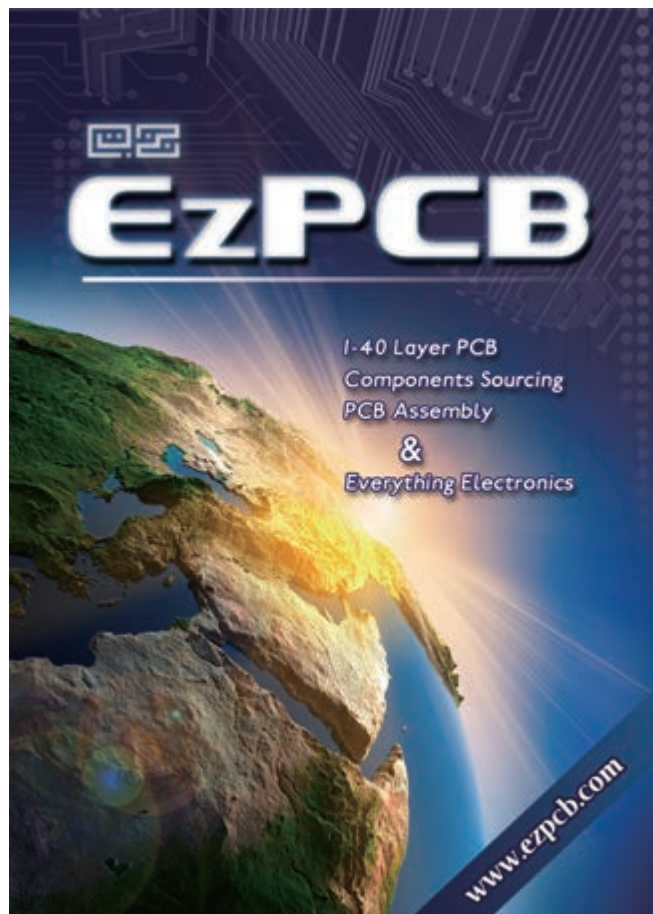
An API for Communication with a Wi-Fi Thermostat

by Jeff Bachiochi

Circuit Cellar 255, 2011

Here you learn how to build applications to communicate with a Wi-Fi Utility Smart Network Access Port (USNAP) thermostat using an application programming interface (API). Topics: Wi-Fi, API, USNAP

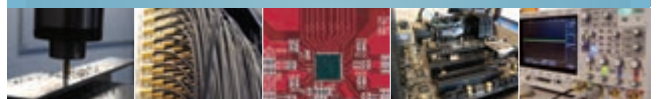
Go to *Circuit Cellar's* webshop to find these articles and more: www.cc-webshop.com



GOD-LIKE ENGINEERS & PROGRAMMERS:

BUILD A 23RD CENTURY MEDICAL DEVICE

Butterfly Network is a fully-funded startup backed by entrepreneurs who have already shaped our future. Be at the intersection of Computer Science, Electrical Engineering and Medicine as we develop advanced diagnosis and treatment technologies that will transform medicine.



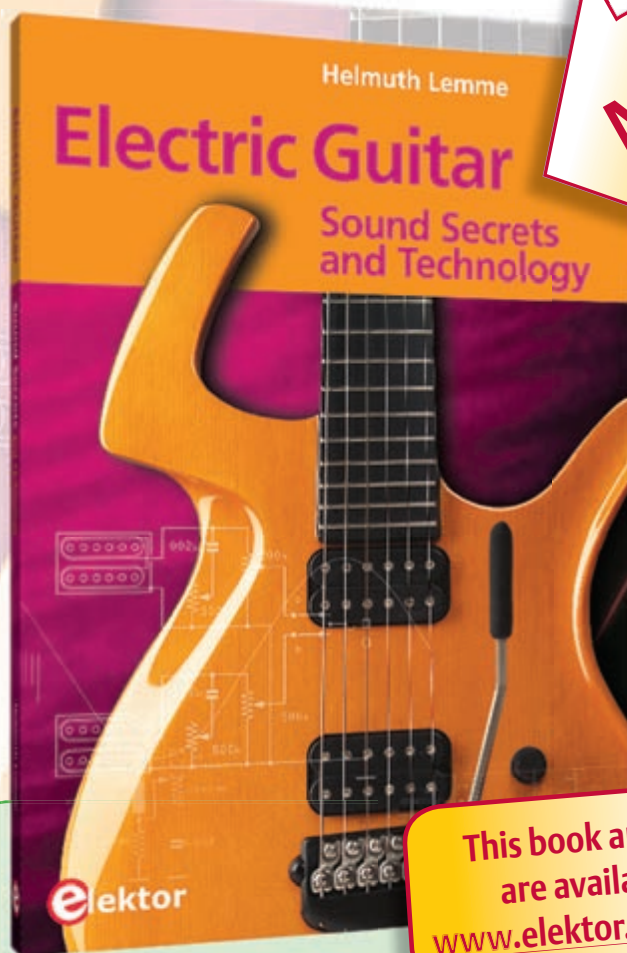
WANT TO CHANGE THE WORLD? JOIN US.

- FPGA and Embedded Design Engineer
- 3D Visualization Computer Scientist
- Device Driver and Software Engineer
- Signal and Image Processing Specialist
- Electrical Engineer or Applied Physicist

info@butterflynetinc.com | www.butterflynetinc.com

Elektor.STORE

The world of electronics
at your fingertips!



NEW!

This book and more
are available at
www.elektor.com/books

Books

Sound Secrets and Technology Electric Guitar

The underlying sound of electric lead and bass guitars is determined largely by their electrical components. But, how do they actually work? Almost no one is able to explain this to the true musician with no technical background. This book answers many questions simply, in an easily-understandable manner. For the interested musician (and others), this book unveils, in a simple and well-grounded way, what have, until now, been regarded as manufacturer secrets. The examination explores deep within the guitar, including pickups and electrical environment, so that guitar electronics are no longer considered highly secret. With a few deft interventions, many instruments can be rendered more versatile and made to sound a lot better – in the most cost-effective manner.

287 pages • ISBN 978-1-907920-13-4 • \$47.60

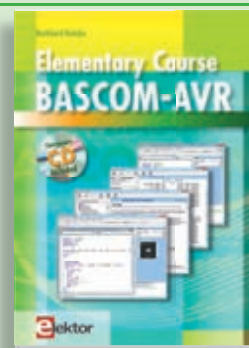


Bestseller!

LabWorX 2: Straight from the Lab to your Brain Mastering Surface Mount Technology

This book takes you on a crash course in techniques, tips and knowhow to successfully introduce Surface Mount Technology in your workflow. Besides explaining methodology and equipment, attention is given to parts technology and soldering technique. Several projects introduce you step by step to handling surface mounted parts and the required technique to successfully build SMT assemblies. Many practical tips and tricks are disclosed that bring surface mounted technology into everyone's reach without breaking the bank.

282 pages • ISBN 978-1-907920-12-7 • \$47.60



Free Software CD-ROM included

Elementary Course BASCOM-AVR

The Atmel AVR family of microcontrollers are extremely versatile and widely used. Elektor magazine already produced a wealth of special applications and circuit boards based on ATmega and ATtiny controllers. The majority of these projects perform a particular function. In this book however the programming of these controllers is the foremost concern. Using lots of practical examples we show how, using BASCOM, you can quickly get your own design ideas up and running in silicon.

