circuitcellar.com

MEASUREMENT & SENSORS
MAY 2014
ISSUE 286

circuit cellar

RFID ENGINEERING

A look at tags, readers, circuitry, and data delivery

- New Embedded Tech Highlights | Q&A: Embedded Systems Expert
- Eco-Friendly Home Controller | RC-Powered Lift | Parse & Execute Remote Commands
- Passive RFID Technology | Data Centers and Smart Grids | Wireless Receivers | Battery Analysis
- Brighter LED Micro-Displays

$9.00US $10.00CAN
05
0 74470 75349 0

Issue 286 May 2014 | ISSN 1528-0608

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by:

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

Periodical rates paid at East Hartford, CT, and additional offices. One-year (12 issues) subscription rate US and possessions $50, Canada $65, Foreign/ ROW $75. All subscription orders payable in US funds only via Visa, MasterCard, international postal money order, or check drawn on US bank.

SUBSCRIPTIONS

Circuit Cellar, P.O. Box 462256, Escondido, CA 92046

E-mail: circuitcellar@pcspublink.com

Phone: 800.269.6301

Internet: circuitcellar.com

Address Changes/Problems: circuitcellar@pcspublink.com

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

ADVERTISING

Strategic Media Marketing, Inc.
2 Main Street, Gloucester, MA 01930 USA

Phone: 978.281.7708

Fax: 978.281.7706

E-mail: circuitcellar@smmarketing.us
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

HEAD OFFICE

Circuit Cellar, Inc. 111 Founders Plaza, Suite 300
East Hartford, CT 06108
Phone: 860.289.0800

COVER PHOTOGRAPHY

Chris Rakoczy, www.rakoczyphoto.com

EDITOR'S LETTER

A TIMELY LOOK AT RFID TECHNOLOGY

Most of us have had that annoying experience of setting off an alarm as we leave a store because one item in our bags has a still-active radio-frequency identification (RFID) tag. So it's back to the cashier for some deactivation (or to security for some questioning).

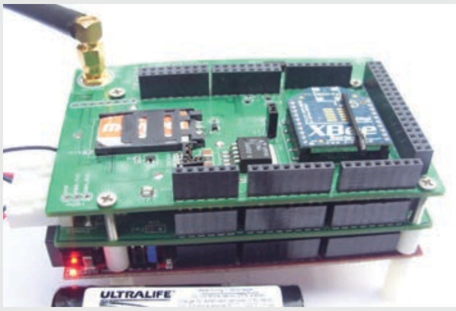
Retailers love RFID, for obvious reasons. So do other industries and governments dealing with limiting building access; tracking goods, livestock and people; collecting highway tolls and public transit fares; checking passports; finding airport baggage; managing hospital drug inventory… The list goes on and on.

RFID is a big business, and it is anticipated to grow despite concerns about privacy issues. Market researcher IDTechEx recently estimated that the RFID market—including tags, readers, and software and services for RFID labels, fobs, cards, and other form factors—will hit $9.2 billion in 2014 and increase to $30.24 billion in 2024. (Refer to http://bit.ly/O4tCKl for more details.)

So it's good timing for columnist Jeff Bachiochi's series about passive RFID tagging. Part 1 appears in this *Circuit Cellar* issue and focuses on read-only tags and transponder circuitry. It also hints at Bachiochi's unfolding RFID project (p. 66)

Other May issue highlights include DIY project articles describing an MCU-based trapdoor lift system (p. 22), a customizable approach to an ASCII interface for sending commands to a sensor tool and receiving data (p. 30), and a solar-powered home automation controller that enables household-device management and cloud connectivity to log temperature, energy use, and other data (p. 38).

In addition, our columnists explore low-power wireless data receivers (p. 50), testing and analyzing old and new batteries in a personal collection (p. 58), and designing data centers to participate in smart-grid power management (p. 46).



If you are a female engineer in search of some inspiration, read our interview with embedded systems expert Elecia White (p. 10). Also, find out why new technology means a bright future for LEDs in emissive micro-displays (p. 80).

Mary Wilson

mwilson@circuitcellar.com

US/UK
Carlo van Nistelrooy
+1 860 289 0800
c.vannistelrooy@elektor.com

ELEKTOR LABS
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

GERMANY
Ferdinand te Walvaart
+49 241 88 909-17
f.tewalvaart@elektor.de

FRANCE
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr

NETHERLANDS
Harry Baggen
+31 46 4389429
h.baggen@elektor.nl

SPAIN
Eduardo Corral
+34 91 101 93 95
e.corral@elektor.es

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

SWEDEN
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

BRAZIL
João Martins
+31 46 4389444
j.martins@elektor.com

PORTUGAL
João Martins
+31 46 4389444
j.martins@elektor.com

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

RUSSIA
Nataliya Melnikova
+7 965 395 33 36
elektor.russia@gmail.com

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

SOUTH AFRICA
Johan Dijk
+31 6 1589 4245
j.dijk@elektor.com

CHINA
Cees Baay
+86 21 6445 2811
ceesbaay@gmail.com

OUR NETWORK

circuit cellar | VOICE COIL | audioxpress | elektor

SUPPORTING COMPANIES

NOT A SUPPORTING COMPANY YET?

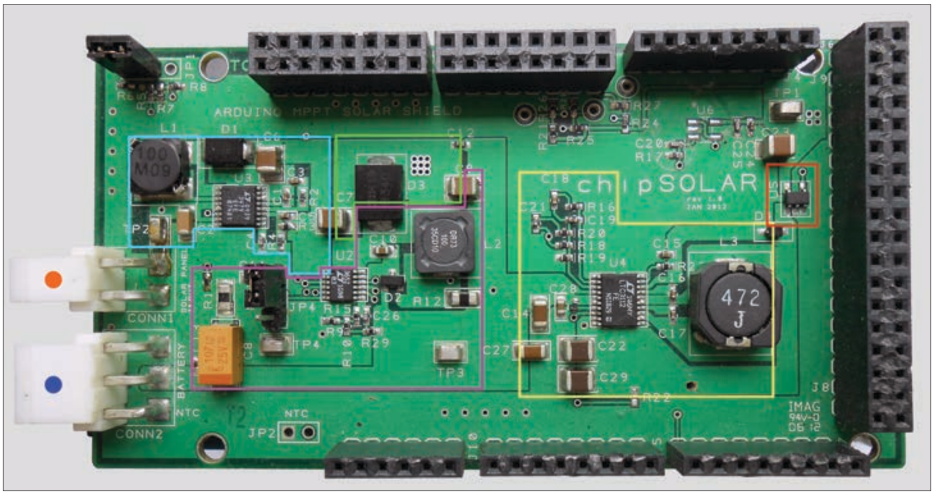
Contact Peter Wostrel (circuitcellar@smmarketing.us, Phone 978.281.7708, Fax 978.281.7706) to reserve your own space for the next edition of our members' magazine.

CONTENTS

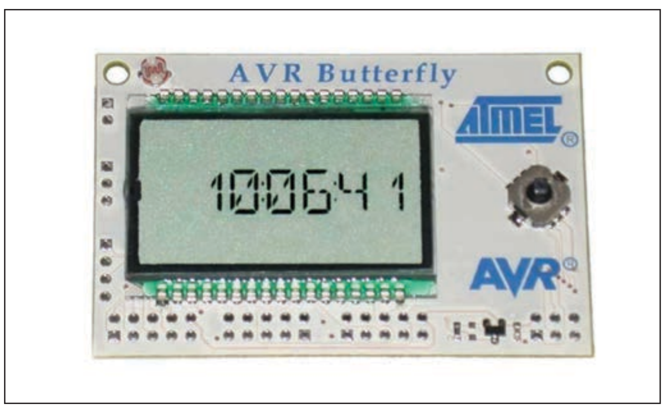
circuit cellar

MAY 2014 • ISSUE 286

MEASUREMENT & SENSORS

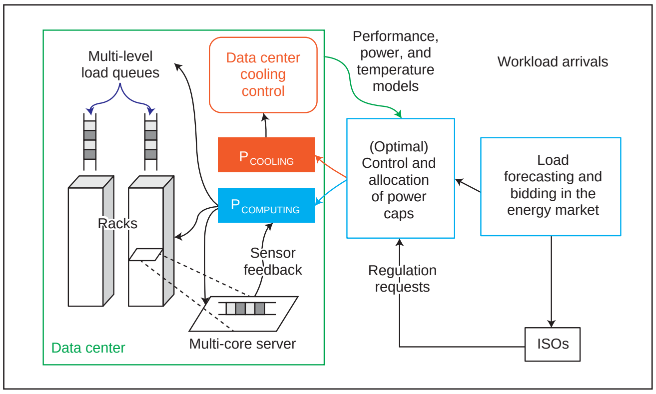


HOME AUTOMATION CONTROLLER'S SOLAR BOARD



USE ATMEL'S AVR BUTTERFLY FOR INTERFACE CODING

CONTENTS

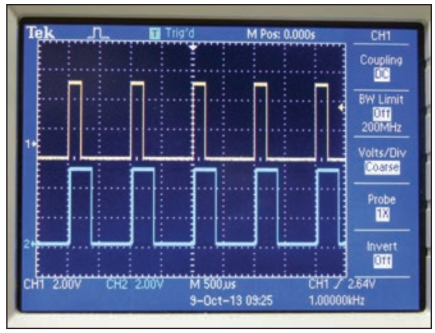


DATA CENTER DESIGN THAT COMPLEMENTS SMART-GRID DEMANDS

EMBEDDED SYSTEMS CONSULTANT'S WORKSPACE

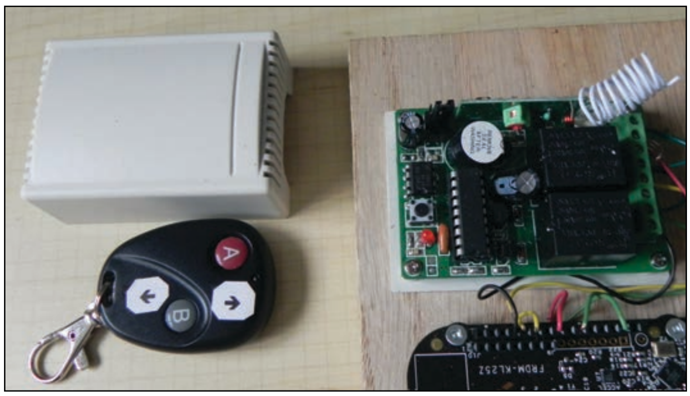
EMISSIVE LED DISPLAY CHIP

COMPUTERIZED ANALYZER AND BATTERY



LOW-POWER DATA SIGNALS

TESTS & CHALLENGES



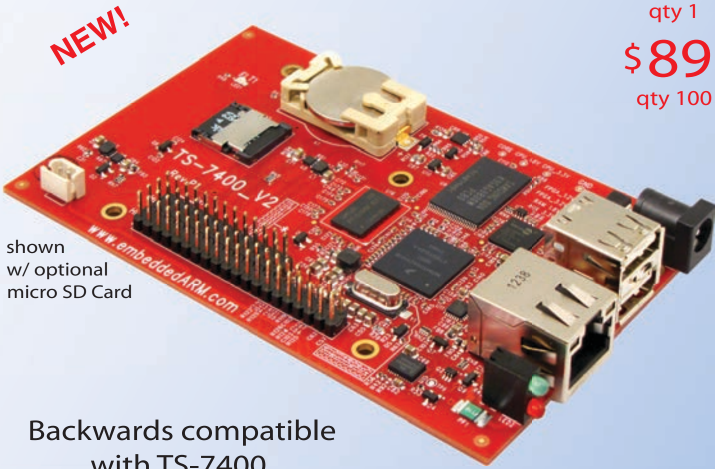
WIRELESS REMOTE CONTROL FOR A TRAPDOOR LIFT

TECH THE FUTURE

Low Power
Low Cost
High Reliability

pricing starts at
$119 qty 1
$89 qty 100

NEW!



shown w/ optional micro SD Card

Backwards compatible with TS-7400

Single Board Computers
TS-7400-V2 450MHz ARM9

Features:

- Up to 256MB RAM
- 2GB NAND Flash
- 2x CAN port
- 20x DIO
- 1x Watchdog Timer
- 2x SD Sockets
- 1x BBRTC
- 1x Temp Sensor

Benefits:

- -40 to +85C Industrial temperature range
- Backward compatible w/ TS-7400
- 2x faster, more features, reduced cost
- Low power with 120mW standby mode
- Boots quickly to your Embedded Application
- Easy development w/ Debian and Linux 2.6

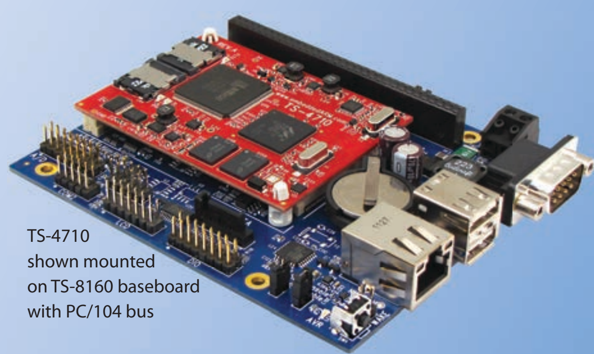
Computer-on-Modules
State of the Art Embedded Design

Our embedded Computer-on-Modules (COMs) are CPU core modules that securely connect to a carrier board. Standard functions are designed into the Computer-on-Module which is mated to your custom carrier board to enable exactly the functions your project requires. COTS carrier boards are available, or design your own carrier board for a custom solution with reduced design time and complexity.

Features can include:

- Up to 1GHz ARM w/ 512MB RAM
- Fanless, low power designs
- All parts soldered on, no moving parts
- Double Store SD for reliable storage
- User-programmable FPGAs
- -40 to +85C Industrial temperature range
- Easy development w/ Debian and Linux 2.6
- Sub-second boot to Linux

pricing starts at $139 qty 1 $99 qty 100



TS-4710 shown mounted on TS-8160 baseboard with PC/104 bus

Computer-on-Modules include:

- TS-4200: Atmel ARM9 w/ super low power
- NEW! TS-4600: 450MHZ low cost w/ 2 Ethernets
- TS-4710: Up to 1066MHz PXA168 w/ video
- NEW! TS-4712: like TS-4710 + 2 Ethernets
- NEW! TS-4720: like TS-4712 + 4GB eMMC Flash
- TS-4800: 800MHz FreeScale iMX515 w/ video

- 30 years in business
- Open Source Vision
- 8+ Year Product Lifecycle Guarantee
- Never discontinued a product
- Next day shipping for most products
- Engineers on Tech Support
- Custom configurations and designs w/ excellent pricing and turn-around time

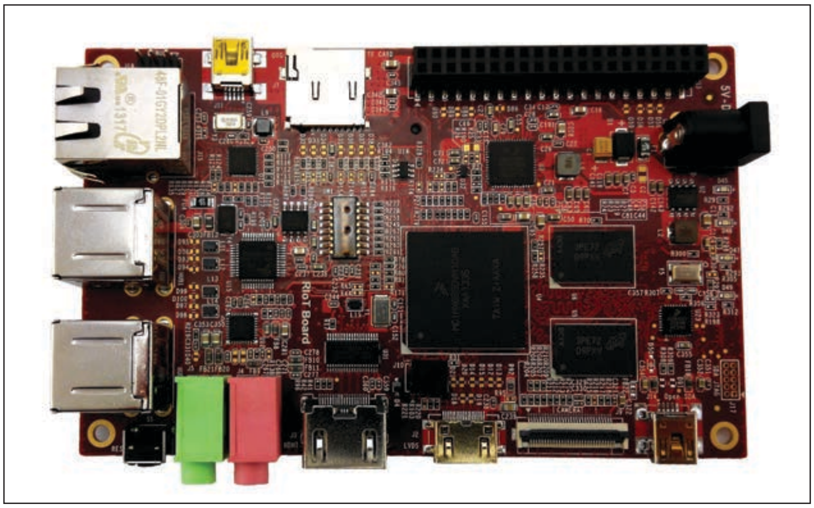
Design your solution with one of our engineers (480) 837-5200

CC WORLD

IoT FOCUS AT EELive! 2014

By CC & EIM Staff (US)

The 2014 EELive! Conference in San Jose, CA, was noticeably smaller in comparison to past embedded systems conferences. (Notable no-shows were Renesas Electronics and Cypress Semiconductor.) Nevertheless, the technologies on display were innovative and promising. According to *Circuit Cellar* and Elektor International Media (EIM) staffers who attended the event, many companies focused their presentations on two timely topics: the Internet of Things (IoT) and rapid prototyping. Below is an overview of some of the more interesting technologies on display relating to these two themes.



The RIoTboard is an open-source evaluation platform for Internet of Things (IoT) applications. (Source: element14)

MIPS-Based Newton Development Board
Imagination Technologies & Ingenic
www.imgtec.com

Imagination Technologies sees its MIPS architecture as an excellent fit for wearable tech engineers and IoT product designers. At EELive!, Imagination Technologies presented Ingenic's Newton, which is a MIPS-based development board that runs Android. It's aimed at electronics enthusiasts.

RIoTboard
RIoTboard.org

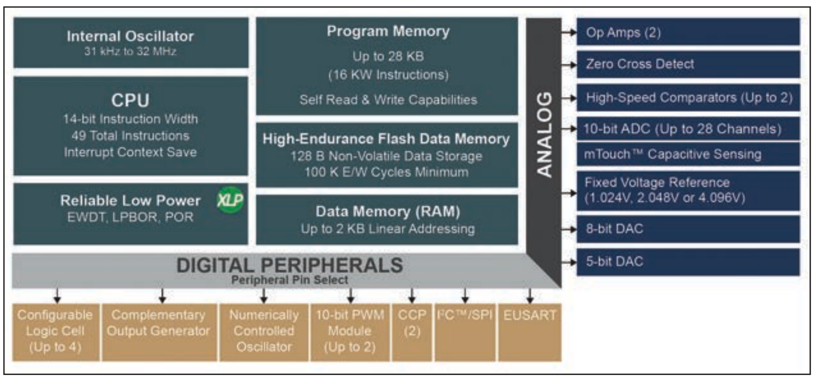
Developed by element14, in partnership with Freescale, the RIoTboard is an open-source platform based on a Freescale Semiconductor i.MX 6Solo applications processor that uses an ARM Cortex-A9 architecture. You can use it for Android and GNU/Linux development.

LAVA Internet Protocol System
LAVA Computer MFG., Inc.
www.lavaipsystem.com

At EELive!, LAVA Computer announced the LAVA Internet Protocol System, a new IoT connectivity solution delivering a powerful shortcut for IoT device developers. The LAVA IP system offers a dozen simultaneous TCP or UDP connections for electronic devices intended for IoT applications.