224 pages • ISBN 978-1-907920-11-0 • \$56.40



Presented by Menno van der Veen

DVD Masterclass Modern Valve Electronics

This filmed seminar (165 mins. video and more) starts with a short discussion of the classic approach using valve load line graphs, followed by current sources and current foldback techniques. Next, the negative effect of cathode electrolytics is covered as well as reducing supply voltage interference. With the help of state of the art measurement techniques the (in)correctness of feedback is proven, while also clarifying what's happening deep within the core of the output transformer.

ISBN 978-1-907920-10-3 • \$40.20



Bestseller!

More than 75,000 components

CD Elektor's Components Database 7

This CD-ROM gives you easy access to design data for over 11,100 ICs, 37,000 transistors, FETs, thyristors and triacs, 25,100 diodes and 2,000 optocouplers. The program package consists of eight databanks covering ICs, transistors, diodes and optocouplers. A further eleven applications cover the calculation of, for example, zener diode series resistors, voltage regulators, voltage dividers and AMV's. A colour band decoder is included for determining resistor and inductor values. All databank applications are fully interactive, allowing the user to add, edit and complete component data. This CD-ROM is a must-have for all electronics enthusiasts!

ISBN 978-90-5381-298-3 • \$40.20



Bestseller!

Embedded Linux Made Easy

Today Linux can be found running on all sorts of devices, even coffee machines. Many electronics enthusiasts will be keen to use Linux as the basis of a new microcontroller project, but the apparent complexity of the operating system and the high price of development boards has been a hurdle. Here Elektor solves both these problems, with a beginners' course accompanied by a compact and inexpensive populated and tested circuit board. This board includes everything necessary for a modern embedded project: a USB interface, an SD card connection and various other expansion options. It is also easy to hook the board up to an Ethernet network.

Populated and tested Elektor Linux Board

Art.# 120026-91 • \$93.30

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

www.elektor.com/store



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

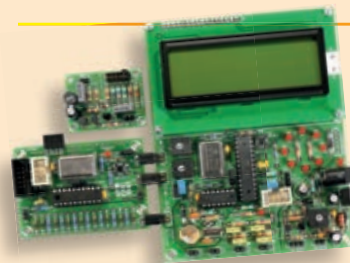


USB Isolator

If your USB device ever suffers from noise caused by an earth loop or if you want to protect your PC against external voltages then you need a USB isolator. The circuit described in Elektor's October 2012 edition offers an optimal electrical isolation of both the data lines as well as the supply lines between the PC and the USB device.

Populated and tested Board

Art.# 120291-91 • \$101.40



AVR Software Defined Radio

This package consists of the three boards associated with the AVR Software Defined Radio articles series in Elektor, which is built around practical experiments. The first board, which includes an ATtiny2313, a 20 MHz oscillator and an R-2R DAC, will be used to make a signal generator. The second board will fish signals out of the ether. It contains all the hardware needed to make a digital software-defined radio (SDR), with an RS-232 interface, an LCD panel, and a 20 MHz VCXO (voltage-controlled crystal oscillator), which can be locked to a reference signal. The third board provides an active ferrite antenna. This bundle also includes the assembled and tested FT232R USB/Serial Bridge/BOB PCB!

*Signal Generator + Universal Receiver +
Active Antenna: PCBs and all components +
USB-FT232R breakout-board*

Art.# 100182-72 • \$133.00

FROM THE ARCHIVES

In celebration of *Circuit Cellar's* 25th year, we're running an article from the archives each month that exemplifies something special about this magazine, its contributors, and its readers. We hope you'll enjoy reading (or perhaps rereading) these innovative projects as much as we did preparing them. This month, we feature a 1998 article (Issue #95) that *Circuit Cellar's* founder, Steve Ciarcia, wrote with columnist Jeff Bachiochi. The interesting article highlights the collaborative nature of *Circuit Cellar* readers and contributors and the staff's well-known penchant for surveillance and home-control projects.

by Steve Ciarcia and Jeff Bachiochi
Circuit Cellar 95, June 1998

Gotcha!

Alarming the Alarm System

Alarm companies fall a little short if what you want is entry and exit printouts. Sure, they'll do them. But only at \$20 a pop—or the cost of a new system. So, what do you do? If you're Steve and Jeff, you add a little electronic sleuthing to the system.

I was just about ready to pack it in for the night when Jeannette called down the cellar stairs.

"Steve, the alarm service is on the phone. The alarm at the office just went off! They called the police. Shall I say you're on the way to meet them?"

As I grabbed my coat and keys, I cast a quick glance back at Jeannette. Her expression said far more than any verbal exchange.

The gist of it was that if anyone was going to play hero tonight, it wasn't going to be her. She's happy to run a business with me, but gunslinger is definitely not in her job description.

There was a time when sharing the "business experience" might have prevailed, but after a real break-in at our office when the police actually dragged a burglar out in handcuffs, she decided this was one event she'd rather stay away from. Besides, anyone stupid enough to break into a building attached to a courthouse and surrounded by a half-dozen TV cameras probably isn't bright enough to listen to reason anyway.

As I ran for the car, I heard her yell, "Be careful...! Call me...!"

Like most businesses, we have a commercial alarm system. The reason isn't as much to deter crime as it is to qualify for discount on insurance rates. That's the good news.

The bad news is that, because I live closest to the office, I'm first on the call list when the alarm goes off. I get to greet the police and walk around a building with a lot of dark hiding places.

There really aren't a lot of options. If you want the police to treat the call seriously, you better meet them there. And, you also have to watch the false alarms.

Everyone had a sense of humor when someone set the alarm while Ken was still working on the third floor. Since then, the last one out is supposed to page the building or check the parking lot. There has to be a bit of seriousness. After all, the police did nab somebody that one time.

As I pulled into the parking lot, the lights from state and local police cruisers greeted me. All I could think was, please, let them find Jeffery Dahmer or someone, not another false alarm.

After the appropriate introductions, we trooped into the building—they with their guns drawn and flashlights blazing. I don't know why they didn't just turn on the lights, but they preferred to search each room in the dark.

I still don't understand the tactic. Maybe they presumed someone this dumb would invariably use tracer ammo or something else that's easy to see in the dark. Needless to say, I waited until the lights were on before roaming anywhere unescorted.

The results of the search were less than spectacular and somewhat embarrassing. Apparently, workmen had left an outside door to the furnace room unlocked and nobody checked it.

Besides the lock, the door needs a 3½" thick wooden bar across the inside to keep it from opening. Of course, when someone leaves the door unlocked and only puts in a standard 2 × 4 (1¾" thick) instead of the usual 4 × 4, the door can open almost 2". Guess what happens when someone pulls on the door from outside? An unlocked door is an embarrassing predicament.

Needless to say, I apologized profusely. It was a false alarm. If we'd nabbed Charles Manson, wasting their time wouldn't be an issue. I probably could have even gotten away with making



Photo 1—The alarm system PIN is entered via a keypad. The ASDL system monitors and records the time and date of all key presses.

police and donut-shop jokes. Under the circumstances, however, my only recourse was to thank them, tuck my tail, and return to the car.

I dialed the cell phone. When Jeannette answered, I said in a disgusted tone, "Someone left the damn door open! The only thing worse would be if they didn't set the alarm at all!"

"Well, Steve, I wasn't going to tell you, but I've had reports from the people who come in early that occasionally the alarm hasn't been on."

Next morning, I called everyone with alarm codes together and asked the pertinent who and when questions. I got back mostly blank stares, to be interpreted as, "Not me, man." Given all the people with independent access to the building, that was hardly reassuring.

The obvious answer was the alarm company. We pay them \$30 a month to monitor the system and call the appropriate people if the alarm goes off. When it was first installed, we got a monthly opening/closing report that listed the date, time, and access code (these days I suppose we'd call it a PIN) for every alarm set or reset. This was the obvious answer.

Calling the alarm-monitoring service is an experience. They're contracted by your alarm installer and not typically selected by the alarm owner. And since they answer the phone "Monitoring station" and use the installer's name once you give them your account number, you might think they're just down the block. Only when you interrogate the autodialer or otherwise see where the call goes do you realize that your personal monitoring can be 2,000 miles away.

Our monitoring company was at the other end of the state, but most large alarm companies deal with centralized service monitors that cover many states at the same time. Regardless of their location, aside from changes to the call list, they're

like talking to a brick! Their pat response is that you should call your installer.

They usually charge the installer a flat rate based on a specific service level for all his customers. If he only contracts for alarm calling and none of the monthly printed reports, it's tantamount to bringing the mountain to Mohammed to get one from the monitoring company. Yes, I could get a report for a specific day—at \$20 each!

Calling the installer reminded me again why we designed our own home-control system. These people have no vision at all.

"What would it take to get daily entry/exit reports?" I asked.

"I suggest you install a new alarm system," he answered matter-of-factly.

"Would a new system be better than our present 10-year-old hard-wired system?"

"Well, sir," he continued. (Subconsciously, I noted that people generally called me "sir" when they were trying to sell me something. Sometime I'll have to test the financial limits of this theory. Do they start at \$100, \$500, \$1,000...?) "It would have the latest technology and use wireless sensors."

"And after I replace the \$2 battery in every sensor each year, would it do more than provide a contact closure to an alarm horn and autodial a digital code to the monitoring station like the present one?" I already knew the answer.

"It would use the latest technology to close the contact and autodial the modem, sir...and we could get you one with a printer output." Finally, he mentioned something we wanted!

I continued, "Do you know if it's a standard serial printer port? Do you have a schematic?"

"Well, I'm not sure what it is, sir, but I can supply you with the printer (extra cost). There are never any schematics of any equipment, sir. I guess they're concerned about liability."

Liability, schmiliability. The only reason there are no schematics is that the alarm manufacturers don't want competition.

In the end, it's just a cost decision. "How much?"

"Well, sir, if we do just the same as your present system and add the printer, probably about \$3,500–\$4,000."

I knew this "sir" thing was going to cost me. "Thanks, I'll have to get back to you."

My next stop was Jeff's desk.

"Jeff, this alarm guy is nuts. You can't believe how much it costs for an alarm-code printout. Worse yet, it only records successful entries. It can't tell who punches in a bunch of numbers, never actually sets the alarm, and then just walks out without checking. That's the guy I want."

I knew telling Jeff about such a ridiculous obstruction would be a technical challenge he couldn't refuse. They may not know how to produce a daily event record, but somehow we would. We gathered up the 'scope and

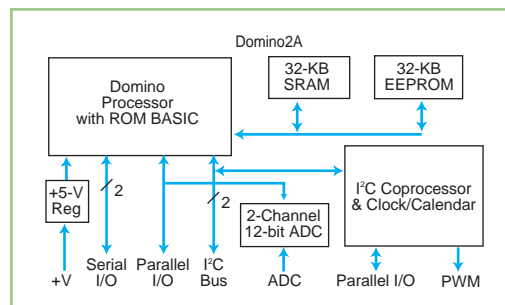


Figure 1—Domino2 is a small encapsulated controller with a built-in floating-point BASIC interpreter and 32 KB each of EEPROM and SRAM.

FROM THE ARCHIVES

headed for the furnace room and the alarm-system controller box.

There was nothing surprising about the system. It was your standard ten-year-old Silent Knight alarm system. Documentation was strictly at the installer level. Motion detectors and sensors connect to these terminals, the alarm horn connects here, the phone line goes in here.

The PCB hardware included two micro-controllers that shared the control tasks. Because they had house numbers, Jeff and I concluded they were probably ROM-programmed 8051-family devices. And because it was ROM coded, we didn't have a prayer of changing any internal operation. The best we could hope to do was monitor the external signals.

It would have been great if the alarm designers had taken a traditional engineering approach. Even a *predictable* approach would have been welcomed. In the world of high-volume consumer-quality electronics however, nothing works the way you expect it to, and “cute” is a term frequently used to describe the design technique.

The methodology is quite straightforward. Take a circuit that works the way you'd normally design it, throw away half the parts (or parts cost), and then make it do all the same control tasks. These are the guys who make a fine art of cycle-stealing, multiplexed operation, and hairy-edge qualification. Want them to design your next medical product?

This unit was no exception. The user interface consisted of a 3" x 4" keypad with a green LED (ready), a red LED (armed), and a single seven-segment LED display (zone). The keypad labels were 0–9, Door, and On/Off. Pressing any button makes an internal beeper sound. [Photo 1](#) shows a close up of the alarm's keypad.

Similar to the operation of most alarm systems, the user looks for a green light indicating the system has no open doors and punches in a four-digit code followed by the

on/off button. The alarm goes on, the red LED lights, and you have about 45 s to vacate the building. The process is reversed to shut off the alarm.

Jeff and I concluded the way for us to monitor entry and exit times and codes was to attach a circuit in parallel with this user interface and tap into its communication with the system box. Punch in 6637 and On, and we'd log it to a printer. The only sticky part was guessing where all these signals were so we could tell which key was pressed.

The keypad in the entryway connects to the system via an 11-wire parallel cable. The 3" x 4" keypad has four rows labeled 1-3, 4-6, 7-9, and Door, 0, and On/Off.

Electrically, we determined that the keys are scanned in two separate groups of six. Two opposite and alternating signals drive the two groups. Pressing none of the keys results in a logic 0 on the three column inputs back at the system board.

Pressing a key diode ORs one of the phases with one or more of the column inputs, like this: 1 = 001, 2 = 010, 3 = 011. A 7 also creates the 001 combination

but in step with the opposite phase. The system knows which key is pressed based on the column inputs and the phase of the input signal.

Other signals to the entry panel enable the red or green LEDs and drive a piezo beeper when any key is pressed. All signals are at the 12-V level.

PACKAGED SOLUTION

Jeff and I now knew there were some logical signals we could monitor. The next task was to decide what kind of data-acquisition system we had to configure. But unlike our lightning device ("Ground Zero," *INK* 90), this wasn't just an illustrative magazine project. I wanted to use this thing.

We could have used anything from a PIC to a full-blown PC as the hardware. Our logging system needed a processor board to acquire and analyze the data, a real-time clock/calendar to time stamp the entries, an LCD to view records, a keypad to direct the logger's activity, and a printer interface for making hardcopies on command.