8-Bit PICs with Intelligent Analog and Core-Independent Peripherals
Microchip Technology, Inc.
www.microchip.com

Microchip Technology announced from EELive! the PIC16(L)F170x and PIC16(L)F171x family of 8-bit microcontrollers, which combine intelligent analog and core-independent peripherals and eXtreme Low Power (XLP) technology. These are the first PIC16 microcontrollers with Peripheral Pin Select, which is a pin-mapping feature for designating the pinout of peripheral functions. (Visit http://bit.ly/1oJrOHE online for more details and a video.)



Microchip Technology's PIC16F170X/171X family features core independent peripherals, such as the configurable logic cell (CLC), complementary output generator (COG), and numerically controlled oscillator (NCO). (Source: Microchip Technology)


mouser.com
The Newest Products for Your Newest Designs®
NEWEST PRODUCTS
More new products
More new technologies
More added every day
Authorized distributor of semiconductors
and electronic components for design engineers.
MOUSER ELECTRONICS
Mouser and Mouser Electronics are registered trademarks of Mouser Electronics, Inc. Other products, logos, and company names mentioned herein, may be trademarks of their respective owners.

QUESTIONS & ANSWERS

Embedded Systems Consultant

An Interview with Elecia White

Elecia White is an embedded systems engineer, consultant, author, and innovator. She has worked on a variety of projects: DNA scanners, health-care monitors, learning toys, and fingerprint recognition.—Nan Price, Associate Editor

COMMUNITY

NAN: Tell us about your company Logical Elegance (www.logicalelegance.com). When and why did you start the company? What types of services do you provide?

ELECIA: Logical Elegance is a small San Jose, CA-based consulting firm specializing in embedded systems. We do system analysis, architecture, and software implementation for a variety of devices.

I started the company in 2004, after leaving a job I liked for a job that turned out to be horrible. Afterward, I wasn't ready to commit to another full-time job; I wanted to dip my toe in before becoming permanent again.

I did eventually take another full-time job at ShotSpotter, where I made a gunshot location system. Logical Elegance continued when my husband, Chris, took it over. After ShotSpotter, I returned to join him. While we have incorporated and may take on a summer intern, for the most part Logical Elegance is only my husband and me.

I like consulting, it lets me balance my life better with my career. It also gives me time to work on my own projects: writing a book and articles, playing with new devices, learning new technologies. On the other hand, I could not have started consulting without spending some time at traditional companies. Almost all of our work comes from people we've worked with in the past, either people we met at companies where we worked full time or people who worked for past clients.

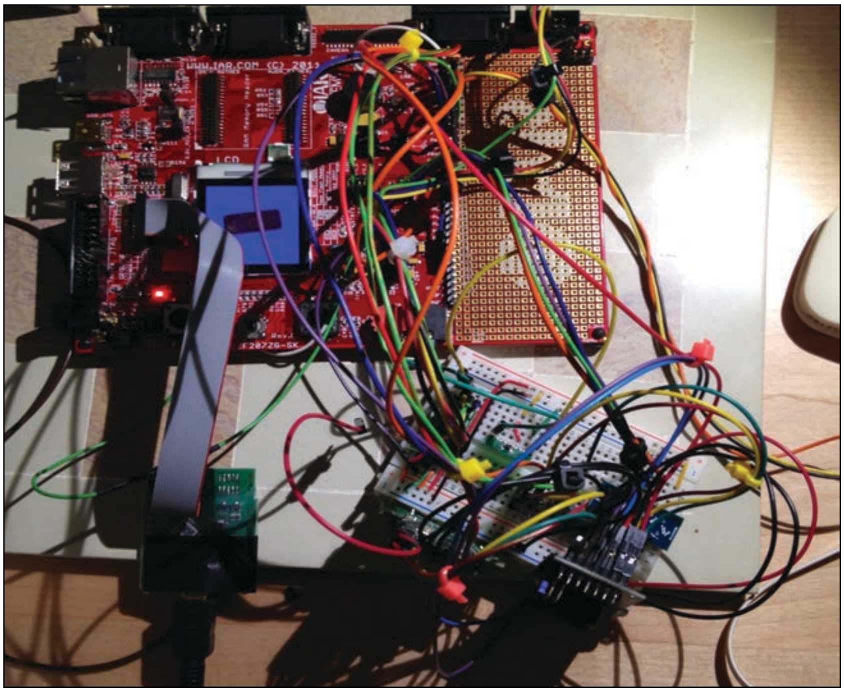
Here is Elecia's home lab bench. She conveniently provided notes.

NAN: Logical Elegance has a diverse portfolio. Your clients have ranged from Cisco Systems to LeapFrog Enterprises. Tell us about some of your more interesting projects.

ELECIA: We are incredibly fortunate that embedded systems are diverse, yet based on similar bedrock. Once you can work with control loops and signal processing, the applications are endless. Understanding methodologies for concepts such as state machines, interrupts, circular buffers, and working with peripherals allows us to put the building blocks together a different way to suit a particular product's need.

For example, for a while there, it seemed like some of my early work learning how to optimize systems to make big algorithms work on little processors would fall to the depths of unnecessary knowledge. Processors kept getting more and more powerful. However, as I work on wearables, with their need to optimize cycles to extend their battery life, it all is relevant again.

We've had many interesting projects. Chris is an expert in optical coherence tomography (OCT). Imagine a camera that can go on the end of a catheter to help a doctor remove plaque from a clogged artery or to aid in eye surgery. Chris is also the networking expert. He works on networking protocols such as Locator/ID Separation Protocol (LISP) and multicast.



Elecia says one of the great things about the explosion of boards and kits available is being able to quickly build a system. However, she explains, once the components work together, it is time to spin a board. (This system may be past that point.)

I'm currently working for a tiny company that hopes to build an exoskeleton to help stroke patients relearn how to walk. I am incredibly enthusiastic about both the application and the technology.

That has been a theme in my career, which is how I've got this list of awesome things I've worked on: DNA scanners, race cars and airplanes, children's toys, and a gunshot location system. The things I leave off the list are more difficult to describe but no less interesting to have worked on: a chemical database that used hydrophobicity to model uptake rates, a medical device for the operating room and ICU, and methods for deterring fraud using fingerprint recognition on a credit card.

Elecia was a judge at the MEMS Elevator Pitch Session at the 2013 MEMS Executive Congress in Napa, CA.

In the last few years, Chris and I have both worked for Fitbit on different projects. If you have a One pedometer, you have some of my bits in your pocket.

The feeling of people using my code is wonderful. I get a big kick seeing my products on store shelves. I enjoyed working with Fitbit. When I started, it was a small company expanding its market; definitely the underdog. Now it is a success story for the entire microelectromechanical systems (MEMS) industry.

Not everything is rosy all the time though. For one start-up, the algorithms were neat, the people were great, and the technology was a little clunky but still interesting. However, the client failed and didn't pay me (and a bunch of other people).

When I started consulting, I asked a more experienced friend about the most important part. I expected to hear that I'd have to make myself more extroverted, that I'd have to be able to find more contracts and do marketing, and that I'd be involved in the drudgery of accounting. The answer I got was the truth: the most important part of consulting is accounts receivable. Working for myself—especially with small companies—is great fun, but there is a risk.

NAN: How did you get from "Point A" to Logical Elegance?

ELECIA: "Point A" was Harvey Mudd College in Claremont, CA. While there, I worked as a UNIX system administrator, then later worked with a chemistry professor on his computational software. After graduation, I went to Hewlett Packard (HP), doing standard software, then a little management. I was lured to another division to do embedded software (though we called it firmware).

Next, a start-up let me learn how to be a tech lead and architect in the standard start-up sink-or-swim methodology. A mid-size company gave me exposure to consumer products and a taste for seeing my devices on retailer's shelves.

From there, I tried out consulting, learned to run a small business, and wrote a *Circuit Cellar Ink* article "Open Source Code Guide" (Issue 175, 2005). I joined another tiny start-up where I did embedded software, architecture, management, and even directorship before burning out. Now, I'm happy to be an embedded software consultant, author, and podcast host.

NAN: You wrote *Making Embedded Systems: Design Patterns for Great Software* (O'Reilly Media, 2011). What can readers expect to learn from the book?

QUESTIONS & ANSWERS

ELECIA: While having some industry experience in hardware or software will make my book easier to understand, it is also suitable for a computer science or electrical engineering college student.

It is a technical book for software engineers who want to get closer to the hardware or electrical engineers who want to write good software. It covers many types of embedded information: hardware, software design patterns, interview questions, and a lot of real-world wisdom about shipping products.

Making Embedded Systems is intended for engineers who are in transition: the hardware engineer who ends up writing software or the software engineer who suddenly needs to understand how the embedded world is different from pure software.

Unfortunately, most college degrees are either computer science or electrical engineering. Neither truly prepares for the half-and-half world of an embedded software engineer. Computer science teaches algorithms and software design methodology. Electrical engineering misses both of those topics but provides a practical tool kit for doing low-level development on small processors. Whichever collegiate (or early career) path, an embedded software engineer needs to have familiarity with both.

I did a non-traditional major that was a combination of computer science and engineering systems. I was prepared for all sorts of math (e.g., control systems and signal processing) and plenty of programming. All in all, I learned about half of the skills I needed to do firmware. I was never quite sure what was correct and what I was making up as I went along.

As a manager, I found most everyone was in the same boat: solid foundations on one side and shaky stilts on the other. The goal of the book is to take whichever foundation you have and cantilever a good groundwork to the other half. It shouldn't be 100% new information. In addition to the information presented, I'm hoping most people walk away with more confidence about what they know (and what they don't know).

NAN: How long have you been designing embedded systems? When did you become interested?

ELECIA: I was a software engineer at the NetServer division at HP. I kept doing lower-level software, drivers mostly, but for big OSes: WinNT, OS/2, Novell NetWare, and SCO UNIX (a list that dates my time there).

HP kept trying to put me in management but I wasn't ready for that path, so I went to HP Labs's newly spun-out HP BioScience to make DNA scanners, figuring the application would be more interesting. I had no idea.

I lit a board on fire on my very first day as an embedded software engineer. Soon after, a motor moved because my code told it to. I was hooked. That edge of software, where the software touches the physical, captured my imagination and I've never looked back.

"I lit a board on fire on my first day as an embedded software engineer. Soon after, a motor moved because my code told it to. I was hooked."

NAN: Tell us about the first embedded system you designed. Where were you at the time? What did you learn from the project?

ELECIA: Wow, this one is hard. The first embedded system I designed depends on your definition of "designed." Going from designing subsystems to the whole system to the whole product was a very gradual shift, coinciding with going to smaller and smaller companies until suddenly I was part of the team not only choosing processors but choosing users as well.

After I left the cushy world of HP Labs with a team of firmware engineers, several electrical engineers, and a large team of software engineers who were willing to help design and debug, I went to a start-up with fewer than 50 people. There was no electrical engineer (except for the EE who followed from HP). There was a brilliant algorithms guy but his software skills were more MATLAB-based than embedded C. I was the only software/firmware engineer. This was the sort of company that didn't have source version control (until after my first day). It was terrifying being on my own and working without a net.

I recently did a podcast about how to deal with code problems that feel insurmountable. While the examples were all from recent work, the memories of how to push through when there is no one else who can help came from this job.

NAN: Are you currently working on or planning any projects?

ELECIA: I have a few personal projects I'm working on: a T-shirt that monitors my posture and a stuffed animal that sends me a "check on Lois" text if an elderly neighbor doesn't pat it every day. These don't get nearly enough of my

QUESTIONS & ANSWERS

Elecia is shown recording a *Making Embedded Systems* episode with the founders of electronics educational start-up Light Up. From left to right: Elecia's husband and producer Christopher White, host Elecia White, and guests Josh Chan and Tarun Pondicherry.

attention these days as I've been very focused on my podcast: *Making Embedded Systems* on iTunes, Instacast, Stitcher, or direct from http://embedded.fm.

The podcast started as a way to learn something new. I was going to do a half-dozen shows so I could understand how recording worked. It was a replacement for my normal community center classes on stained glass, soldering, clay, hula hooping, laser cutting, woodshop, bookbinding, and so forth.

However, we're way beyond six shows and I find I quite enjoy it. I like engineering and building things. I want other people to come and play in this lovely sandbox. I do the show because people continue to share their passion, enthusiasm, amusement, happiness, spark of ingenuity, whatever it is, with me.

To sum up why I do a podcast, in order of importance: to talk to people who love their jobs, to share my passion for engineering, to promote the visibility of women in engineering, and to advertise for Logical Elegance (this reason is just in case our accountant reads this since we keep writing off expenses).

"I like engineering and building things. I want other people to come and play in this sandbox."

NAN: What are your go-to embedded platforms? Do you have favorites, or do you use a variety of different products?

ELECIA: I suppose I do have favorites but I have *a lot* of favorites. At any given time, my current favorite is the one that is sitting on my desk. (Hint!)

I love Arduino although I don't use it much except to get other people excited. I appreciate that at the heart of this beginner's board (and development system) is a wonderful, useful processor that I'm happy to work on.

I like having a few Arduino boards around, figuring that I can always get rid of the bootloader and use the Atmel ATmega328 on its own. In the meantime, I can give them to people who have an idea they want to try out.

For beginners, I think mbed's boards are the next step after Arduino. I like them but they still have training wheels: nice, whizzy training wheels but still training wheels. I have a few of those around for when friends' projects grow out of Arduinos. While I've used them for my own projects, their price precludes the small-scale production I usually want to do.

Professionally, I spend a lot of time with Cortex-M3s, especially those from STMicroelectronics and NXP Semiconductors. They seem ubiquitous right now. These are processors that are definitely big enough to run an RTOS but small enough that you don't have to. I keep hearing that Cortex-M0s are coming but the price-to-performance-to-power ratio means my clients keep going to the M3s.

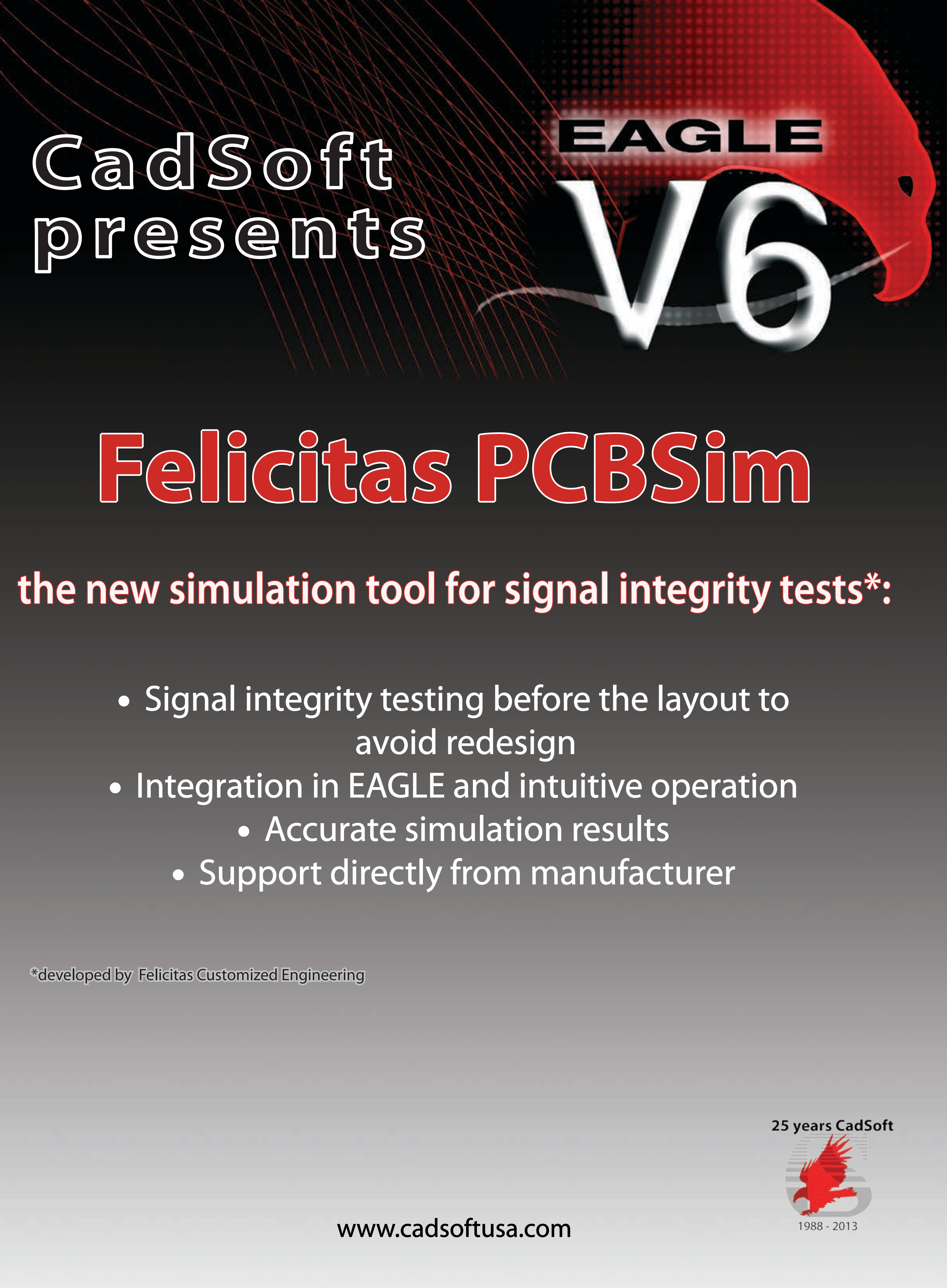
Finally, I suppose I'll always have a soft spot for Texas Instruments's C2000 line, which is currently in the Piccolo and Delfino incarnations. The 16-bit byte is horrible (especially if you need to port code to another processor), but somehow everything else about the DSP does just what I want. Although, it may not be about the processor itself: if I'm using a DSP, I must be doing something mathy and I like math.

NAN: Do you have any predictions for upcoming "hot topics?"

ELECIA: I'm most excited about health monitoring. I'm surprised that *Star Trek* and other science fiction sources got tricorders right but missed the constant health monitoring we are heading toward with the rise of wearables and the interest in quantified self.

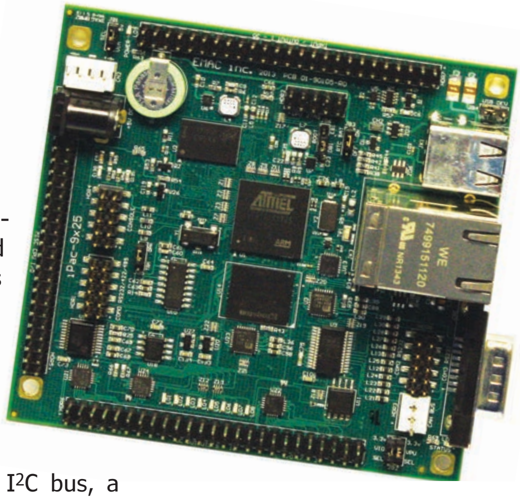
I'm most concerned about connectivity. The Internet of Things (IoT) is definitely coming, but many of these devices seem to be more about applying technology to any device that can stand the price hit, whether it makes sense or not.

Worse, the methods for getting devices connected keeps fracturing as the drive toward low-cost and high functionality leads the industry in different directions. And even worse, the ongoing battle between security and ease of use manages to give us things that are neither usable nor secure. There isn't a good solution (yet). To make progress we need to consider the application, the user, and what they need instead of applying what we have and hoping for the best.


CadSoft
presents
EAGLE
V6
Felicitas PCBSim
the new simulation tool for signal integrity tests*:
• Signal integrity testing before the layout to avoid redesign
• Integration in EAGLE and intuitive operation
• Accurate simulation results
• Support directly from manufacturer
*developed by Felicitas Customized Engineering
25 years CadSoft
1988 - 2013
www.cadsoftusa.com

PRODUCT NEWS

INDUSTRIAL TEMPERATURE SBCs



The **iPAC-9X25** embedded single-board computer (SBC) is based on Atmel's AT91SAM9X25 microprocessor. It is well suited for industrial temperature embedded data acquisition and control applications.

This web-enabled microcontroller can run an embedded server and display the current-monitored or logged data. The web connection is available via two 10/100 Base-T Ethernet ports or 802.11 Wi-Fi networking. The iPAC-9X25's connectors are brought out as headers on a board.

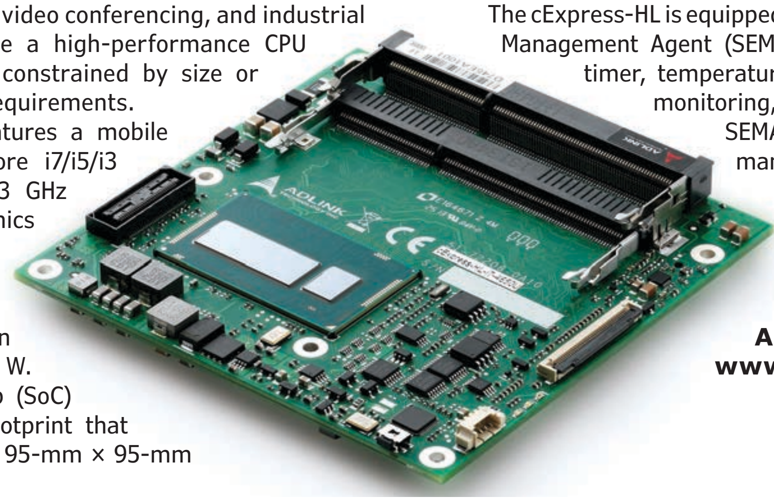
The SBC has a –40°C-to-85°C industrial temperature range and utilizes 4 GB of eMMC flash memory, 16 MB of serial data flash memory (for boot), and 128 MB of double data rate (DDR) RAM. Its 3.54" × 3.77" footprint is the same as a standard PC/104 module.

The iPAC 9X25 features two USB 2.0 host ports and one USB device port. The board has seven channels of 12-bit audio/digital (0 to 3.3 V) and an internal real-time clock/calendar (RTCC) with battery backup. The embedded SBC also includes eight high-drive open-collector dedicated digital output lines with configurable voltage tolerance, two PWM I/O lines, five serial I/O lines (I^2S), five SPI lines (two SPI CS), an I^2C bus, a CAN bus, a micro-SD socket, external Reset button capabilities, and power and status LEDs.

The iPac-9X25 costs **$198**.

EMAC, Inc.
www.emacinc.com

COMPACT COMPUTER-ON-MODULE



The **cExpress-HL** utilizes an Intel Core processor (formerly known as Haswell-ULT) to provide a compact, high-performance computer-on-module (COM) solution. The cExpress-HL is well suited for embedded systems in medical, digital signage, gaming, video conferencing, and industrial automation that require a high-performance CPU and graphics, but are constrained by size or thermal management requirements.

The cExpress-HL features a mobile 4th Generation Intel Core i7/i5/i3 processor at 1.7 to 3.3 GHz with Intel HD Graphics 5000 (GT3). The COM delivers high graphics performance while still keeping thermal design power (TDP) below 15 W. Intel's system-on-a-chip (SoC) solution has a small footprint that enables it to fit onto the 95-mm × 95-mm COM.0 R2.0 Type 6. The cExpress-HL provides rich I/O and wide-bandwidth data throughput, including three independent displays, four PCIe x1 or one PCIe x4 (Gen2), four SATA 6 Gb/s, two USB 3.0 ports, and six USB 2.0 ports.

The cExpress-HL is equipped with ADLINK's Smart Embedded Management Agent (SEMA), which includes a watchdog timer, temperature and other board information monitoring, and fail-safe BIOS support. SEMA enables users to monitor and manage stand-alone, connected, or remote systems through a cloud-based interface.

Contact ADLINK Technology for pricing.

ADLINK Technology, Inc.
www.adlinktech.com

DYNAMIC EFFICIENCY MICROCONTROLLERS

The **STM32F401** Dynamic Efficiency microcontrollers extend battery life and support innovative new features in mobile phones, tablets, and smart watches. The microcontrollers help manage microelectromechanical systems (MEMS) sensors in smart-connected devices and are designed for Internet of Things (IoT) applications and fieldbus-powered industrial equipment.

The Dynamic Efficiency microcontrollers include an ART accelerator, a prefetch queue, and a branch cache. This enables zero-wait-state execution from flash memory, which boosts performance.

The STM32F401 microcontrollers integrate up to 512 KB of flash memory and 96 KB SRAM in a 3.06-mm × 3.06-mm chip-scale package and feature a 9-µA at 1.8-V Stop mode current. The devices' peripherals include three 1-Mbps I^2C ports, three USARTs, four SPI ports, two full-duplex I^2S audio interfaces, a USB 2.0 On The Go (OTG) full-speed interface, a Secure Digital Input Output (SDIO) interface, a 16-channel ADC, and up to 10 timers.

Pricing for the STM32F401 microcontrollers starts at **$2.88** in 10,000-unit quantities.

STMicroelectronics
www.st.com

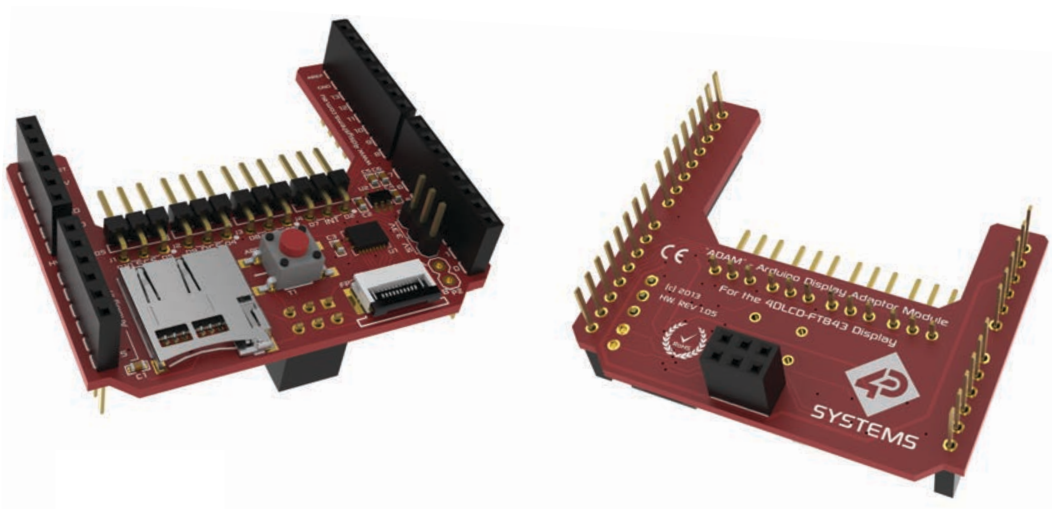
HMI DEVELOPMENT ON INTELLIGENT DISPLAYS

4D Systems and Future Technology Devices International (FTDI, a.k.a., FTDI Chip) recently introduced an intelligent display solution called the **4DLCD-FT843**. The solution incorporates FTDI Chip's FT800 Embedded Video Engine (EVE) with the subsequent introduction of two additional products. This combined product gives design engineers a foundation to quickly and easily construct human-machine interfaces (HMIs).

The first of these products is the ADAM (Arduino Display Adaptor Module). This 47.5-mm × 53.4-mm Arduino-compatible shield permits communication between the Arduino via the SPI. The shield is suitable for use with Arduino Uno, Due, Duemilanove, Leonardo, Mega 1280/2560, and Pro 5V. The shield's micro-SD card provides the Arduino-based display system with ample data storage.

The 4DLCD-FT843 solution can use the micro-SD card to retrieve objects (e.g., images, sounds, fonts, etc.). Drawing power from the Arduino's 5-V bus, the ADAM regulates the 4DLCD-FT843's supply to 3.3 V. The FT800 EVE controller can handle many of the graphics functions that would otherwise need to be managed by the Arduino.

The ADAM shield is complemented by The 4DLCD-FT843-Breakout board. This simple 12-mm × 26.5-mm footprint breakout module enables the 4DLCD-FT843 to be attached to a general host or breadboard for prototyping purposes. It features a 10-way flexible printed circuit (FPC) connection for attachment with the 4DLCD-FT843 along with a 10-way, 2.54-mm pitch male pin header that enables it to directly connect to the host board. Both products support a –10°C-to-70°C operational temperature range.



The EVE-driven 4DLCD-FT843 has a 4.3″ thin-film transistor video graphics array (VGA) display with a four-wire resistive touchscreen. It features a 64-voice polyphonic sound synthesizer, a mono PWM audio output, a programmable interrupt controller, a PWM dimming controller for the display's backlight, and a flexible ribbon connector.

Contact 4D Systems or FTDI Chip for pricing.

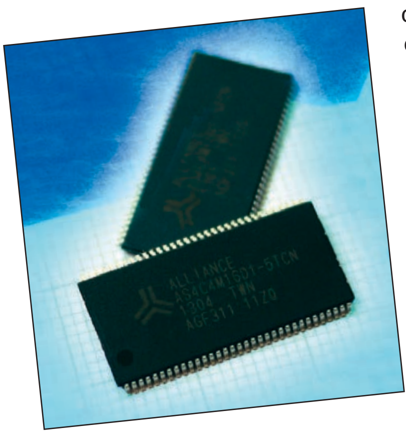
4D Systems
www.4dsystems.com.au

Future Technology Devices International, Ltd. (FTDI, a.k.a., FTDI Chip)
www.ftdichip.com

HIGH-SPEED CMOS DDR SDRAMs

The **AS4C4M16D1-5TIN**, the **AS4C8M16D1-5TIN**, the **AS4C16M16D1-5TIN**, and the **AS4C32M16D1-5TIN** are CMOS double data rate synchronous DRAMs (DDR SDRAMs). The devices feature densities of 64 MB (AS4C4M16D1-5TIN), 128 MB (AS4C8M16D1-5TIN), 256 MB (AS4C16M16D1-5TIN), and 512 MB (AS4C32M16D1-5TIN) with a –40°C-to-85°C industrial temperature range.

The high-speed DDR SDRAMs provide reliable drop-in, pin-for-pin-compatible replacements for industrial, medical, telecommunications, and communications products requiring high memory bandwidth. The devices are well-suited for high performance in PC applications. Internally configured as four banks of 1M, 2M, 4M, or 8M word × 16 bits with a synchronous interface, the DDR SDRAMs operate from a single 2.5-V (±0.2 V) power supply and are lead- and halogen-free.



The AS4C4M16D1-5TIN, the AS4C8M16D1-5TIN, the AS4C16M16D1-5TIN, and the AS4C32M16D1-5TIN feature a 200-MHz clock rate. The DDR SDRAMs provide programmable read or write burst lengths of 2, 4, or 8. An auto pre-charge function provides a self-timed row pre-charge initiated at the end of a burst sequence.

Easy-to-use refresh functions include auto- or self-refresh. A programmable mode register enables the system to choose a suitable mode for maximum performance.

Pricing for the AS4C4M16D1-5TIN, the AS4C8M16D1-5TIN, the AS4C16M16D1-5TIN, and the AS4C32M16D1-5TIN starts at **$0.90** per piece.

Alliance Memory, Inc.
www.alliancememory.com

PRODUCT NEWS

ARM mbed PLATFORM FOR BLUETOOTH SMART APPLICATIONS

The **nRF51822-mKIT** simplifies and accelerates the prototyping process for Bluetooth Smart sensors connecting to the Internet of Things (IoT). The platform is designed for fast, easy, and flexible development of Bluetooth Smart applications.

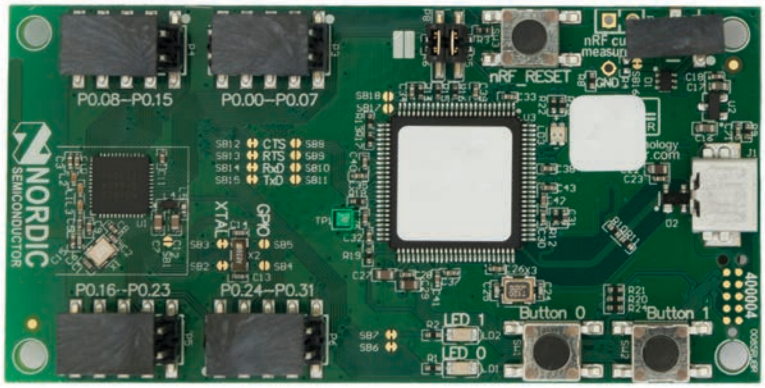
The nRF51822 system-on-a-chip (SoC) combines a Bluetooth v4.1-compliant 2.4-GHz multiprotocol radio with an ARM Cortex-M0 CPU core on a single chip optimized for ultra-low-power operation. The SoC simplifies and accelerates the prototyping process for Bluetooth Smart sensors connecting to the IoT.

The nRF51822-mKIT's features include a Bluetooth Smart API, 31 pin-assignable general-purpose input/output (GPIO), a CMSIS-DAP debugger, Programmable Peripheral Interconnect (PPI), and the ability to run from a single 2032 coin-cell battery.

Through mbed, the kit is supported by a cloud-based approach to writing code, adding libraries, and compiling firmware. A lightweight online IDE operates on all popular browsers running on Windows, Mac OSX, iOS, Android, and Linux OSes. Developers can use the kit to access a cloud-based ARM RealView Development Suite (RVDS) 4.1 compiler that optimizes code size and performance.

The nRF51822-mKIT costs **$59.95.**

Nordic Semiconductor ASA
www.nordicsemi.com



THREE-AXIS MAGNETOMETER SENSOR

The **IST8301C** is a single-chip three-axis digital magnetometer sensor that is housed in a 1-mm × 2.5-mm × 2.5-mm, 12-pin ball-grid array (BGA) package. The integrated chip includes three-axis magnetic sensors with an ASIC controller.

The IST8301C outputs 13-bit data over a ±1,000-µT magnetic field range in a fast-mode 400-kHz I^2C digital output. The compact form factor is easily surface mounted and is well suited for high-volume production consumer electronics, navigation systems, and magnetometers.

The IST8301C embeds 32 slots of 16-bit first in, first out (FIFO) data for each of three output channels X, Y, Z. Since the host processor does not need to continuously read data from a sensor, the FIFO's "wake up only as needed" operation enables consistent system-power saving. Functioning on 2.4 V with a 10-uA standby current and full operation at 300 uA, increased battery life can be attained in many portable applications.

The IST8301C offers a ±1° heading accuracy and a ±10-gauss magnetic field range. The sensor includes anti-offset and anti-temperature to help eliminate errors caused by temperature and factory mismatch.

Contact Saelig for pricing.

Saelig Co., Inc.
www.saelig.com

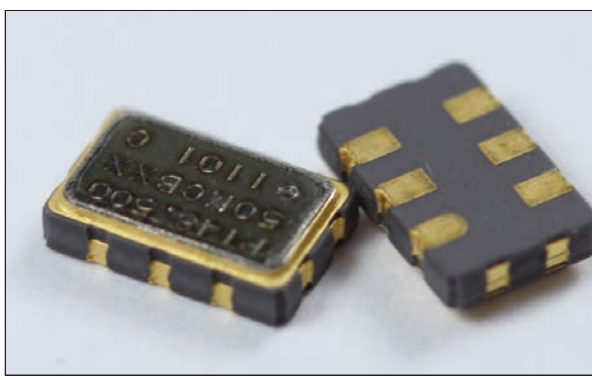
MINIATURE PECL AND LVDS OSCILLATORS

The PDI Model **LV5** and **PE5** series of oscillators provide precision timing in a 3.2-mm × 5-mm ceramic hermetically sealed package. The LV5 is a low-voltage differential signaling (LVDS) oscillator. The PE5 is a PHP Extension Community Library (PECL) oscillator.

These high-performance clock oscillators offer low integrated phase jitter (0.2 pS for the LV5 and 0.3 pS for the PE5). They are available in frequencies up to 200 MHz and feature a –40°C-to-85°C industrial temperature range. Stabilities can be held down to ±25 ppm (depending on the temperature range).

Contact Precision Devices for pricing.

Precision Devices, Inc.
www.pdixtal.com



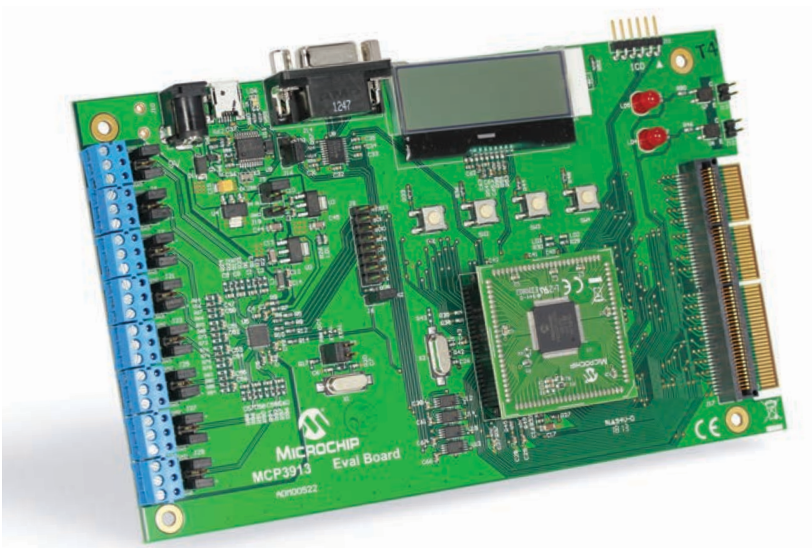
ENERGY-MEASUREMENT AFEs

The **MCP3913** and the **MCP3914** are Microchip Technology's next-generation family of energy-measurement analog front ends (AFEs). The AFEs integrate six and eight 24-bit, delta-sigma ADCs, respectively, with 94.5-dB signal-to-noise and distortion ratio (SINAD), –106.5-dB total harmonic distortion (THD), and 112-dB Spurious-Free Dynamic Range (SFDR) for high-accuracy signal acquisition and higher-performing end products.