Beyond the strict hardware necessity, system selection is always a tradeoff of

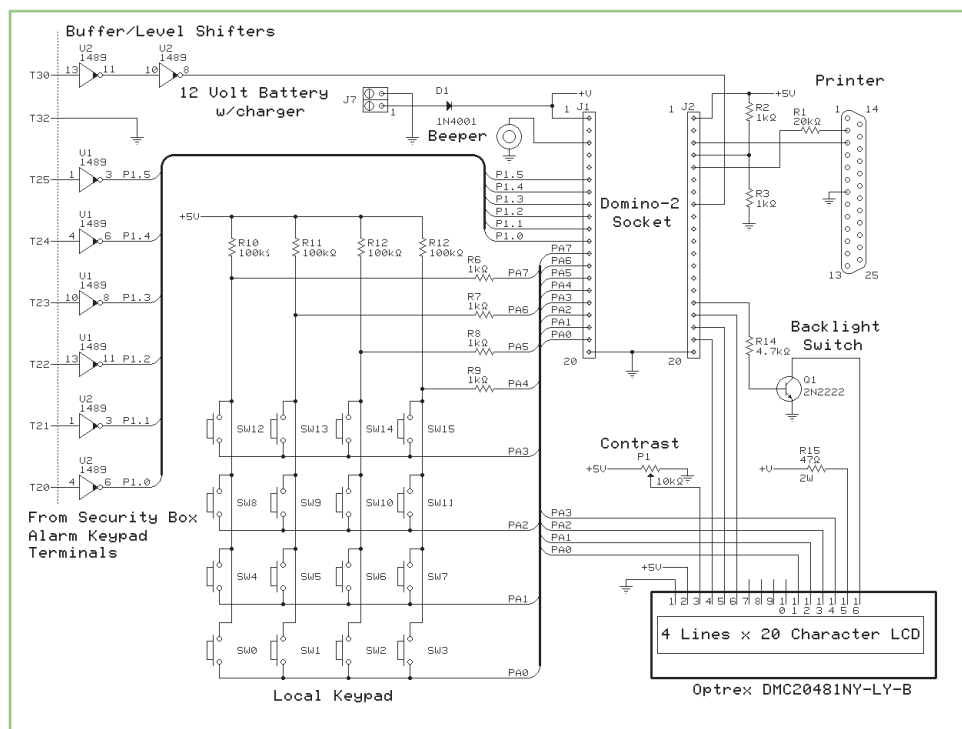


Figure 2—This schematic shows the ASDL system and how it connects to the alarm signals.

FROM THE ARCHIVES

competing ideals:

- time (getting this much software done quickly enough to meet a magazine deadline typically rules out assembly language)
- I/O capability (obviously, we needed a serial port and a lot of parallel I/O)
- speed (just how fast does this thing need to be anyway?)
- cost (are we making a few or is it a volume-production device?)
- political bias (some designers will jam in a PC even if it can be done on a PIC)

This analysis pretty much fits half the board ads in *INK*. Fortunately, I get to apply a little political bias of my own.

While there's a little fancy footwork in the interrupt routines, most of the software is a lot of text shuffling among the peripherals (it's easy for the guy who doesn't write the software to say stuff like this). When we looked at the requirements, it seemed like a perfect application for a Domino—or more precisely, a Domino2.

As [Figure 1](#) shows, Domino2 is a small encapsulated controller with a built-in floating-point BASIC interpreter and 32 KB each of EEPROM and SRAM. Best of all, it contains a serial port, lots of parallel I/O, and a real-time clock.

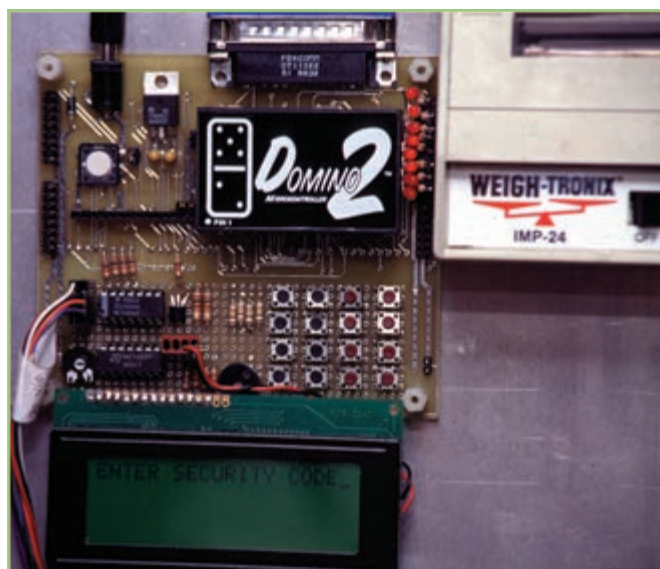


Photo 2—The prototype ASDL system is physically assembled using a Domino development board with an LCD and keypad attached.

Even if you don't have a 10-year-old alarm, I'm sure you'll find that our method of solving the problem provides some interesting examples of using BASIC-52 in control applications.

Gigabit Technology

- ARM-based
- System on a Chip
- Gigabit Ethernet
- Small, Cheap, Fast
- Both QFP and BGA Packages
- Standard Development Tools
- 10 Year Life Guarantee
- Royalty Free RTOS, TCP/IP V4/6

\$10.00 each (Qty 100)

The gridARM™ System on a Chip (SOC) is a high performance, low cost, low power, highly integrated single chip with 10 / 100 / 1000 Mbps Ethernet, USB, CAN, Serial, SRAM Memory, SPI, I2C, RTC and internal peripherals designed to provide a complete solution for embedded applications.

THE NETWORKING EXPERTS

gridconnect.

800.975.4743 USA • 1 630.245.1445
gridconnect.com/gridarm.html

Leaders in the embedded and networking marketplace providing network hardware, high quality software and services

The Convenient All-in-One Solution for Custom-Designed Front Panels & Enclosures

FREE Software

ONLY \$90.24 with custom logo engraving

You design it
to your specifications using our FREE CAD software, Front Panel Designer

We machine it
and ship to you a professionally finished product, no minimum quantity required

- Cost effective prototypes and production runs with no setup charges
- Powder-coated and anodized finishes in various colors
- Select from aluminum, acrylic or provide your own material
- Standard lead time in 5 days or express manufacturing in 3 or 1 days

FRONT PANEL EXPRESS
FrontPanelExpress.com
1(800)FPE-9060

FROM THE ARCHIVES

ALARM-SYSTEM DATA LOGGER

Figure 2 is the schematic of the alarm-system data logger (ASDL). The circuitry was added to Domino's proto1 board, as shown in Photo 2. The first task was mating the 12-V level alarm signals with Domino's TTL input levels.

There are lots of ways to do this, but one cute way is to use MC1489 RS-232 input level shifters. These inexpensive inverters can withstand ± 30 -V inputs while interfacing directly with TTL on the output side. Six keypad lines connect to the processor.

The next detail is determining when a key is pressed. We had two alternatives. We could create a falling-edge key-press interrupt by NORing the three column inputs together. Or, we could take advantage of the fact that pressing any key caused the beeper to sound.

Ultimately, Jeff chose to use the signal applied to the beeper as a key-press interrupt. The three column lines, two phase outputs, and on/off button are read anytime there's an interrupt. Together, they determine the physical key combination.

Normal entry and exit profiles run a total of 10 key presses a day. This plus a 7-byte time and date stamp amounts to fewer than 20 bytes of data-logging space required per day.

Our plan isn't to check this thing daily but to have a record of events available when we need it. If we have a good chunk of data storage space available, how often we dump the records will never really be an issue. Six months of data fits easily in less than 5 KB of memory. We have about 24 KB available.

Program development started with writing the time and date to the LCD. Instead of using a couple I²C peripheral chips to connect the 4 × 20 LCD and 4" × 6" keypad as typically described for Domino, Jeff chose to minimize external hardware and scan the command keypad in software.

Using this number of I/O bits for the keypad necessitated using the LCD in 4-bit mode rather than the typical 8-bit parallel interface. The 4-bit mode requires two nibble writes to the LCD for each printed character, which entails eight physical operations for each character—set up the control register, raise the strobe line, set up the data register, drop the strobe line. The process repeats for the second data nibble.