The MCP3914's two extra ADCs enable the monitoring of more sensors with one chip. The programmable data rate of up to 125 ksps with low-power modes enables designers to scale down for better power consumption or to use higher data rates for advanced signal analysis (e.g., calculating harmonic content).

The MCP3913 and the MCP3914 AFEs improve application performance and provide flexibility to adjust the data rate to optimize each application's rate of performance vs power consumption. The AFEs feature a CRC-16 checksum and register-map lock, for increased robustness.

The MCP3913 and the MCP3914 AFEs cost **$3.04** each in 5,000-unit quantities. Microchip Technology also announced the MCP3913 Evaluation Board and the MCP3914 Evaluation Board, two new tools to aid in the development of energy systems using the MCP3913 and the MCP3914 AFEs. Both evaluation boards cost **$99.99**.



Microchip Technology, Inc.
www.microchip.com

CLIENT PROFILE

Lemos Int'l Technology Co., Inc.

www.lemosint.com

580 Maple Avenue, Suite 1, Barrington, RI 02806

CONTACT: sales@lemosint.com

EMBEDDED PRODUCTS: Lemos International distributes, designs, and manufactures off-the-shelf and custom RF components including long- and short-range ultra-high frequency (UHF), very high frequency (VHF), and data radios. The company offers high-performance OEM-style embedded RF modules and industrial Bluetooth, ZigBee, Wi-Fi, and IC semiconductors for industrial, remote control, home automation, oil and gas, supervisory control and data acquisition (SCADA), medical, commercial, and military applications. Lemos can help you design and implement your wireless products.

FEATURED PRODUCT: The license-free SHX1-151-5-12k5-MURS (multi-use radio service) is a small multichannel 25-kHz narrow-band VHF transceiver with up to 500-mW RF power output. The transceiver is designed for 144-MHz US band amateur, 151-MHz US MURS, and European 169-MHz band high-power applications.

The fully screened, low-profile SHX1 features a high-performance double superheterodyne phase-locked loop (PLL) synthesizer and data rates up to 5 kbps for a standard module, with a usable range over 5 km. The transceiver's feature-rich interface includes received signal strength indicator (RSSI), analog, and digital baseband. It incorporates a 1,200-bps modem and is reprogrammable via an RS-232 interface.

The SHX1 is designed for applications including amateur radio, Adaptive Multi-Rate (AMR), MURS units, industrial telemetry and telecommand, high-end security systems, Automatic Packet Reporting System (APRS), vehicle data up/download, and real options valuations (ROV) and machinery controls.

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.


PRINTED CIRCUIT BOARDS
ARE YOU READY FOR THE ACCUTRACE CHALLENGE?
TRY BEATING OUR PCB PRICES.
If you do, we will match it AND give you $100 off towards your next order!
Visit www.PCB4u.com and see why our prices can not be beaten.
• From same day quick turn prototype to production in under 10 days
• Full CAD and CAM review plus design rule check on ALL Gerber files.
• Materials: Fr4, Rigid, Flex, Metal Core (Aluminum), Polymide, Rogers, Isola, etc.
• HDI Capabilities: Blind/Buried Mocrovias, 10+N+10, Via-in-Pad Technology,Sequential Lamination, Any Layer, etc.
• Our HDI Advantage: Direct Laser Drilling, Plasma De-Smear Technology, Laser Micro via Conductive Plate Shut, etc.
Accutrace inc.
www.PCB4u.com • sales@PCB4u.com

Remote-Control Powered Trapdoor Lift

You can build a remote-controlled power lift to operate a manually opening trapdoor. This project uses a remote-control module, a relay board, and a microcontroller board to create a powered lift system.

By William Wachsmann (Canada)

This multi-disciplinary project includes mechanical, electronic, and software components and uses inexpensive off-the-shelf modules to provide the electrical functions for a powered trapdoor lift system. I started the project as a way to make a powered lift system for a trapdoor access to my home's basement.

Lifting the trapdoor was hard on my wife's back. If her arms were full when she came upstairs and the door was open, she had to twist her body to release the mechanical latching mechanism while simultaneously stopping the door from falling on her head. The large 33.5″ × 48″ trapdoor is in a main hallway. When open, it blocks access between the office and the rest of the house.

INS AND OUTS

I used a screw lift from an old treadmill. It has a 48-VDC motor that draws about 1 A under a 100-lb load. The screw mechanism has a 6″ travel distance. Built-in limit switches shut off the motor when the screw reaches the end of its travel in each direction. The screw's length is nominally 30″ when closed and 36″ when open. The length can be adjusted slightly by rotating the screw by hand, but the overall travel distance is still 6″.

A simple switch would have sufficed to control the screw lift's DC motor, but I wanted it to be remotely controlled so the trapdoor could be raised and lowered from anywhere upstairs and from the basement. When someone is in the basement with the trapdoor closed there is no visible way for them to know if the door is obstructed. Initially, I was going to install a warning beeper that the door was opening, but that wouldn't help if an inanimate object (e.g., a bag of groceries) was on top of the door. I needed to incorporate some form of sensing mechanism into the design.

THE MECHANICS

I needed a levered system design that used a pivoted bar and the motorized screw lift. The lift also had to enable the door to be manually opened in case of a power failure.

I used IMSI/Design's TurboCAD program for the mechanical design. By using CAD software, I could experiment with the pivot position, moment arms, and torque requirements to meet the mechanical constraints imposed by the screw lift and the trapdoor's size and weight.

Figure 1 shows a diagram of the trapdoor, which is hinged on the left. The opposite side of the door exerts a 15.2-lb downward force. This means the torque (force × distance) required to open the door is 509.2 in-lbs. The pivot arm in red is the position when the door is closed. The blue pivot arm shows the position when the door is open to an 80° angle.

To keep within the 6″ lift constraint, I used a 4.25″ moment arm to pull down on the pivot arm. This left me with the force required to initially lift the door at 119.5 lb. Also this did not include the added torque due to the pivot arm's weight.

After mulling this over for a couple of days (and nights) I had an idea. I realized that 119.5 lb is only needed when the door is completely closed. As the door opens, the torque requirement lessens. I incorporated a heavy spring (see **Photo 1**). When the door is closed the spring extension provides an additional downward force of about 35 lb. This is enough to lessen the load on the screw lift and to compensate for the pivot arm's additional 2.2 lb. Using a screw lift meant the arm would not spring up if the door was manually opened.

I used an angle iron for the pivot arm. It is 28″ long because it had to push up on the door to the right of the door's center of gravity at 16.75″ without adding too much additional torque. The roller is the type used as feet on beds. I used an arbor and 0.75″-diameter bolt through the floor joist for the pivot (see **Photo 2**).

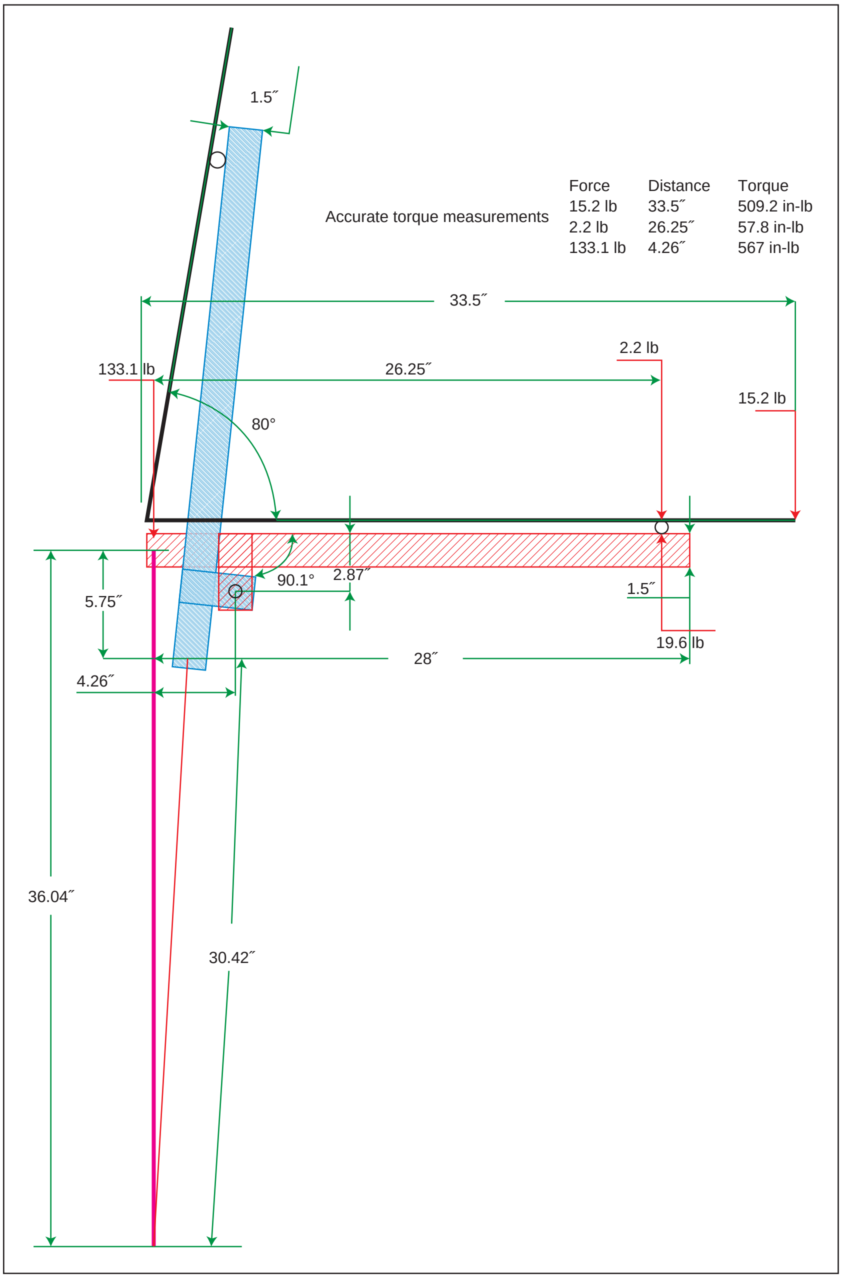


FIGURE 1
This diagram represents the trapdoor mechanics. The arm's down position is shown in red; the up position is shown in blue. Vertical red arrows are labeled with the downward force in pounds.

THE HARDWARE

I had set an arbitrary $100 limit for the rest of the system and I quickly realized I would easily come in under budget. I used a $24.25 two-channel RF wireless garage door remote-control receiver, which I purchased from eBay (see **Photo 3**). This controller can be used in a latched or an unlatched mode. The latched mode requires a momentary push of one of the buttons to cause one of the relays to switch and stay in the On position. When the controller is in unlatched mode, you must hold the button down to keep the relay switched.

Unfortunately, this remote control and any similar ones only come with single-pole double-throw (SPDT) relays. What I really wanted were double-pole double-throw (DPDT) relays to switch both sides of the motor to enable current reversal through the motor.

A remote control system with two remotes seemed ideal and was possible to design with SPDT, so I purchased the relays. **Figure 2** shows the circuit using two bridge rectifier DC power supplies. It turns out there were problems with this approach.

SW1 and SW2 represent the Up and Down relays. In latched mode, the door would open when SW1 was energized using the A button on a remote. Pressing the A button again would stop the motor while the door was opening. So would pressing the B button, but then to continue opening the door you needed

FEATURES

PHOTO 1
The screw lift and pivot arm mechanism with a spring assist are shown.

PHOTO 2
An arbor is used as a bearing with a 0.75″ bolt through the floor joist. The lift mechanism pivots at this point.

to press the B button again. Pressing the A button in this state would cause the door to close because SW2 was still energized. Added to this confusion was the necessity of pressing the A button again when the door was fully opened and stopped due to the internal limit switches. If you didn't do this, then pressing the B button to close the door wouldn't work because SW1 was still energized.

I decided to just use the door in unlatched mode and continuously hold down the A button until the door was fully open. What was the problem with this? Noise! Interference from the motor was getting back into the control and causing the relay to frequently switch on and off. This is not good for mechanical relays that have a finite contact life.

After playing around for a while with both operation modes, I noticed that even in the latched mode the motor would sometimes stop and it would occasionally even reverse itself. This was really bad and it became worse.

If both SW1 and SW2 happened to switch at the same time and if the current was at a maximum and there was arcing at the terminal, there could conceivably be a momentary short through a diode in each of the bridge rectifiers that would burn them out. Arc suppression devices wouldn't help because when active at high voltages, they would almost look like a short between the switch's terminals A,C. I needed to step back and rethink this.

I found a $8.84 two-channel DPDT relay switch board module on eBay. The module enabled me to use a single-power supply and isolated the motor current from the remote-control board. These relays boards have TTL inputs, so it is tricky to use relays on the remote control board to control the relays on the second relay board. You have to contend with contact bounce. Even if I incorporated debounce circuits, I still didn't have a way stop the door from opening if it was obstructed.

It was time to get with the 21st century. I needed to use a microcontroller and handle all the debounce and logic functions in firmware.

I bought a $12.95 Freescale Semiconductor FRDM-KL25Z development board, which uses the Kinetis L series of microcontrollers. The FRDM-KL25Z is built on the ARM Cortex-M0+ core. This board comes with multiple I/O pins, most of which can be programmed as required. It also has two micro-USB ports, one of which is used for downloading your program onto the microcontroller and for debugging. More on this later.

PUTTING IT ALL TOGETHER

The wireless control module requires 6 mA at 12 V. The relay board and the FRDM-KL25Z

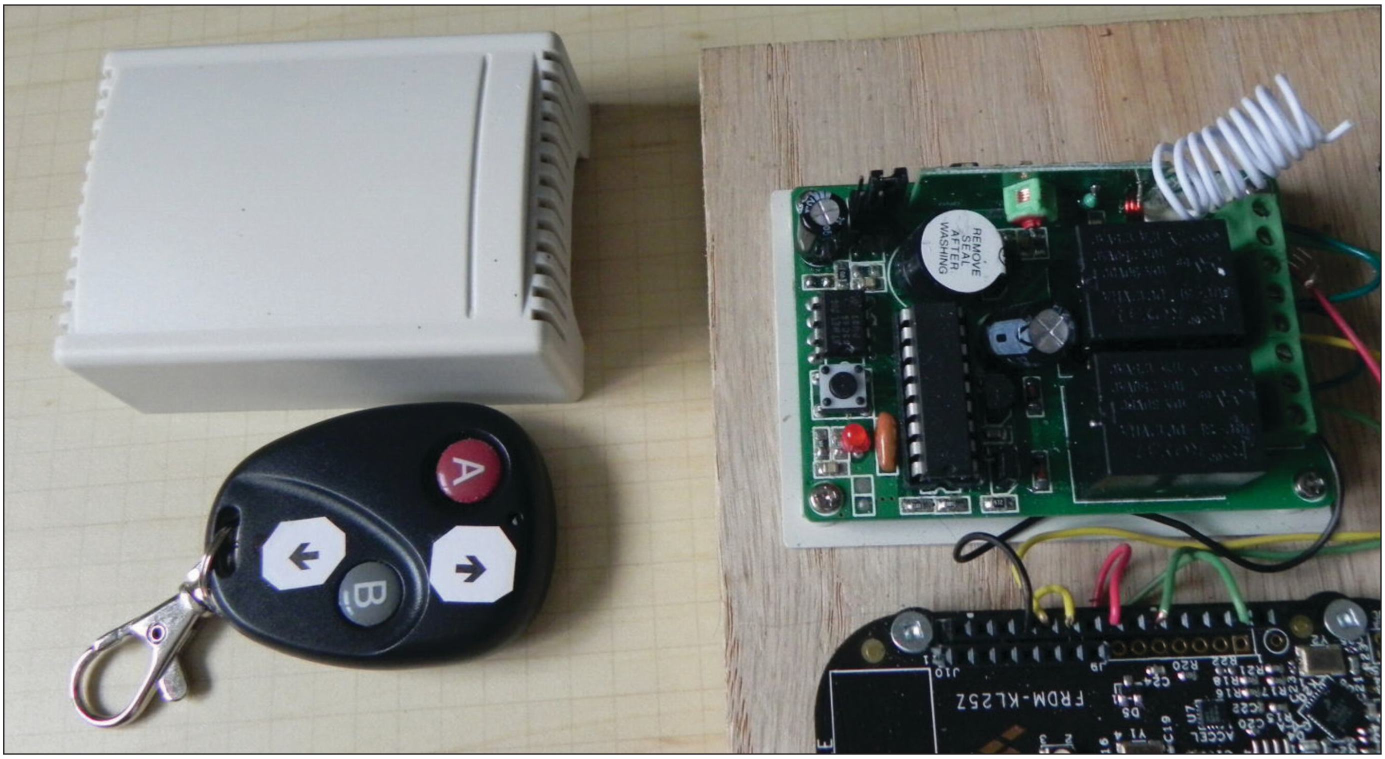


PHOTO 3
The two-channel wireless remote control is shown with the cover removed from the receiver. It came with two keychain-style remotes, which I marked with Up and Down arrows.

microcontroller are powered by 5 V. The microcontroller is powered via the USB port while under development. However, when it is stand-alone running, it can be supplied from 5 V via the V_{IN} pin.

Figure 3 shows a power supply for all three boards and an optoisolator circuit. For simplicity I used a single 12-V unregulated DC supply. I used LM7812 and LM7805 modules to regulate the voltages.

The optoisolator is used to monitor the motor current. The Motor 1 and Motor 2 lines are connected across a resistor that is wired in series with the motor. Because the current's polarity is switched to reverse motor direction, the voltage across the resistor passes a full-wave rectifier before going to the optoisolator. Variable resistor R2 is used for calibration.

Although the transfer characteristics of the optoisolator rectifier combination are not very linear, it doesn't matter since I was only comparing peaks with the firmware. I used a 3.3-V output from the FRDM-KL25Z with R3 to provide the current sense signal back to the microcontroller.

The control wiring diagram shows how it all goes together (see **Figure 4**). It's hard to find 48-V transformers, but 24-V transformers are plentiful and inexpensive, so I connected the secondary windings of two of them in series to power the motor.

PTB1 and PTB2 are defined by the FRDM-KL25Z's firmware to be digital inputs with pull-up resistors. They are pulled to ground by the remote-control board's relays. The DPDT relays on the two-channel relay module control the polarity of power to the motor.

mbed IDE PROGRAMMING

I used a PC running Microsoft Windows XP, but a later version could be used. Programming the FRDM-KL25Z can be done in C and or C++ using the mbed online IDE. Once you have set up an mbed account and defined the processor you are using for the compiler, it is simply a matter of using the menu to create a new program. The Getting Started page covers this well, so I will not go into detail.