The program executes a short initialization routine and then prints day of the week (DOW), month/day/year, and hour/minute/second on the LCD. Using the row and column positioning capability of the LCD (a control register routine), only new data needs to be updated. This makes a great idle screen that also indicates the system is running.

If the displayed time and date is incorrect, as it would be on initial powerup, the user enters the present DOW, month, day, year, hour, minute, and second. Maintaining the time during a power outage is a simple matter of adding a 4.5-V back-up battery to the real-time clock.

Since the alarm system is normally battery-backed and may be needed during a power outage, we decided to battery-back the whole Domino2 circuit. Besides keeping the RTC alive, it maintains any collected data that hasn't been printed yet. Figure 3 gives an overall view of how the complete system was integrated.

FANCY FOOTWORK

As I mentioned, it's easy to say it's just a matter of software when you don't have to write it yourself. One of the more challenging aspects of the software was catching the key presses while doing all the other system functions.

This task was accomplished via the BASIC-52 ONEX1 command. This interrupt command temporarily redirects the normal program flow to a special subroutine.

ONEX1 is triggered by a high-to-low transition on Domino's INT1 input pin. Obviously, it should be reserved for a high-priority function. Listing 1 illustrates the high points.

The piezo beeper output connects to INT1. When ONEX1 is triggered, the routine samples the I/O port where all the row and column data is connected to the Domino. This beeper interrupt can occur during either phase of the row drivers.

To ensure the sample is valid, the program checks whether there is a high level on at least one column input line. If none of the first three bits are high, we assume it's the wrong phase and take another sample. Once there is a valid sample, the row driver levels (indicating the two phases) can be compared with the column inputs to determine which of the 12 keys was pressed.

Before leaving the interrupt routine, there are two more tasks. First, save the key-press code into the data-logging buffer and increment the buffer pointer. If the on/off key was pressed, a time/date stamp is added to the log.

Finally, we need to make sure the physical key-press action has ended (to prevent interrupt on the same key press). This is accomplished by repeatedly sampling the port and looking for invalid data (columns all low) on both phases of the row driver outputs. When RETI is executed, the program resumes where it left off.

GETTING THE GOODS

Displaying and dumping the stored data log is a secondary

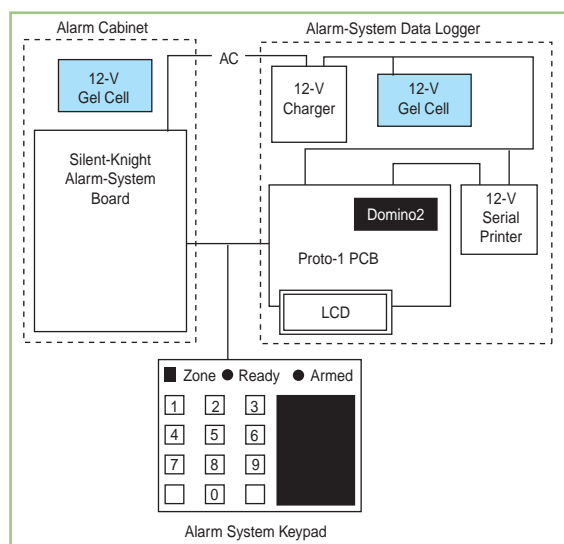


Figure 3—The ASDL attaches in parallel with the existing alarm keypad. The Domino2-based monitor program registers and records keypad activity and outputs the data to an LCD and printer on command.

function chosen from the command menu. The ASDL LCD can show a command list, the time and date, or the data. The data is printed to the LCD in blocks.

When the log is dumped, all the blocks up to and including the present time and date are dumped. We didn't see the necessity for individual date interrogation.

You want the data log? Here's everything, a block at a time.

While the data is being sent to the LCD, it is also sent out the console serial port. Although this could be connected to a PC and logged to a file, Jeff

mounted a plain-paper 12-VDC-powered serial printer (measures only 5" × 5" × 3") next to the Domino, so a hardcopy of the data could be printed and retained if necessary.

A short pause between each block lets the printer keep up with the data and provides time to study each block on the LCD before the next display. The final menu selections enable the user to clear all data from the data buffer and return from the menu screen to the idle screen (displaying time/date).

Since the capturing and logging of

alarm codes is a security risk, it would be foolish not to try to prevent unauthorized use. Accessing the menu screen shouldn't be open to just anyone.

Any ASDL entry prompts the user for a PIN. An incorrect entry simply returns to the idle screen.

We also specifically chose to store the data log in SRAM rather than EEPROM. If the Domino is removed from the system, the data log disappears.

GOTCHA

While solving the problem this way gave me the satisfaction of not having to buy more useless equipment from an alarm company, it turns out there was no better alternative.

The monitoring station and a commercial alarm printer could only give me a list of successful alarm operation. But, my major irritation was with people who didn't look for a green LED (indicating that the building was clear) before blindly punching in a code (that the alarm doesn't accept) and blowing out the door.

The ASDL stores every key pressed. It only adds the time/date stamp when the on/off key is pressed. If the alarm isn't set, a quick review of the record should show what code was being entered, even if unsuccessfully.

All of our work may have been in vain. We haven't had any false alarms, and interestingly, there haven't been any mornings with an unset alarm.

Either everyone has become educated by involvement in publishing this article, or seeing Jeff and me constantly playing with the alarm system has made them aware that something is going on.

Regardless, they seem to recognize the seriousness of it all. Jeff and I just have to be careful that we don't let on that we really have fun. ☺

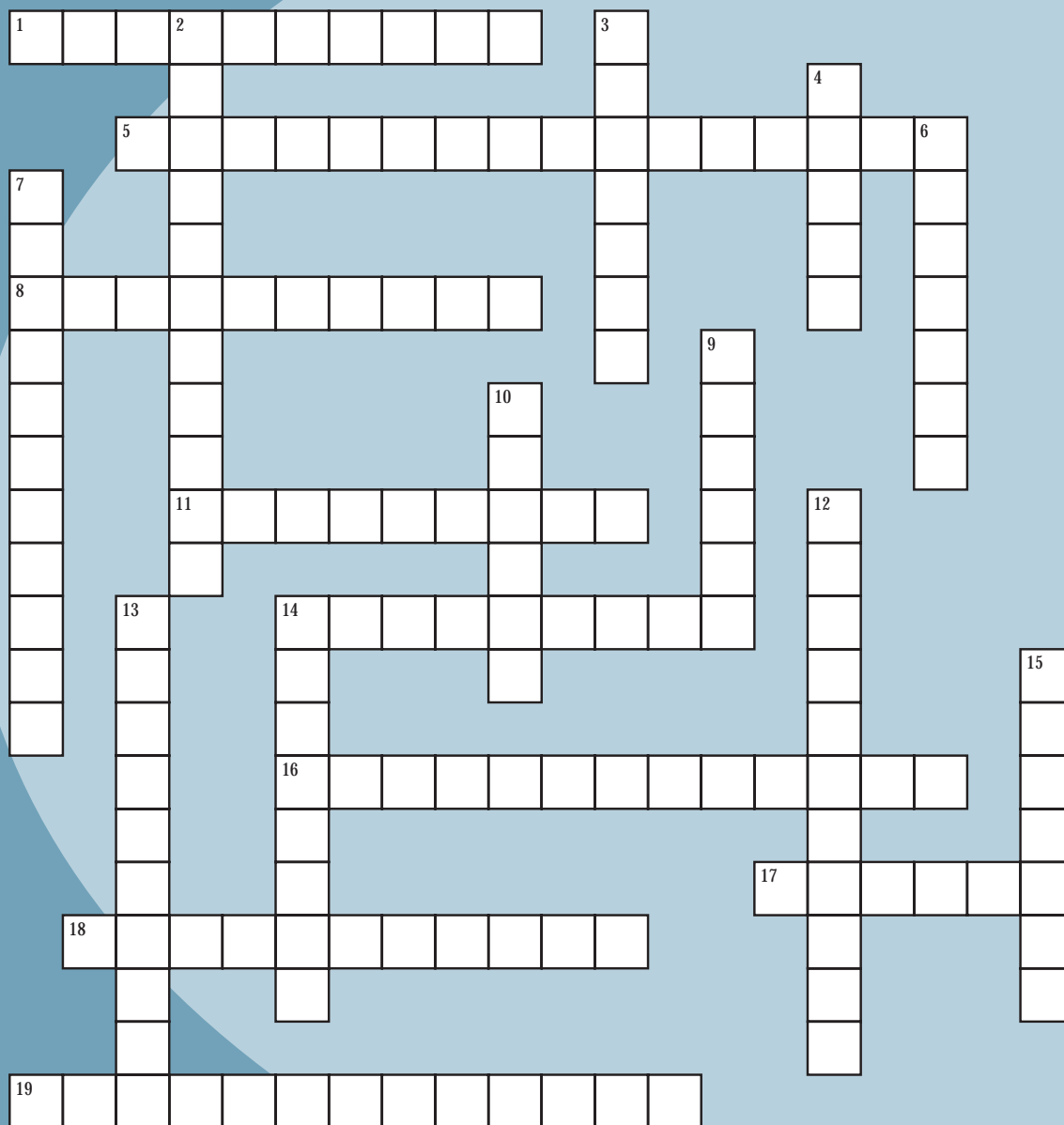
Steve Ciarcia is an electronics engineer who founded Circuit Cellar in 1988. His background is in process control, digital design, and product development.

Jeff Bachiochi is an electrical engineer and monthly Circuit Cellar columnist. His background includes product design and manufacturing.

Listing 1—The highest priority was given to the monitoring of the alarm keypad's key presses. The BASIC ONEX1 command branches to line 20000 to decode a key press and log it.

```
120 ONEX1 20000 : REM CAPTURE ALARM KEY PRESSED
...
20000 REM INTERRUPT ROUTINE - EXTERNAL BUTTON PUSHED
20010 LED=PORT1.XOR.255 : REM INVERT DATA
20020 IF (LED.AND.07H)=0 THEN GOTO 20010 : REM CHECK FOR DATA
20025 LED=LED.AND.3FH
20026 BUT=LED.AND.37H : REM MASK OFF ALL BUT BUTTON INFO
20030 IF BUT=15H THEN GOTO 21000 : REM BUTTON 0
20035 IF BUT=21H THEN GOTO 21100 : REM BUTTON 1
20040 IF BUT=22H THEN GOTO 21200 : REM BUTTON 2
20045 IF BUT=23H THEN GOTO 21300 : REM BUTTON 3
20050 IF BUT=24H THEN GOTO 21400 : REM BUTTON 4
20055 IF BUT=25H THEN GOTO 21500 : REM BUTTON 5
20060 IF BUT=26H THEN GOTO 21600 : REM BUTTON 6
20065 IF BUT=11H THEN GOTO 21700 : REM BUTTON 7
20070 IF BUT=12H THEN GOTO 21800 : REM BUTTON 8
20075 IF BUT=13H THEN GOTO 21900 : REM BUTTON 9
20080 IF BUT=14H THEN GOTO 22000 : REM BUTTON DOOR
20085 IF BUT=16H THEN GOTO 22100 : REM BUTTON ARM/DISARM
20090 RETI
20100 Z=Z+1 : XBY(Z)=OFFH : REM TAG END OF BLOCK
20110 LED=PORT1.XOR.255
20120 IF (LED.AND.17H)<>10H THEN 20110 : REM WAIT FOR NO DATA
20130 LED=PORT1.XOR.255
20140 IF (LED.AND.27H)<>20H THEN 20130 : REM WAIT FOR NO DATA
20190 RETI
21000 XBY(Z)=30H
21010 GOTO 20100
...
22000 XBY(Z)=ASC(D)
22010 GOTO 20100
22100 REM ARMED?
22101 IF (LED.AND.08H)=0 THEN GOTO 22200
22110 XBY(Z)=ASC(N) : REM NOT ARMED
22111 GOSUB 23000
22120 GOTO 20100
22200 XBY(Z)=ASC(A) : REM ARMED
22201 GOSUB 23000
22220 GOTO 20100
23000 REM STICK IN TIME/DATE STAMP
23005 Z=Z+1
23006 XBY(Z)=0AAH
23010 Z=Z+1
23011 XBY(Z)=MTH
...
23060 Z=Z+1
23061 XBY(Z)=SEC
23080 RETURN
```


CROSSWORD



Across

1. The first widely used television camera tube
5. Responds to disturbances [two words]
8. On the right of an integer, on the left of a fraction [two words]
11. Bridge circuit used to measure resistance
14. A versatile, easy-to-design filter [two words]
16. Light scattering [two words]
17. A kind of passive filter
18. MCU pin [two words]
19. The result never changes [two words]

Down

2. Combines two types of functions in a binary circuit with two or more inputs and one output [two words]
3. Won the Nobel Prize in Physics twice
4. Developed in 1904 by English engineer John Ambrose Fleming
6. A device that receives part of a transmitted pulse and transmits it back to the receiver [two words]
7. Used to simplify algebra expressions [two words]
9. English scientist (1791–1867) who published the law of induction
10. Tuning that uses a single control to tune two or more circuits
12. French instrument maker Hippolyte Pixii developed a prototype for this in 1832 [two words]
13. What an LED does
14. A waveform with a slow linear rise time and a fast fall time
15. An absolute-positioning actuator that is typically limited to a 180° rotation [two words]

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

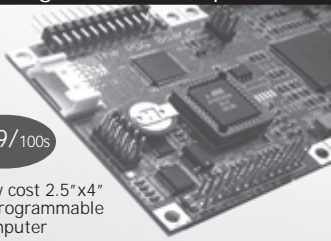
IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

AD FORMAT: Advertisers must furnish digital files that meet our specifications (www.circuitcellar.com/advertise). ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2" x 3" FORMAT. E-mail adcopy@circuitcellar.com with your file or send it to IDEA BOX, Circuit Cellar, 4 Park Street, Vernon, CT 06066. For current rates, deadlines, and more information contact Peter Wostrel at 978.281.7708 or peter@smmarketing.us.

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

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



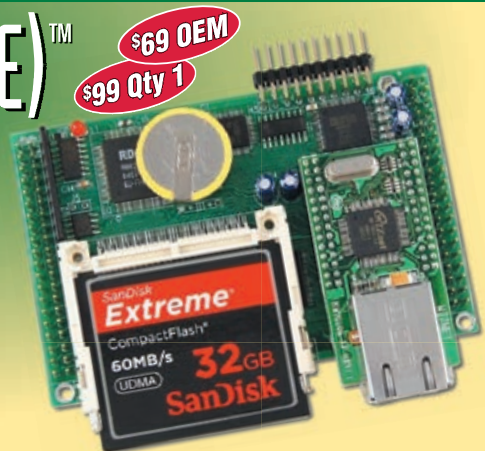
\$159/100s

- 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

B-Engine (BE)™

- **OEM CPU board with high speed $\pm 10V$ bipolar 16-bit ADC (AD7606), DACs**
- **100M Base-T Ethernet, CompactFlash**
- **3.6" x 2.3", C/C++ programmable.**



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.