Libraries for the FRDM-KL25Z are automatically inserted for each new program. The libraries have been written in C++. You can look into the source code if you prefer, but it really isn't necessary. As long as you are familiar with standard C and use the right port definitions you can forget

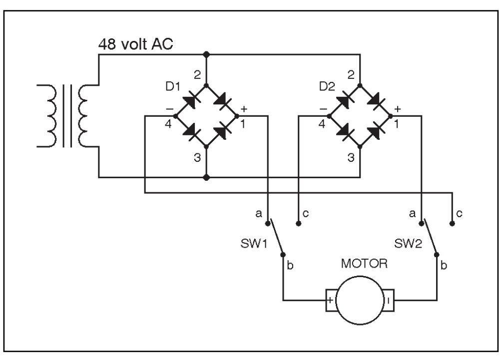


FIGURE 2
It would be theoretically possible to use dual-power supplies and single-pole double-throw (SPDT) switches to control a motor in two directions. When SW1 (b,c) is connected, current flows through D2. When SW2 (b,c) is connected, current flows through D1 in the opposite direction.

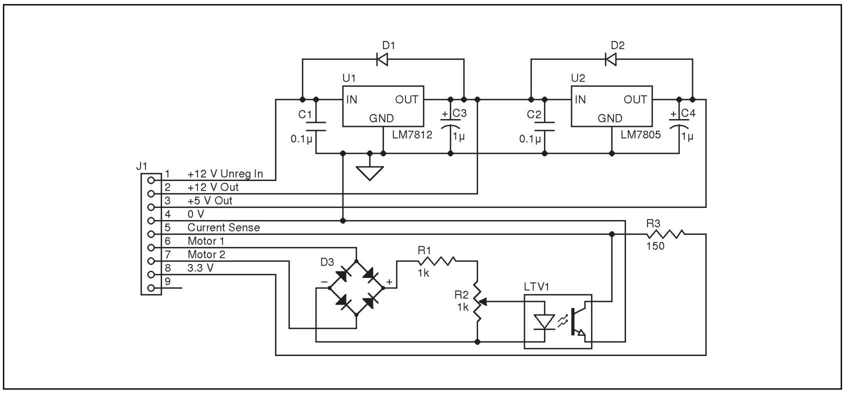


FIGURE 3
The circuit board contains 12- and 5-V regulated power supplies. I used the optoisolator circuit to monitor current through the motor.

about the C++ underneath.

The mbed software development kit (SDK) was designed to support ARM processors and mbed has recently added FRDM-KL25Z to the list of supported platforms. Most of the code examples are for the Arduino. The FRDM-KL25Z user manual makes it relatively easy to figure out.

I defined PTE20 and PET21 as digital outputs and PTB1 and PTB2 as digital inputs. PTB0 has been defined as an analog input. I used very little of this microcontroller's I/O capabilities, but it didn't bother me. The price was right and I got an inexpensive, high-performance machine to program.

There are a couple of ways to get started but the easiest is to just connect the micro-USB port marked SDA to a USB port on your computer. You will need a USB-to-micro USB cable for this.

The FRDM-KL25Z should appear as a new drive labeled "BOOTLOADER (*X*:)" with *X* being your PC's next available drive letter. There are four files on this drive. If you double-click on FSL_WEB.HTM you will be taken to a page on Freescale Semiconductor's website. Download the FRDM-KL25Z Quick Start package and unzip it and then drag and drop or copy the MSD-FRDM-KL25Z_Pemicro_v105.SDA file onto the BOOTLOADER drive. Unplug the cable from the SDA port and plug it back in.

You will get a "Found new hardware" message and, depending on your OS, it may want to search for drivers. You can allow it, but it won't find any, and eventually it will figure out what to do. You should then have a drive labeled "FRDM-KL25Z (*X*:)." You are now ready to start programming.

I suggest trying one of the example programs first. Once you have compiled it, you can download the binary file and copy it to the FRDM-KL25Z (*X*:) drive. Unplug the USB cable and plug it back in. The FRDM-KL25Z will then be running your program. You can then make changes to the program or write a new one, compile it, and download it to the FRDM-KL25Z (*X*:) as you wish.

THE TRAPDOOR PROGRAM

I wrote the trapdoor control program in C. It does not use an RTOS, although that is available in the mbed IDE. An RTOS would be more elegant, but in this case it was overkill. Instead, I used timed software loops. The digital filter is a 1-ms loop to sample the input while the loops to test motor current operate on 10-ms cycles.

Two digital inputs are defined for the Up and Down signals and an analog input is defined for the motor current sense signal from the power supply board. Two digital outputs are sent to the two-channel DPDT relay module to handle the motorized screw lift control. Since there is a three-colored LED on the FRDM-KL25Z module, I used it to show the control direction. Green indicates the system is ready and waiting for input, blue is used for forward or up, and red is used for down or reverse.

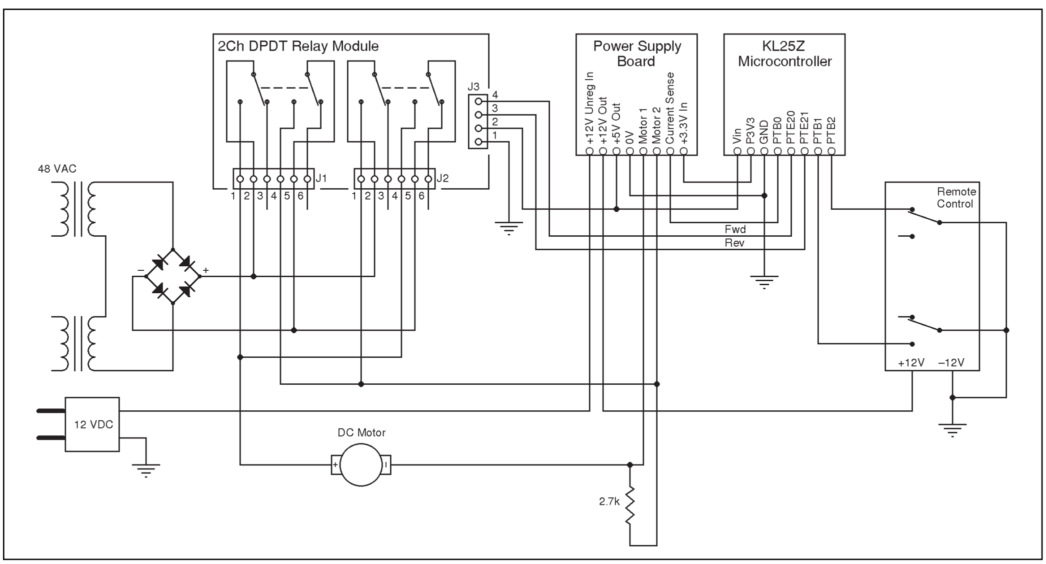


FIGURE 4
This is the system's complete wiring diagram. On the left is a 48-V AC supply and an unregulated 12-V DC motor. A 2.7-Ω, 5-W resistor, which is used for current sensing, is in series with the motor.

FEATURES

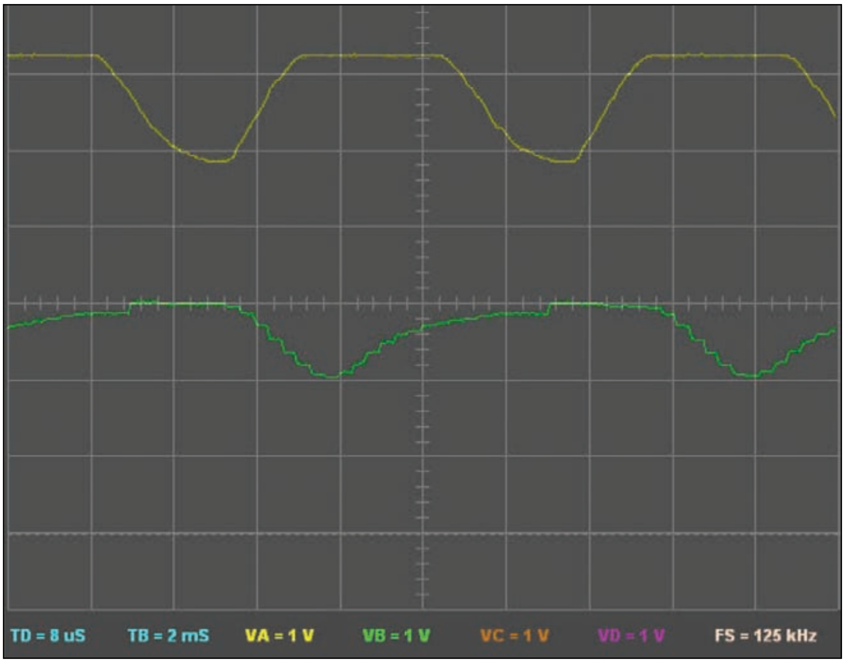


PHOTO 4
The top trace (yellow) is the optoisolator's output, which inverts the voltage seen across the motor current sense resistor. The bottom trace (green) is the output from the digital filter for a 1-A motor current.

There are three major parts to the program. First a routine monitors the digital inputs and provides the debounce function. A digital low-pass filter is included to reduce noise. It has the form:

$$y[n] = y[n-1] + k(x[n-1] - y[n-1])$$

where x is the sequence of input values, y is the sequence of output values, and k = 1/T. T is the time constant, similar to the RC values in an analog filter.

Photo 4 shows an oscilloscope trace of the analog input and the low-pass filter's digital result. Note that the current sense signal is 3.3 V and goes lower as the current increases. I adjusted the power supply module's calibration potentiometer so there was a 1-V peak after the low-pass filter when the motor current was at normal maximum (i.e., when the door is just beginning to open).

A state machine handles the control logic. On startup it is in the Off state, where it just monitors the inputs. When a debounced input is detected, it goes into a Forward (FWD) or Reverse (REV) state and outputs the appropriate signal to the relay module.

In the FWD or REV state, the current is monitored to see if the motor has actually started and if it is an overcurrent situation. If there is an overcurrent, the motor is turned off and the machine goes back to the Off state. Also if there is no current within `MAX_IN_WAIT_COUNT` time, it returns to the Off state (see Project Files). Otherwise, the machine goes to the `FWD_WITH_CURRENT` or the `REV_WITH_CURRENT` state.

The current is continually monitored for either overcurrent or no current. If either condition occurs or if the same button is pressed again, the motor is turned off and the machine returns to the Off state. The functionality here is for the motor to start in one direction. It can be manually stopped by pressing the same button again. If it is moving in the Up direction, for example, the Down button is ignored and vice versa.

DEBUGGING AND TESTING

Debugging can be problematic with microcontrollers if you want to do sophisticated things (e.g., setting break points, single stepping the program, and checking the values of registers and variables). When you are not multitasking, the easiest way is to go back to the early days of microprocessors and use `printf()` statements.

The FRDM-KL25Z's programmable I/O pins can also be used in conjunction with an oscilloscope or a logic analyzer. As mentioned earlier, a three-colored LED can be used to provide some idea of what your program is doing. I used all three methods in this project.

The FRDM-KL25Z's SDA USB port can also be used as a serial port for debugging. To do this, you first need to install the "mbedWinSerial_16466.exe" driver on your PC. It is available in the mbed handbook under the "Serial Communication with a PC" section. Once it is installed you can use a terminal program to communicate with the FRDM-KL25Z when it is connected to your PC via the SDA USB port.

I used LogMeTT.com's Tera Term open-source program. When running the terminal program, just select the com port the mbedWinSerial_16466 driver has installed. All `printf()` statements in the program will then be routed out to the terminal.

To calibrate the system I used a simple version of the program to turn on the motor. I used a meter and an oscilloscope to measure

PROJECT FILES

circuitcellar.com/ccmaterials

SOURCES

ARM Cortex-M0+ processor
ARM, Ltd. | www.arm.com

FRDM-KL25Z Development platform and Kinetis L series microcontrollers
Freescale Semiconductor, Inc. | www.freescale.com

TurboCAD program
IMSI/Design, LLC | www.turbocad.com

Tera Term terminal application
LogMeTT.com | http://logmett.com

mbed IDE and SDK
mbed | http://mbed.org

the voltage across R3. I temporarily defined an analog output and added code to display the filtered values' results to see the low-pass filter's output (see **Photo 4**). It was then a simple matter of adjusting the 1-kΩ potentiometer on the power supply board to get the peak values I wanted to use in the program.

Testing the system was straightforward after performing bench tests to ensure that the remote control and the relay board worked. Once the entire system was hooked up, the last thing to test was whether it would shut down if the trapdoor was obstructed. To do this, I placed a 24-bottle case of beer on the door and attempted to open the trapdoor. As the door began to open, the trapdoor quickly stopped. Further attempts to open the trapdoor moved the door up a few degrees before stopping. At the risk of having the case of beer slide across the floor and cause possible breakage, I considered three attempts a successful test and drank a beer.

CHEERS!

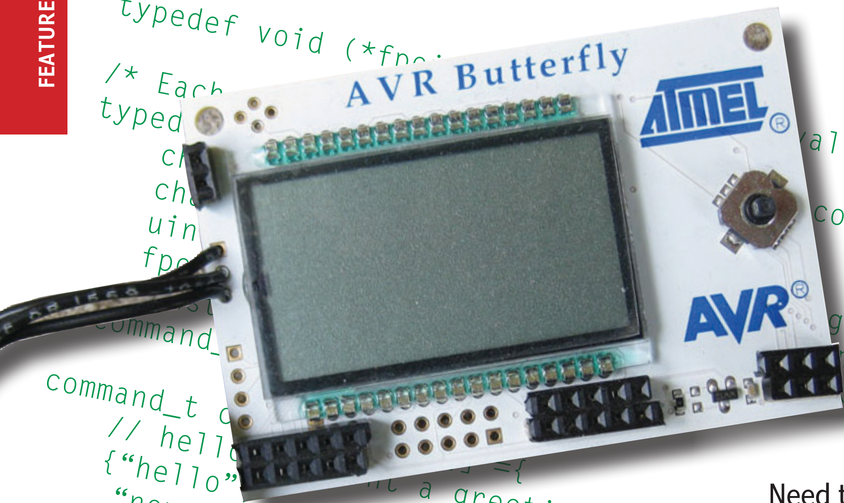
Although it can be fulfilling to design your own hardware, why reinvent the wheel if you don't have to? Many reasonably priced modules can be wired together. Add some software to express the functionality required and you can quickly and inexpensively put together a project. (It took me as much time to write this article as it did to design and implement the electronic and software portion.) Not only is this method useful for a homebuilt one-of-a-kind application, but it can also be used for proof-of-concept prototyping. It leaves you free to concentrate on solving the problems pertinent to your application.

ABOUT THE AUTHOR

William Wachsmann (bwachsmann@cabletv.on.ca) is a retired electronic technologist who lives in on the shore of Lake Huron, Canada. He has more than 35 years of experience working with minicomputers, microcomputers, embedded systems, and programming in a variety of industries from nuclear to aerospace, to voicemail and transportation systems. William spent nearly half of his career as a consultant for his company Meta3+ in Montreal and Toronto.

FEATURES

A Coding Interface for an Evaluation Tool



Need to send a few commands from a PC to Atmel's AVR Butterfly? This article describes the basics of using the AVR Butterfly to recognize, parse, and execute remote commands.

By John Peck (US)

I love test equipment with open, well-documented, ASCII command sets. The plain text commands give a complicated instrument a familiar interface and an easy way to automate measurements. I like using Python's gnuplot and pySerial interfaces to acquire and make pretty plots of my data—all for free. So when I needed to automate the process of reading the voltage output from an ultrasonic range finder, I wanted an ASCII interface to a voltmeter. But, since I wanted this meter also to convert volts into distance, I needed more than just a voltmeter. I had an Atmel AVR Butterfly, and I thought it would be easy to give it a plain text interface to a PC.

This became a bigger project than I expected. I came up with a simple command interface that's easy to customize and extend. It's not at the level of a commercial instrument, but it works well for sending a few commands and getting some data back.

WORKING WITH THE AVR BUTTERFLY

The AVR Butterfly board includes an Atmel ATmega169 microcontroller and some peripherals. **Figure 1** shows the connections I made to it. I only used three wires from the DB9 connector for serial communication with the PC. There isn't any hardware handshaking. While I could also use this serial channel for programming, I find that using a dedicated programmer makes iterating my code much faster.

A six-pin header soldered to the J403 position enabled me to use Atmel's AVRISP mkII programmer. Finally, powering the board with an external supply at J401 meant I wouldn't have to think about the AVR Butterfly's button cell battery. However, I did need to worry about the minimum power-on reset slope rate. The microcontroller won't reset at power-on unless the power supply can ramp from about 1 to 3 V at more than

0.1 V/ms. I had to reduce a filter capacitor in my power supply circuit to increase its power-on ramp rate. With that settled, the microcontroller started executing code when I turned on the power supply.

After the hardware was connected, I used the AVR downloader uploader (AVRDUDE) and GNU Make to automate building the code and programming the AVR Butterfly's flash memory. I modified a makefile template from the WinAVR project to specify my part, programmer, and source files. The template file's comments helped me understand how to customize the template and comprehend the general build process. Finally, I used Gentoo, Linux's cross-development package, to install the AVR GNU Compiler Collection (AVR-GCC) and other cross-compilation tools. I could have added these last pieces "by hand," but Gentoo conveniently updates the toolchain as new versions are released.

HANDLING INCOMING CHARACTERS

To receive remote commands, you begin by receiving characters, which are sent to the AVR Butterfly via the USART connector shown in **Figure 1**. Reception of these characters triggers an interrupt service routine (ISR), which handles them according to the flow shown in **Figure 2**. The first step in this flow is loading the characters into the Receive buffer.

Figure 3 illustrates the Receive buffer loaded with a combined string. The buffer is accessed with a pointer to its beginning and another pointer to the next index to be written. These pointers are members of the `recv_cmd_state_t`-type variable `recv_cmd_state`.

This is just style. I like to try to organize a flow's variables by making them members of their own structure. Naming conventions aside, it's important to notice that no limitations are imposed on the command or argument size in this first step, provided the total character count stays below the `RECEIVE_BUFFER_SIZE` limit.

When a combined string in the Receive buffer is finished with a carriage return, the string is copied over to a second buffer. I call this the "Parse buffer," since this is where the string will be searched for recognized commands and arguments. This buffer is locked until its contents can be processed to keep it from being overwhelmed by new combined strings.