1950 5th Street, Davis, CA 95616 USA

Tel: 530-758-0180 • Fax: 530-758-0181



www.tern.com • sales@tern.com

NEW CCS www.ccsinfo.com
sales@ccsinfo.com
262-522-6500

VERSION 5 IS THE CODE CONQUEROR

- Flow control and Interrupt buffered to serial routines libraries-specify size of transmit buffer, size of receive buffer, Interrupt usage or no Interrupt usage, pin for CTS and pin for RTS
- C Profiler-continuous logging and analyzing run-time events to give a profile of the program
- IDE Enhancements-faster debugging, upgraded wizards and Windows 8 compatible!
- Input Capture and PWM Libraries-make better use of capture/compare PWM, input capture and output capture peripherals on the PIC MCU

WWW.CCSINFO.COM/CCVER5

ANALOG and DIGITAL for PC/104



GPIO-104
8 ANALOG INPUTS
4 ANALOG OUTPUTS
24 DIGITAL I/O

ADIO-104
16 ANALOG INPUTS
8 ANALOG OUTPUTS
24 DIGITAL I/O
PULSE ACCUMULATOR

Need More Analog I/O?

SCIDYNE Also Available:
• 96-DIGITAL I/O
• 24 HIGH POWER I/O
• RELAY/OPTO I/O
• EXTERNAL EXPANSION
• PC/104 PROTOTYPING

1-877-SCIDYNE
info@scidyne.com
www.scidyne.com

I²C™/SMBus



- Bus Monitors
- Protocol Analyzers
- Host Adapters
- Multiplexers
- Battery Applications
- Software Tools

MCC
Micro Computer Control

www.mcc-us.com

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

BPS BusBoard
Prototype
Systems

**Prototyping
PC-Boards**
Many Patterns

BusBoard
StripBoard
PadBoard
ProtoBoard-2H
ProtoBoard
PowerBoard
SMTpads
PC BreadBoards

See our site:
www.BusBoard.us

Available From
JAMECO ELECTRONICS
MOUSER ELECTRONICS
amazon.com amazon.co.uk

LISTEN TO YOUR MACHINES

Ethernet PLCs for OEMs



**FMD88-10
and FMD1616-10**

Integrated Features :

- ETHERNET / Modbus TCP/IP
- 16 or 32 digital I/Os
- 10 analog I/Os
- RS232 and RS485
- LCD Display Port
- I/O Expansion Port
- Ladder + BASIC Programming

\$229 and \$295

before OEM Qty Discount

tel : 1 877 TRI-PLCS
web : www.tri-plc.com/ccf.htm

TRI TRIANGLE
RESEARCH
INTERNATIONAL

**Get
PUBLISHED.
Get
NOTICED.
Get
PAID.**

Circuit Cellar feature
articles are contributed by
professional engineers,
academics, and students
from around the globe.

Do you have what it takes?

Contact C. J. Abate, Editor-in-Chief,
to discuss the embedded design
projects and programming
applications you've been working
on and your article could be
featured in an upcoming issue.



microEngineering Labs, Inc.

www.melabs.com 888-316-1753

Programmers for Microchip PIC® Microcontrollers



Starting at \$79.95

PC-Tethered USB Model (shown):

- Standalone software
- Command-line operation
- Hide GUI for automated use
- Override configuration with drop-downs

Stand-Alone Field Programmer:

- Power from target device or adapter
- Program file stored on SD-CARD
- Programming options stored in file
- Single-button operation

Program in-circuit or use adapters for unmounted chips.

Zero-Insertion-Force Adapters available for DIP, SOIC, SSOP, TQFP, and more.

PIC is a registered trademark of Microchip Technology Inc. in the USA and other countries.

CIRCUIT CELLAR
editor@circuitcellar.com

3-Port CAN Repeater helps distribute CAN bus to multiple trunk lines.

CAN REPEATER
\$164

(623)-399-4688
www.apoxcontrols.com
sales@apoxcontrols.com

USB-CAN \$125
USB-ISOCAN \$157

Free Windows Example source code written in C++ available.

All CAN Features are programmable. 11&29 bit identifiers, Filters, Masks, Baud rate up to 1Mbps.

Optically Isolated version available for harsh conditions.

CAN-Repeater has DIN rail mount for your industrial machine applications. (Requires +9 to +24V supply) Great device to distribute power and CAN to many Nodes on your Bus! This is a great device to help clean up the noise on CAN buses which have too many nodes. Each branch has selectable termination to optimize bus reflections.

Techsol Engineering

**Commercial
Industrial
Medical
ANDROID
DEVICES**

www.techsoleng.com

MaxSonar
Ultrasonic Ranging is EZ

LV-ProxSonar-EZ
• People sensing made easy
• Proximity Sensor
• MSRP \$29.95

HRXL-MaxSonar-WR (IP67)
• 1-mm resolution
• Industrial packaging
• Weather resistant
• MSRP \$109.95

I2CXL-MaxSonar-EZ
• Great for UAV's
• High noise immunity
• I2C interface
• Starting at \$39.95

I2CXL-MaxSonar-WRC (IP67)
• I2C interface
• Compact packaging
• Weather resistant
• MSRP \$109.95

www.maxbotix.com

A Guide to Powerful Programming for Embedded Systems

Assembly Language Essentials

Larry Cockfield

Shop for this book, and others, at
www.cc-webshop.com

CIRCUIT CELLAR

Serial Commands Touch Events

Your Microcontroller Our Display Module

Add a Touch Screen to Your Embedded Product

- No special OS or Library Required.
- Programming GUI is Simple.
- Development Kit = Up and Running in Days

Learn more at
www.reachtech.com,
or contact us at
510-770-1417 or
sales@reachtech.com.

REACH
TECHNOLOGY INC.

Development Kits include serial LCD controller board, display, touch screen, cable sample images/code, power supply, technical support, 100% Satisfaction Guarantee

Propeller Debugger

ViewPort brings powerful tools to the Parallax Propeller

- Step line by line/breakpoint
- Performance profiler
- 80 MSps analysis of IO pins
- Realtime graphs
- Change values with custom interfaces
- Rewind time and zoom in/out
- Win/OSX/Linux
- C/BASIC/SPIN IDE

hannoware.com/viewport

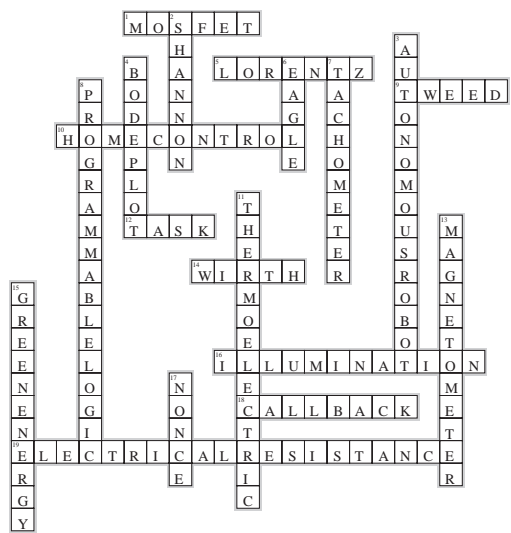
CROSSWORD ANSWERS from Issue 269

Across

- MOSFET**—According to Ed Nisley in his *Circuit Cellar* 265, 2012 article, this type of tester characterizes a transistor's behavior by computing the drain resistance at each combination of measured voltage and current
- LORENTZ**—Type of force on a charged particle caused by electromagnetic fields
- TWEED**—Tests your engineering know-how in every issue of *Circuit Cellar*
- HOMECONTROL**—In last month's "Task Manager," *Circuit Cellar* Editor-in-Chief C. J. Abate mentioned that this was one of the hottest topics in the magazine's earliest issues [two words]
- TASK**—In his article in the December issue, Bob Japenga defines this as an instance of a software program that is utilizing CPU resources to accomplish some purpose
- WIRTH**—Swiss computer scientist who designed the Pascal programming language
- ILLUMINATION**—An LED's purpose
- CALLBACK**—Enables a lower-level software layer to request a higher-level-defined subroutine
- ELECTRICALRESISTANCE**—German physicist Georg Ohm 1789–1854 first introduced this concept [two words]