Sending commands faster than they can be processed will generate an error and combined strings sent to a locked parse buffer will be dropped. The maximum command processing frequency will depend

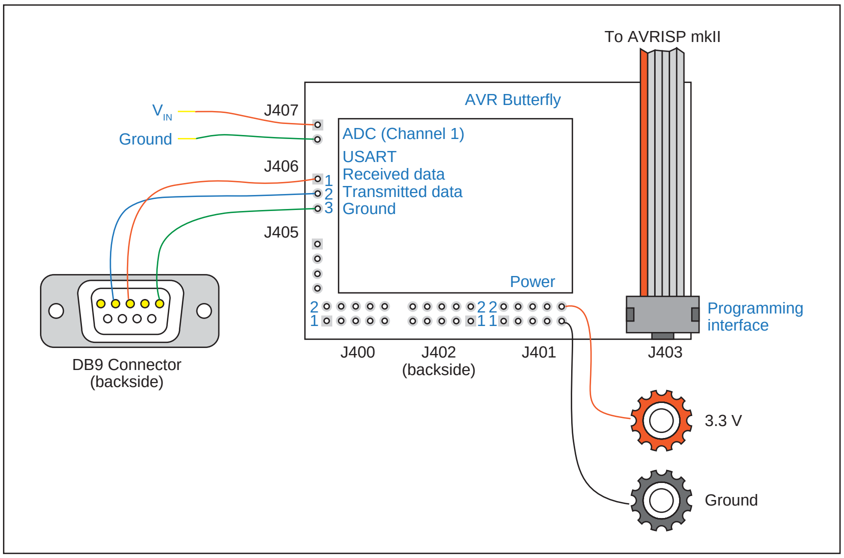


FIGURE 1
These are the connections needed for Atmel's AVR Butterfly. Atmel's AVRISP mkII user's guide stresses that the programmer must be connected to the PC before the target (AVR Butterfly board).

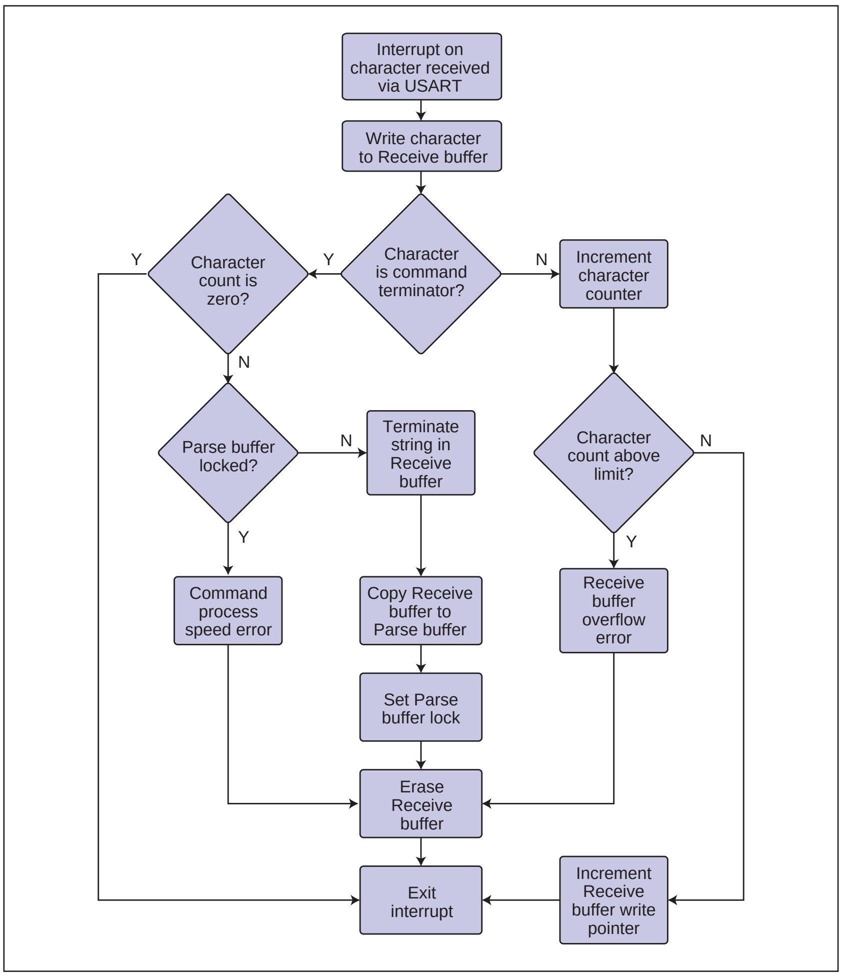


FIGURE 2
This is the program flow for processing characters received over the Atmel AVR Butterfly's USART. Sending a command terminator (carriage return) will always result in an empty Receive buffer. This is a good way to ensure there's no garbage in the buffer before writing to it.

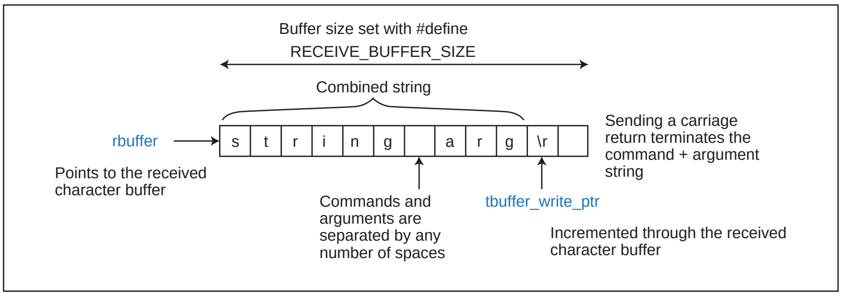


FIGURE 3
The received character buffer and pointers used to fill it are shown. There is no limit to the size of commands and their arguments, as long as the entire combined string and terminator fit inside the `RECEIVE_BUFFER_SIZE`.

on the system clock and other system tasks. Not having larger parse or receive buffers is a limitation that places this project at the hobby level. Extending these buffers to hold more than just one command would make the system more robust.

WATCHING THE PROGRESS

I set up a logging system to help me understand how commands were being processed. It's inconvenient to have the command and logging interfaces share the same communication channel, but the AVR Butterfly only has one USART. Reading command replies back without getting confused by general log messages requires some control over how the logging system works. **Photo 1** shows the system processing the `hello` command with logging enabled at its most verbose level.

Each log message is tagged with a character representing the message severity (e.g., ISR, information, warning, error) followed by a string representing the subsystem responsible for the message. These tags can be used to adjust the log messages to be suppressed.

I made the ISR tag an integer representing a message severity level so I could quickly suppress messages generated by the ISR using an integer comparison. The last `Hello yourself!` message shown in **Photo 1** is the reply to the `hello` command. This reply isn't a log message, thus it doesn't have any tags.

I should mention that these log message strings can easily overwhelm the AVR Butterfly's RAM if they're not stored and referenced in flash memory. Savannah and Dean Camera have written great instructions for using the pgmspace module to handle this problem (see Resources). This module supplies the `PROGMEM` and `PSTR` macros I use throughout my code.

CONFIGURING THE LOGGER

The logger acts as a gatekeeper for messages emitted from other subsystems. **Photo 1** shows a terminal output in which the command subsystem has a lot to say about every character received over the USART. At 9,600 bps, sending an 80-character message (10 bits per character) takes almost 100 ms. The processor will not be able to handle new commands (or even new characters) during this time, as it won't get around to processing the Parse buffer or even finishing the interrupt. So you need to suppress messages you don't need to see.

The `logger_msg_p` function is used to send messages to the logger. The `_p` suffix indicates that the message must be stored and referenced in flash memory. These messages are tagged with the origin subsystem and a severity level. These two tags have corresponding members in the logger configuration structure, which the logger consults when deciding whether or not to print the message. The system boots with the severity level set above the ISR. If it didn't, you'd have to send characters spaced by about 100 ms to send commands.

After the system boots, you can use the `logreg` and `loglevel` commands to customize which systems to print messages from and how severe they must be. Before describing these commands, I need to say something about how the logger sorts messages. The logger first checks to see if logging has been turned off. This is done by clearing all bits in a logger configuration register, which is the fastest way to veto all messages. If logging hasn't been turned off, the incoming message's severity is compared with the configuration level. This level is an enumerated type, with the `log_level_ISR` member given the lowest value and `log_level_ERROR` given the highest. If the severity is high enough, the logger will check to see if the message's subsystem has been enabled for logging.

Subsystem tags are recognized by

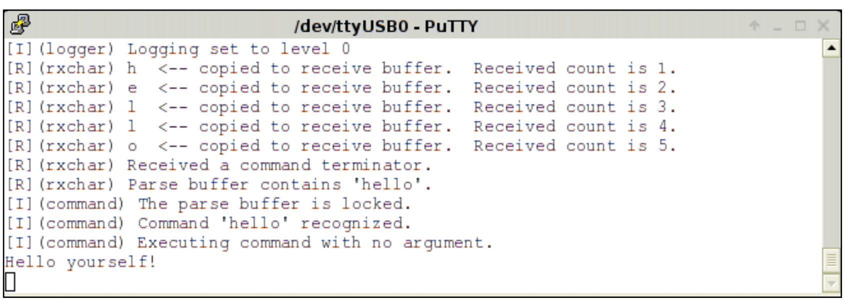


PHOTO 1
This terminal output shows the reception of and reply to the `hello` command. The logging system has been set to its most verbose level, so there are "unsolicited" messages as well as the command reply. The messages can be suppressed by clearing bits in the logger configuration register.

comparing the tag string with strings in the logger module's system array. This can be slow, which is why I vetoed messages from the ISR with a severity level and not by disabling subsystems.

Each subsystem has one bit in a 16-bit register, which is the logger configuration register. Setting the bit enables the subsystem's messages. Clearing the bit suppresses the messages. Clearing all bits sets the "logger is turned off" state described earlier. The `logreg` remote command enables users to set bits in this register while the system is running. Of course, for this command to be useful, the user has to be told how

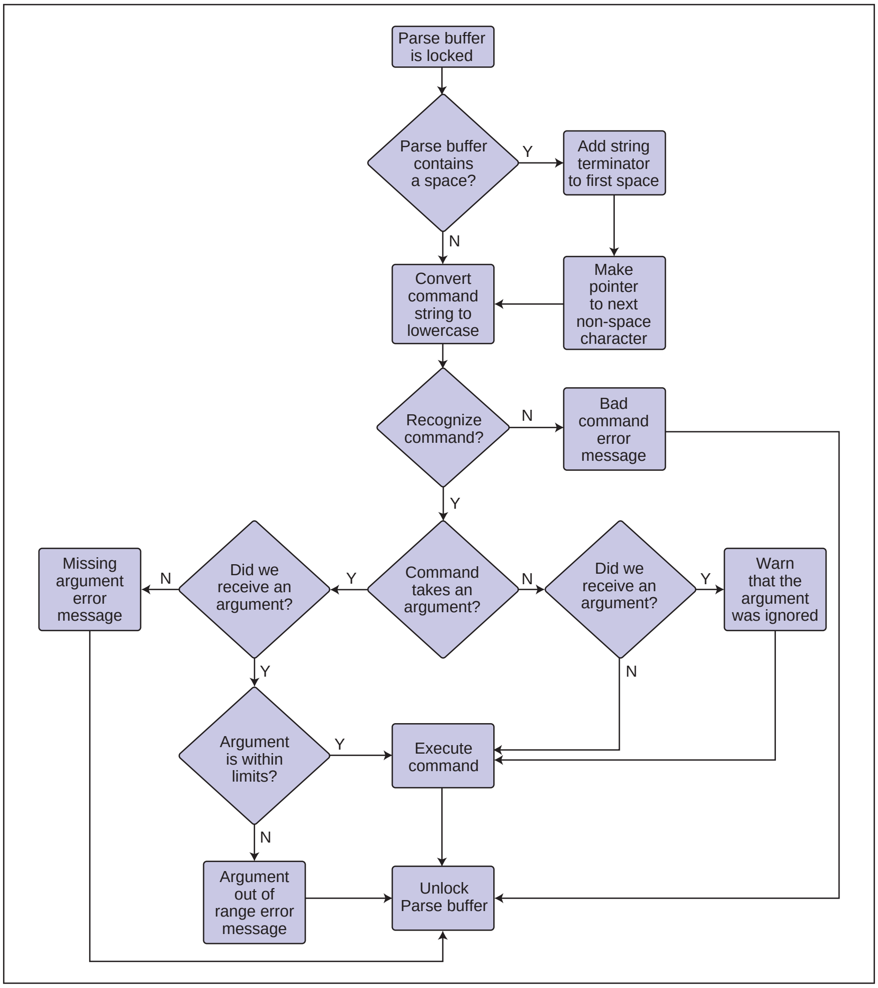


FIGURE 4
This program flow processes fully formed commands. This processing step happens in the main loop, so it will only run when the system isn't busy doing something else. Notice that all incoming commands are converted to lowercase, so there's no case sensitivity.

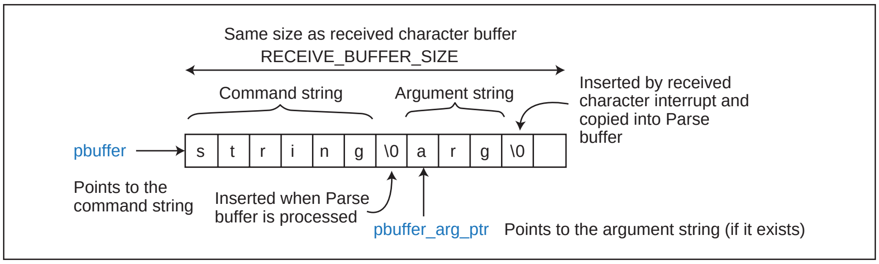


FIGURE 5
The Parse buffer with pointers to the command and argument strings is shown. The single combined string copied from the Receive buffer is made into two strings by inserting a string terminator.

```
typedef void (*fpointer_t)(uint16_t argval);

/* Each command_struct will describe one command */
typedef struct command_struct {
    char *name; // The name of the command
    char *arg_type; // A string representing the type of argument expected
    uint8_t arg_max_chars; // The maximum number of characters in the argument
    fpointer_t execute; // The function to execute
    const char *help;
} command_t;
```

LISTING 1
This is the definition of `command_t` from `bc_command.h`. This code snippet shows how the command structure is defined. When the command processor is run, commands in the Parse buffer are matched with names from the command array. Commands can be added to the system by adding to this array of command types.

```
command_t command_array[] ={
    // hello -- Print a greeting.
    {“hello”,           // Name of the command
     “none”,            // Argument type (can be “none” or “hex” right now)
     0,                 // Maximum number of characters in argument
     &cmd_hello,        // Address of function to execute
     helpstr_hello},    // The help text (defined above)
    // logreg -- Set the logger enable register.
    {“logreg”,
     “hex”,
     4,
     &cmd_logreg,
     helpstr_logreg},
     // logreg? -- Query the logger enable register.
    {“logreg?”,
     “none”,
     0,
     &cmd_logreg_q,
     helpstr_logreg_q},
     // help -- Print all the help strings
     {“help”,
     “none”,
     0,
     &cmd_help,
     helpstr_help},
     // End of table indicator.  Must be last.
    {“”,””,0,0,nullstr}
};
```

LISTING 2
This code snippet shows how to initialize the command array in `bc_command.c` (with some commands not shown).

to match bit positions with subsystems. The `loglevel` command is more intuitive. To set the severity threshold, it takes an integer from 0 to 3, with 3 being the most severe.

Each subsystem needs an entry in the logger's subsystem array for this scheme to work. It might be nice to automate the creation of this array so the programmer doesn't have to think about it when adding a new subsystem. For now, the logger module must be made aware of new subsystems by manually adding these array entries.

RECOGNIZING COMMANDS

After combined strings are copied from the Receive buffer to the Parse buffer, the system separates them into command and argument strings. **Figure 4** shows the program flow. Commands in the Parse buffer are then separated from their arguments with a string terminator inserted into the first space between the two.

Pointers to the beginning of the Parse buffer and the beginning of the argument will then reference two separate strings (see **Figure 5**). The first of these two, the command string, is converted to lowercase and compared with those in the command definition array to look for a match.

Each entry must be the `command_t` type, which is defined in **Listing 1**. **Listing 2** shows my simple command definition array

ABOUT THE AUTHOR

John Peck, PhD (john@johnpeck.info) is a test engineer at Knowles Electronics in Itasca, IL. He has been using ASCII interfaces to test equipment since he was a graduate student at the University of Wisconsin at Madison.

with entries for the hello, logreg, logreg?, and help commands. This array's structure is largely taken from E. White's *Making Embedded Systems: Design Patterns for Great Software*.

Notice that the function pointer member of this command type only accepts one integer argument. This can be changed to make the system more flexible, but remember that every function called by remote commands must accept the same arguments. If the function pointer is expanded to accept both an integer and a string pointer argument, all the functions the pointer will point to must also be expanded.

The other members of the command type contain the command's name (must be lowercase), its argument type, the maximum number of characters in the argument, and a help string. The argument type tells the command processing system how it should handle the argument string. For example, hexadecimal argument strings are converted to integers before being passed on. The character limit is a basic way of limiting incoming arguments. Finally, the help string is what is printed to the USART when the help command is issued.

ADDING MORE COMMANDS

Since the ability to extend this system is so important, I developed a procedure for adding new commands. First, choose your command's name. This can be harder than it sounds. I like names to be short enough to be easily entered on the command line, but long enough to be descriptive. The name can contain any ASCII character except a space. Remember the command characters, the argument characters, one space, and one string terminator must all fit in the received character buffer.

Next, decide whether or not you need more than an unsigned hexadecimal integer string argument. My argument-handling ability is limited to just strings representing hexadecimal integers. Extending the system isn't difficult, but remember that every function called by a command has to accept the same arguments. If you add a character-pointer argument, even commands that don't take any arguments must then call functions that need both integer and character-pointer arguments. This can be tedious if you already have a lot of commands defined.

If your arguments fit your ability to handle them, add a function for your command to call. You may want to put all the functions called by remote commands in their own module. I prefer to put these functions in the module with which they're associated. I prefix their names with cmd_. For example,

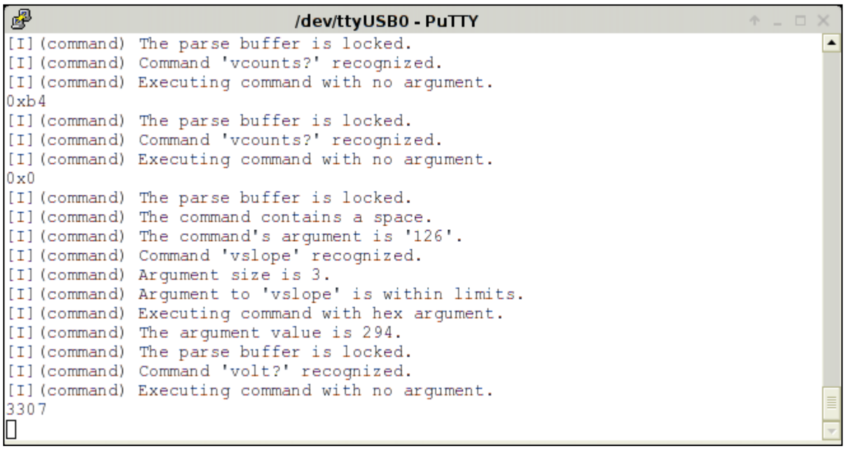


PHOTO 2
This terminal output shows the voltage reader's calibration. I recorded ADC counts first with the power supply voltage applied (3.308 V) and then with the reader grounded. I used these two outputs to calculate a calibration factor and used the vslope command to apply it. No offset was needed. The final output shows the power supply measurement reported in fixed-point millivolts.

the function called by the logreg remote command is cmd_logreg and I implement it in the bc_logger module.

Once the command's attributes have been specified, it's time to give it an entry in the command array. Add the new entry before the array's end marker. You've already chosen the command's name, argument type, and function call. The remaining entry members are the argument's maximum size in characters and the command's help string. Remember that hexadecimal arguments larger than four characters don't make sense for 16-bit integers unless you allow the user to add a leading "0x." The argument parser will return the same value for 0xff and ff, but allowing the extra two prefix characters opens the door to much larger arguments. As for the help strings, I was only able to use the `PROGMEM` macro on them if I defined them outside of the command array. This breaks up the array organization, but these strings will quickly eat up the AVR Butterfly's RAM if they're not located and referenced in flash.

SENDING CALIBRATION VALUES

My original goal was to calibrate measurements made with the AVR Butterfly's voltage reader and to use the PC to read those measurements. I added the commands `vcounts?`, `volt?`, `vslope`, and `voffset` to handle the calibration flow. The `vcounts?` query returns raw counts from the voltage reader's ADC and `volt?` returns the calibrated measurement. Raw counts are multiplied by the factor set with `vslope` and that product is divided by 16 to get the voltage reading in millivolts including offset. If there's offset, it can be subtracted by sending the number of millivolts to subtract with `voffset`.

Of course the functions I call with commands only accept positive integers, so the offset value can't be negative. I didn't have a problem with this here, but providing for a negative offset or slope factor would be a good reason to extend the arguments accepted by these functions.

Photo 2 shows the voltage reader's logger output during calibration. I used the power supply rail (3.308 V) and ground as references. The slope factor comes from:

$$\text{vslope} = 16\left(\frac{V_H - V_\ell}{N_H - N_\ell}\right) = 16\left(\frac{3{,}308 - 0}{180 - 0}\right) = 294$$

where V_H and V_ℓ are the voltage references in millivolts and N_H and N_ℓ are the raw ADC counts at those references. The factor of 16 increased my calibration precision while constraining the system to measure less than about 3.5 V. After I set the slope factor, I could measure the ground reference again to see if I need to adjust the offset. There wasn't any (within my roughly 20-mV precision), so the last line of **Photo 2** shows the calibrated result of measuring the power supply rail.

Why was the result 3.307 instead of 3.308? I divided the product of the raw ADC counts and the slope factor by 16 with a simple bit shift. I discarded the fractional remainder instead of rounding it properly. The AVR Butterfly's voltage reader isn't a good way to make a measurement with millivolts of precision anyway.

IS THERE ANY SPACE LEFT?

I built the code with AVR-GCC, using the -Os optimization level. The output of avr-gcc --version is avr-gcc (Gentoo 4.6.3 p1.3, pie-0.5.1) 4.6.3.

The resulting memory map file shows a 306-byte .data size, a 49-byte .bss size, and a 7.8-KB .text size. I used roughly half of the AVR Butterfly's flash memory and about a third of its RAM. So there's at least some space left to do more than just recognizing commands and calibrating voltages.

I'd like to work on extending the system to handle more types of arguments (e.g., signed integers and floats). And I'd like to port the system to a different part, one with more than one USART. Then I could have a dedicated logging port and log messages wouldn't get in the way of other communication. Making well-documented interfaces to my designs would help me with my long-term goal of making them more modular.

PROJECT FILES

circuitcellar.com/ccmaterials

RESOURCES

D. Camera, "GCC and the PROGMEM Attribute," Four Walled Cubicle, 2014.

Free Software Foundation, Inc., "AVRDUDE—AVR Downloader/UploaDEr," www.nongnu.org.

———, "GNU Make," www.gnu.org.

Savannah, "Data in Program Space," www.nongnu.org.

SourceForge, "pySerial," http://pyserial.sourceforge.net.

———, "WinAVR: AVR-GCC for Windows," http://winavr.sourceforge.net.

———, "Gnuplot.py," http://gnuplot-py.sourceforge.net.

E. White, *Making Embedded Systems: Design Patterns for Great Software*, O'Reilly Media, Inc., 2011.

SOURCES

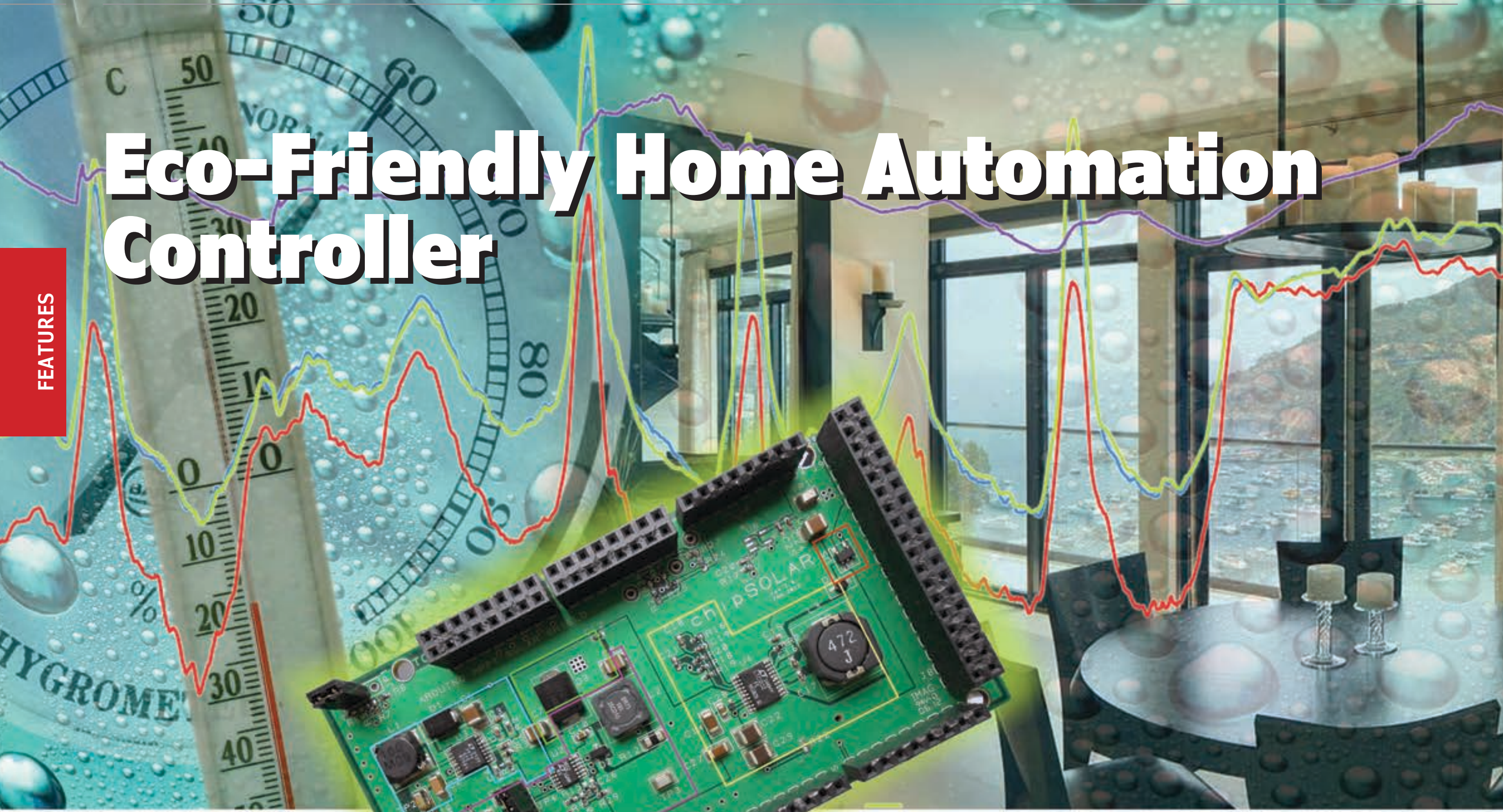
AVR Butterfly, ATmega169 microcontroller, and AVRISP mkII programmer

Atmel Corp. | www.atmel.com

AVR GNU Compiler Collection

Free Software Foundation, Inc. | gcc.gnu.org

Eco-Friendly Home Automation Controller



This system, which is built around an Arduino-compatible board, connects to a custom board that uses a solar panel and two rechargeable lithium-ion cells to provide continuous power. A wireless board integrating a GSM/GPRS modem, an XBee socket, an SD card connector, and a real-time clock and calendar (RTCC) enables home sensor and cloud connectivity.

By Manuel Iglesias Abbatemarco (Venezuela)

I built an autonomous home automation controller that enables users to monitor and control household devices (e.g., HVAC, lighting, automatic doors, and other appliances). It can also provide security features (e.g., an anti-theft alarm). An important feature is the ability to log sensor data (e.g., temperature, humidity, and electricity consumption) and upload it to "the cloud." **Photo 1** shows the complete project.

I consider autonomy a basic feature of my dream eco-friendly controller. Before starting this project, I did my homework on commercial devices that enable you to control and monitor your home. They were costly and none had all the features I wanted on a single device. Typically, you cannot modify or add new features to such devices.

This article presents a device prototype with several features that make it autonomous. The design was part of my entry for the 2012 DesignSpark chipKIT Challenge. Planning the device was difficult, primarily because I live outside the US and prototyping takes a long time.

Since the contest, I have made some additions to the system. The device's aim is uninterrupted household monitoring and control. To accomplish this, I focused on two key features: the power controller and the communication with external devices (e.g., sensors). I used DesignSpark software to create two PCBs for these features.

DESIGN OVERVIEW

The system's design is based on a Digilent chipKIT Max32 board, which is an Arduino-compatible board with a Microchip Technology 32-bit processor and 3.3-V level I/O with almost the same footprint as an Arduino Mega microcontroller. The platform has all the computational power needed for the application and enough peripherals to add all the required external hardware.

I wanted to have a secure and reliable communication channel to connect with the

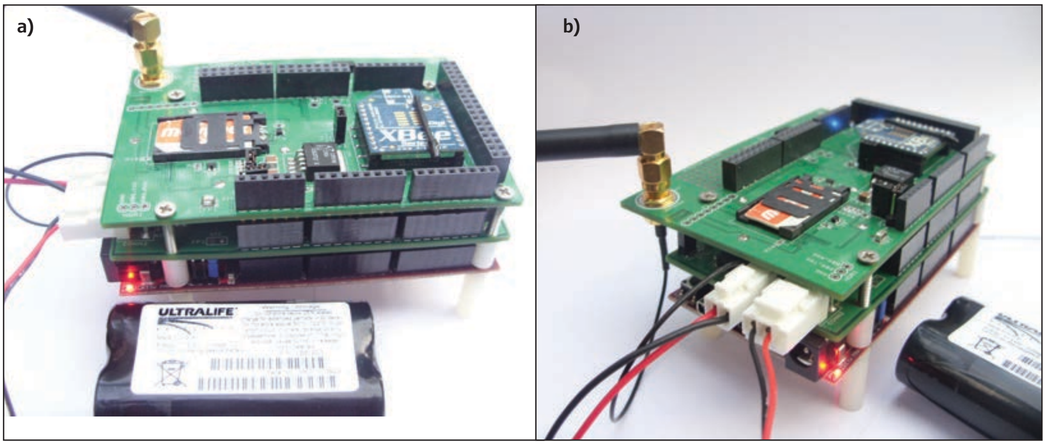


PHOTO 1
a—This is the complete controller project that I entered in the 2012 DesignSpark chipKIT Challenge. **b—**The design comprised a Digilent chipKIT board (bottom), my MPPT charger board (chipSOLAR, middle), and my wireless board (chipWIRELESS, top).

outside world, so I incorporated general packet radio service (GPRS). This enables the device to use a TCP/IP client to connect to web services. It can also use Short Message Service (SMS) to exchange text messages to cellular phones. The device uses a serial port to communicate with the chipKIT board.

I didn't want to deal with cables for the internal-sensor home network, so I decided to make the system wireless. I used XBee modules, as they offer a good compromise between price and development time. Also, if properly configured, they don't consume too much energy. The XBee device uses a serial port to communicate with the chipKIT board.

To make the controller "green," I designed a power-management board that can work with a solar panel and several regulated DC voltages. I chose a hardware implementation of a maximum power point tracking (MPPT) controller because I wanted to make my application as reliable as possible and have more software resources for the home controller task.

One board provides power management and the other enables communication, which includes additional hardware such as an SD card, an XBee module, and a real-time clock and calendar (RTCC). Note: I included the RTCC since the chipKIT board does not come with a crystal oscillator. I also included a prototyping area, which later proved to be very useful.

I was concerned about how users inside a home would interact with the device. The idea of a built-in web server to help configure and interact with the device had not materialized before I submitted the contest entry. This solution is very practical, since you can access the device through its built-in server to configure or download log files while you are on your home network.

POWER MANAGEMENT BOARD

To make the system eco-friendly, I needed to enable continuous device operation using only a solar panel and a rechargeable lithium ion (Li-ion) battery. The system consumes a considerable amount of power, so it needed a charge controller. Its main task was to control the battery-charging process. However, to work properly, it also had to account for the solar panel's characteristics.

A solar panel can't deliver constant power like a wall DC adapter does. Instead, power varies in a complex way according to atmospheric conditions (e.g., light and temperature).

For a given set of operational conditions, there is always a single operating point where the panel delivers its maximum power. The idea is to operate the panel in the maximum power point regardless of the external conditions.

I used Linear Technology's LT3652 MPPT charger IC, which uses an input voltage regulation loop. The chip senses the panel output voltage and maintains it over a value by adjusting the current drawn. A voltage divider network is used to program the setpoint.

You must know the output voltage the panel produces when operated at the maximum power point. I couldn't find the manufacturer's specification sheet for the solar panel, but the distributor provides some experimental numbers. Because I was in a hurry to meet the contest deadline, I used that information. Based on those tests, the solar panel can produce approximately 8 V at 1.25 A, which is about 10 W of power.

I chose 8 V as the panel's maximum power point voltage. The resistor divider output is connected to the LT3652's V_{IN_REG} pin. The chip has a 2.7-V reference, which means the charge current is reduced when this pin's voltage goes below 2.7 V.

I used a two-cell Li-ion battery, but since the LTC3652 works with two, three, and four cells, the same board with different components can be used with a three- or four-cell battery. The LT3652 requires an I/O

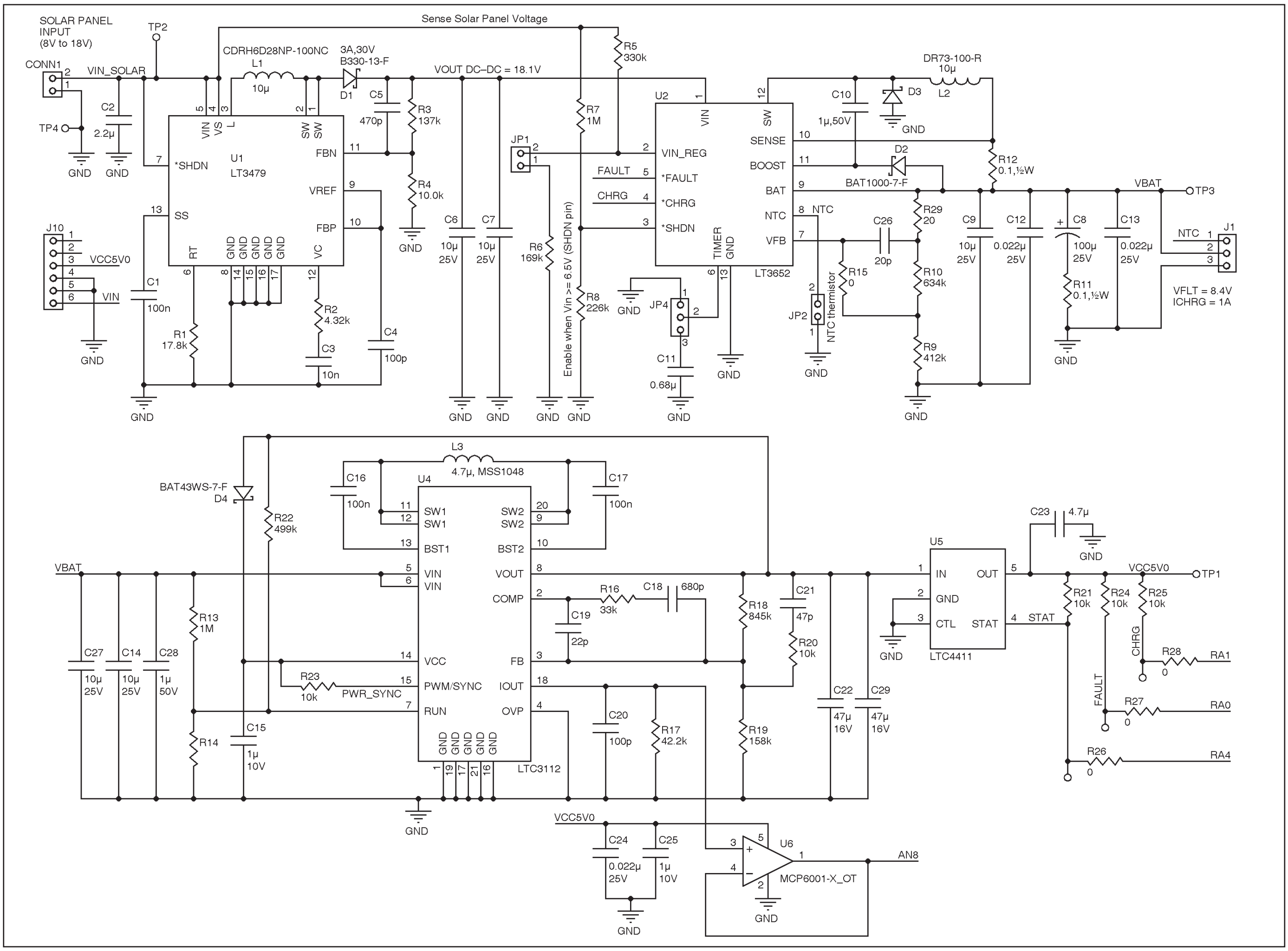


FIGURE 1
This is the power management board.

voltage difference of at least 3.3 V for reliable start-up, and it was clear that the panel's 8-V nominal output would not be enough. I decided to include a voltage step-up stage in front of the LT3652.