Down

- SHANNON**—Cryptographer known as the "father of information theory"
- AUTONOMOUSROBOT**—Does not rely on human interaction [two words]
- BODEPLOT**—Represents a system's gain and phase as a frequency function [two words]
- EAGLE**—Commonly used for PCB design
- TACHOMETER**—A device that can determine RPM
- PROGRAMMABLELOGIC**—These types of projects utilize FPGAs, PLDs, and other chips [two words]
- THERMOELECTRIC**—Type of cooling that relies on the Peltier effect to alter heat between two types of materials
- MAGNETOMETER**—Used to measure magnetic fields' strength and intensity
- GREENENERGY**—Focus of Renesas's 2012 design challenge [two words]
- NONCE**—Available for a limited time



Can anyone deny that we're on the verge of some major breakthroughs in the fields of microcomputing, wireless communication, and robot design? Tech the Future is a recurring section devoted to the ideas and stories of innovators who are developing the groundbreaking technologies of tomorrow.

Can MoS₂ Outperform Silicon?

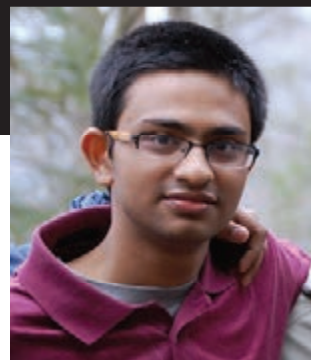
by Saptarshi Das

After decades of relentless progress, the evolutionary path of the silicon CMOS industry is finally approaching an end. Fundamental physical limitations do not enable silicon to scale beyond the 10-nm technology node without severely compromising a device's performance. To reinforce the accelerating pace, there is an urgent and immediate need for alternative materials. Low-dimensional materials in general, and 2-D layered material in particular, are extremely interesting in this context. They offer unique electrical, optical, mechanical, and chemical properties. In addition, they feature excellent electrostatic integrity and inherent scalability, which makes them attractive from a technological standpoint. Graphene, hexagonal boron nitride (h-BN), and more recently the rich family of transition metal dichalcogenides—comprising Molybdenum disulfide (MoS₂), WS₂, WSe₂, and many more—have received a lot of scientific attention as the future of nanoelectronics. The most widely studied material, graphene, had reported intrinsic field effect mobility value as high as 10,000 cm²/Vs. However, the absence of an energy gap in the electronic band structure of graphene, along with the challenges associated with making a stable interface with the gate dielectric, raises a lot of concern for graphene-based nanoelectronics for logic applications. Hence, it paves the way for semiconducting 2-D materials such as MoS₂ and others.

MoS₂ is a stack of single layers held together by weak van der Waals interlayer interaction, and, therefore, enables micro-mechanical exfoliation of one or a few layers—similar to the fabrication of graphene from graphite. It is a semiconductor with an indirect bandgap of 1.2 eV. Single- and multilayer MoS₂ field-effect transistors (FETs) with high on/off-current ratios (108) and excellent subthreshold swing (74 mV/decade) close to the ideal limit have been demonstrated. Basic integrated circuits (e.g., inverters and ring oscillators) have been reported. And initial studies also indicate that MoS₂ has great potential in future nanoelectronics, sensing, and energy harvesting.

While there is a growing interest in MoS₂-based nanoelectronics devices, the practice of evaluating their potential usefulness for electronic applications is still in its infancy since we don't have a complete picture of their performance potential and scaling limits. My research addresses the major issues about the realization of high-performance logic devices based on ultra-thin MoS₂ flakes. One of the major challenges in the realization of high-performance nano devices arises from the fact that these nanostructures need to be connected to the "outside" world to capitalize on their ultimate potential. Any interface between a low-dimensional nanostructure and a 3-D metal contact will inevitably affect the total system's performance, which will strongly depend on the said contact's quality. We have demonstrated that through a proper understanding and design of source/drain contacts and the right choice of the number of MoS₂ layers to use, the excellent intrinsic properties of this 2-D material can be realized. Using scandium contacts on 10-nm-thick exfoliated MoS₂ flakes that are covered by a 15-nm Al₂O₃ film, record high mobilities of 700 cm²/Vs are achieved at room temperature. This breakthrough is largely attributed to the fact that we succeeded in eliminating contact resistance effects that limited the device performance in the past unrecognized. We have also investigated the ultimate scaling potential of multilayer MoS₂ field effect transistors (FETs) with channel lengths ranging from 1 μm down to 50 nm. Our results indicate that the multilayer MoS₂ FETs are extremely resilient to short channel effects. We have demonstrated record high drive current density of 2.5 mA/μm and record high transconductance of 500 μS/μm for a 50-nm-long MoS₂ transistor, which are comparable to state-of-the-art silicon technology.

In short, MoS₂ preserves all the important properties of silicon with the added advantage of an ultra-thin layer structure, which allows for aggressive channel length scaling down to 2 nm and, therefore, has the potential to outperform silicon beyond the 10-nm technology node. Properly nourishing the development of MoS₂ can be a real game changer for the future of the micro- and nanoelectronics industry.



Saptarshi Das was born in Kolkata, India, in 1986. He holds a PhD in Electrical Engineering from Purdue University and a BS from Jadavpur University. Saptarshi received the IBM PhD Fellowship award for the 2011–2012 academic year. During the summer of 2011, he worked at IBM's Systems and Technology Group in Fishkill, NY. In 2009, Saptarshi interned at IBM's T. J. Watson Research Center in Yorktown, NY. His research focuses on the experimental realization of high-performance and low-power device design using low-dimensional nano materials. He also works on device simulation, circuit modeling, and the optimization of novel nano transistors.

www.ftdichip.com



ENHANCED USB PERFORMANCE

Streamlined USB
Bridge Solutions

X-CHIP

EXtensive Interfaces
UART, FIFO, SPI, I²C, FT1248

EXtended Features

Battery charger detection

Low active power (8 mA, typical)

Internal MTP memory

Expandable clocking; clock generation
and system clock out

EXceptional Drivers

Windows, MacOS, Android, and Linux



Power of the **Quoting Revolution** in the palm of your hand!

At a click of a button we will find the best
price in the market compiled in one easy to
read spreadsheet
...just like your BOM.



Join the Revolution
Happening Now @
www.PCBnet.com

847-806-0003 sales@PCBnet.com
ITAR, I SO 9001:2008, UL Approved