I used Linear Technology's LT3479 DC/DC converter to get the panel output to around 18 V to feed the MPPT controller. This only works if the LT3562's voltage control loop still takes the V_{IN_REG} reference directly from the panel output. **Figure 1** shows the circuit.

I could have fed the chipKIT on-board 5-V linear regulator with the battery, but I preferred to include another switching regulator to minimize losses. I used Linear Technology's LTC3112 DC/DC converter. The only problem was that I needed to be able to combine its output with the chipKIT board's 5 V, either through the USB port or the DC wall adapter option.

The chipKIT board includes a Microchip Technology MCP6001 op-amp in comparator configuration to compare USB voltage against a jack DC input voltage, enabling only one to be the 5-V source at a given time. Something similar was needed, so I included a Linear Technology LTC4411 IC, which is a low-loss replacement ORing diode, to solve the problem.

To my knowledge, when I designed the board a battery gauge for two-cell lithium batteries (e.g., a coulomb counter that can indicate accumulated battery charge and discharge) wasn't available. The available options needed to handle most of the computational things in software, so I decided it was not an option. I included a voltage buffer op-amp to take advantage of the LTC3112's dedicated analog voltage output, which gives you an estimate of the instantaneous current being drawn. Unfortunately, I wasn't able to get it to work. So I ended up not using it.

Building this board was a challenge, since most components are 0.5-mm pitch with exposed pads underneath. IC manufacturers suggest using a solid inner ground layer for switching regulators, so I designed a four-layer board. If you have soldering experience, you can imagine how hard it is to solder the board using only a hot air gun and a soldering iron. That's why I decided it was time to experiment with a stencil, solder paste, and a convection oven. I completed the board by using a commercially available kitchen convection oven and manually adjusting the temperature to match the reflow profile since I don't have a controller.

WIRELESS BOARD

The wireless board has all the components for GPRS communication and the 802.15.4 home network, as well as additional components for the SD file system and the RTCC. **Figure 2** shows the circuit.

At the time of the contest, I used a SIMCom Wireless Solutions SIM340 GPRS modem. The company now offers a replacement, the SIM900B. The only physical differences are the board-to-board connectors, but the variations are so minimal that you can use the same footprint for both connectors.

During the contest, I only had the connector for the SIM340 on hand, so I based almost all the firmware on that model. Later, I got the SIM900B connector and modified the firmware. The Project Files include the `#if defined` clause for SIM900 or SIM340 snippets.

A couple of things made me want to test the SIM900B module, among them the Simple Mail Transfer Protocol (SMTP) server functionality and Multimedia Messaging Service (MMS). Ultimately, I discovered that my 32-MB flash memory version of the SIM900B was not suitable for those firmware versions. The 64-MB version of the hardware is required.

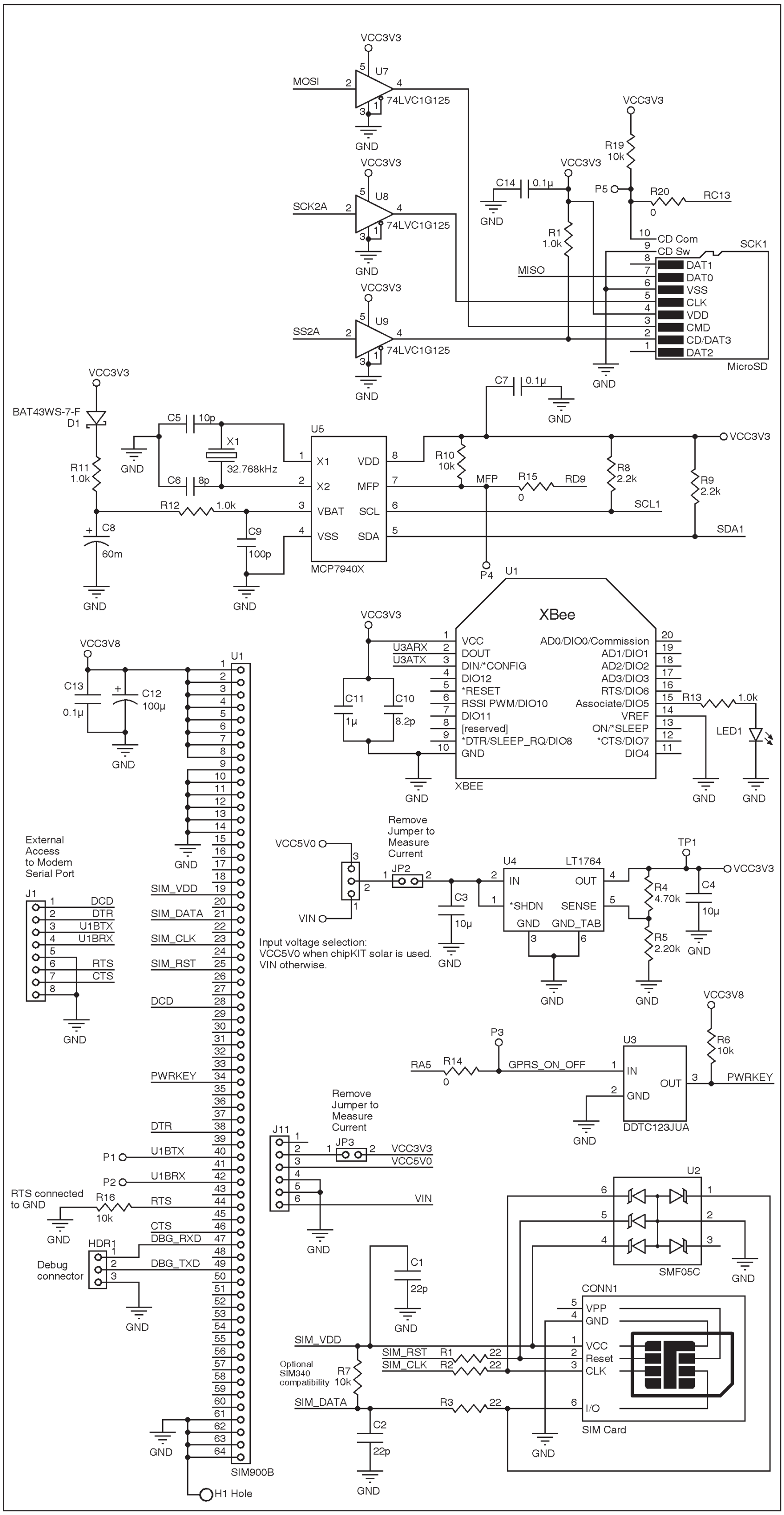
The subscriber identity module (SIM) card receptacle and associated ESD protection circuitry are located on the upper side of the board. The I/O lines connected to the modem are serial TX, RX, and a power-on signal using a transistor.

The chipKIT Max32 board does not have a 32,768-Hz crystal, so Microchip Technology's PIC32 internal RTCC was not an option. I decided to include Microchip Technology's MCP79402 RTCC with a super capacitor, mainly for service purposes as the system is already backed up with the lithium battery.

I should have placed the SD card slot on the top of the board. That could have saved me some time during the debugging stage, when I have had some problems with SD firmware that corrupts the SD file system. When I designed the board, I was trying to make it compatible with other platforms, so I included level translators for the SD card interface. I made the mistake of placing a level translator at the master input slave output (MISO), which caused a conflict in the bus with other SPI devices. I removed it and wire-wrapped the I/O lines.

Another issue with this board was the XBee module's serial port net routing, but it was nothing that cutting some traces and wire wrap could not fix. **Photo 2** shows all the aforementioned details and board component location.

FIGURE 2
The communication board schematic is shown.



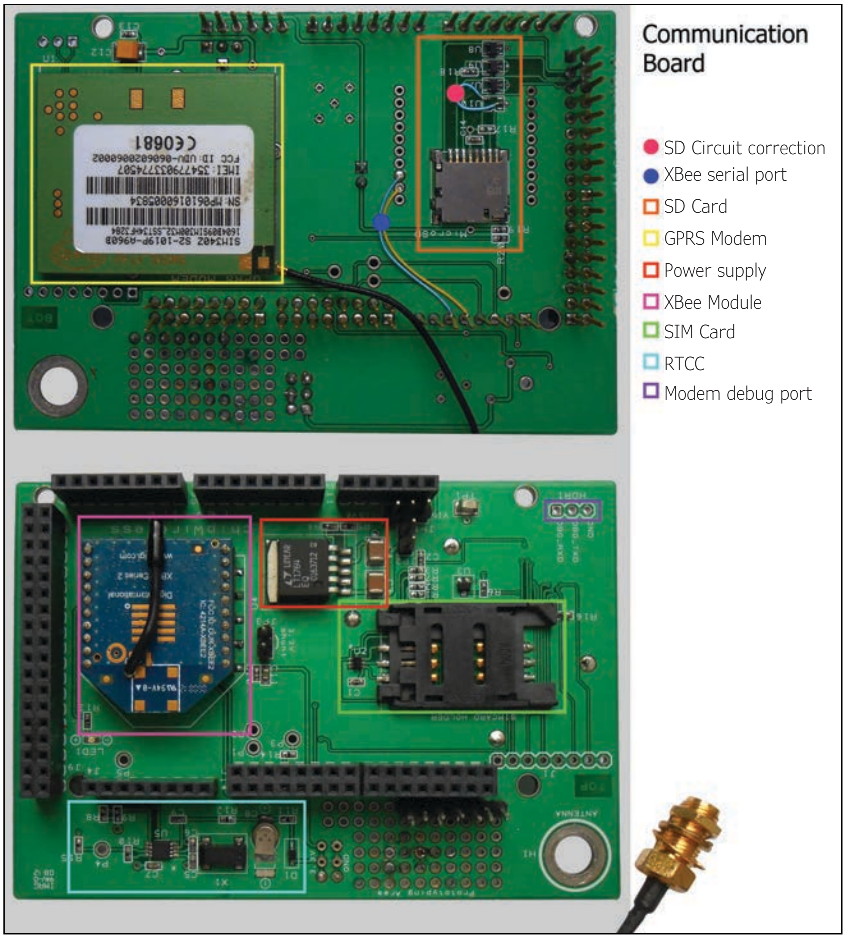


PHOTO 2
This communication board includes several key components to enable wireless communication with sensors, the Internet, and cellular networks.

POST-CONTEST HARDWARE ADDITIONS

I included a web server to have a stand-alone device and provide more flexibility to the home controller. I used WIZnet's embedded Ethernet controller, the W5100 IC, by means of the WIZ811J development board. Using the SPI port and a couple of additional lines connected to a header in the wireless board, I was ready to start coding. I used wire wrap to connect the signals from the header in the prototyping area to the corresponding pins in the chipKIT Max32 board.

Important things to consider here are the multiple SPI devices and the W5100 reset line. I tied the W5100 and PIC32 reset lines together, which enabled me to reset both devices when starting the serial console. Sometimes the W5100 does not reset or seems to hang, so any new design should consider a separate reset line for the W5100.

As for the SPI bus, a W5100 SPI application note explains the need to use the SPI_EN signal instead of the chip select signal if more than one device is on the same bus. Fortunately, the WIZ811J board has a /SCS signal that is already passed through a NOT gate that connects to the SPI_EN signal, which leads to the same logic as other chip select devices in use.

GPRS MODEM FIRMWARE

I wrote the firmware in C/C++ using Digilent's MPIDE software, as the contest required. It later proved to be very useful since I could adapt some available Arduino libraries to the chipKIT platform. I will describe the home controller firmware's core, which turned out to be the communication with the outside world through the GPRS modem.

All the modem code functions (e.g., initialization, send/receive data, and check SMS) are included in their own MPIDE software tab. Serial Port 1 communicates with the modem using AT commands. Almost all the code that accesses the modem is blocking.

Command responses are returned quickly except during initialization, when some commands take a few seconds to complete. In that case, it's a good idea to wait and check that nothing is wrong with modem. In situations such as connecting to a server, the response could take some time and a non-blocking code is used.

The three main tasks with the modem are: connect to a Xively web server, send/receive SMS, and send e-mail. The last task is done by connecting to an SMTP server, as SIM340 doesn't have MMS functionality. I am waiting to get my hands on a SIM900, which has MMS among other improvements.

Meanwhile, connecting to a web server (e.g. Xively) is done manually by executing the AT command `AT+CIPSTART="TCP", "<IP>", "<PORT>"` where you need to replace `<IP>` with the IP address and `<PORT>` with the port number to be connected. After the connection is established, you use the `modemSend(char *data, unsigned int len)` function to send and receive data. This function basically executes the `AT+CIPSEND=<len>` command, where `len` is the number of characters to send.

The SMTP server I used during the tests required me to perform an authorization login authentication. The server expects the username and password in a Base64 encoding. I left those values hard-wired in code for simplicity. After the authentication succeeds, the sender and recipient e-mail addresses are provided. Finally, the DATA keyword is used to indicate the message content transmission.

I used Telnet to ensure the server used the exact phrases to acknowledge the commands. `int ReceiveDataBlock(char findbuffer[])` was implemented to receive SMTP server command responses.

The parameter is a string that will be checked against the response. For example, after sending the `auth login` string to server, the code executes the

FEATURES

ReceiveDataBlock("VXNlcm5hbWU6") function, which basically waits for a server response and looks for the VXNlcm5hbWU6 string, which means "Username:" after it has been encoded in Base64.

With regard to the Xively datastream update, the application connects with the Xively server and sends a PUT HTTP request method with a channel number, a datastream name, and an API key. Once the server processes the request, it sends a 200 OK response if it succeeds.

To check the response, I implemented a non-blocking ReceiveData() function, but I have hard-wired the strings to check. The function works in a re-entrant way invoked from the main loop to get any asynchronous data from the modem. Basically, the modem has a way to output data (e.g., a new SMS or TCP/IP response) and lets the application discern if it's an AT command response or other type of data with a special header. The header for TCP/IP data arrival is a sequence of characters of the form +IPD:<length>, where <length> should be replaced by an ASCII sequence corresponding to the number of characters actually sent in decimal format.

As for the SMS, the same function also checks for any incoming message. If a new SMS arrives, the modem will output through the serial port a string of the form +CMTI: "SM", "<msg>", where <msg> is a number representing the corresponding position in modem memory.

Note the use of a queue to store the incoming message numbers as pending so they can be processed later by another function. I used Efstathios Chatzikyriakidis's QueueList Library for Arduino to implement the queues (see Resources).

Incoming messages are stored in modem memory unless deleted by issuing the AT command AT+CMGD=<index>, where <index> is the SMS number in modem memory. When an SMS is received, only the memory position is saved into the queue and a flag is activated. Later, the int cmd_SMS(void) function extracts it and processes the content.

To send SMS, the send_SMS_Status() function starts by writing the message to modem memory using the first configured phone number in EEPROM. To instruct the modem to send SMS, the AT+CMGW="<phnum>"<cr><message><0x1A> command is used, where <phnum> is the phone number, <cr> is carriage return, and <message> is the message body. Finally, the byte 0x1A is sent to accept the command. That command instructs the modem to send the message and the modem response contains the position of the message in the modem memory. To send the same message to additional numbers, the execution continues to look for configured phone numbers in EEPROM and executes the AT+CMSS=<pos>,"<phnum>" command where <pos> is the previously provided position.

It's necessary to note that EEPROM is used to store phone numbers and e-mail addresses, since that information will not change frequently. Additionally, the EEPROM encoding requires some pointers that are handled using the RTCC IC's battery backup static random-access memory (SRAM).

WIRELESS SENSOR NETWORK

I used Andrew Rapp's Arduino XBee API library, which has proven to be very reliable during my tests (see Resources). The library enabled me to abstract from the XBee ZigBee API implementation so I could focus on my own sensor network library. The final goal was to have a plug-and-play XBee sensor network so non-engineer users can add sensors without having to reprogram the microcontroller.

In general terms, the library enables instantiating an XBee sensor device, defining its I/O channels, and associating the log file name to use and the Xively account information. When new samples arrive, the library extracts the information based on definitions, logs the received samples to a file in the SD card, and creates the HTTP request to send to the Xively

PROJECT FILES

circuitcellar.com/ccmaterials

RESOURCES

J. Celani, "Battery Charger's Unique Input Regulation Loop Simplifies Solar Panel Maximum Power Point Tracking," *LT Journal of Analog Innovation*, January 2011.

E. Chatzikyriakidis, "QueueList Library for Arduino," 2012.

A. Rapp, "Arduino Library for Communicating with XBees in API Mode," *xbee-arduino,* 2009.

Xively, https://xively.com.

SOURCES

Arduino Mega microcontroller
Arduino | http://arduino.cc

chipKIT Max32 Board and MPIDE development platform
Digilent, Inc. | www.digilentinc.com

LT3652 Battery charger IC, LT3479 and LTC3112 DC/DC converters, and LTC4411 IC
Linear Technology Corp. | www.linear.com

MCP6001 Op-amp, PIC32 microcontroller, and MCP79402 RTCC
Microchip Technology, Inc. | www.microchip.com

SIM340 and SIM900 GPRS Modules
SIMCom Wireless Solutions Co., Ltd. | http://wm.sim.com

W5100 IC and WIZ811J Development board
WIZnet Co., Ltd. | www.wiznet.co

website to update a datastream.

The library has several classes organized in a Sensors.cpp file. I haven't had the time to optimize it—or better yet to rewrite it—and it's probably not clean in terms of C++ principles because I come from a C world. Anyway, it works for my demonstration purposes.

Library usage starts by initializing a list of XBee devices in the MPIDE sketch. For each XBee device, the user should fill in some information such as the XBee address, type, parent, channels, datastreams, alarm, comma-separated value (CSV) filename, and Xively feed ID. You can see the details in the SensorNetwork tab, where an instance of `UserSensorsList` class is created. You can then use the aforementioned parameters to call its method `add2list`. The result is that the configured values are saved in RAM for each sensor using an instance of the `SensorData` class named `XbeeSensorPtr` in Sensors.cpp.

The current library status requires initializing all the available sensors at the beginning of the execution. For simplicity, the initialization is also hard-wired in code, but it's just a matter of modifying the web interface so the user can configure them on the fly.

You can define a given sensor type as digital, analog, temperature, or a combination. (I haven't tested the last option, but it should be possible.) Each sensor instance has a method of processing the ZigBee I/O sample that receives and saves the data in a public variable of the class for further processing before the next sample overwrites it.

The alarm configuration consists of a string and an associated value for each channel. The current implementation permits alarms for digital sensors, but with some additional code, it can also be extended for analog. For details, you can look at the `ShowSample` method of `XBeeDeviceBase` class.

Once values have been set for a particular sensor, there is a check for any configured alarm. The configured alarm trigger value is used to compare with a sample value. If there is a match, an alarm message string is created to be sent using SMS or e-mail.

The library uses the channel feed ID and datastream assigned at configuration time to create the Xively PUT update. The datastream and its sample associated value are used to build a string called CSV format in Xively syntax that is an ASCII text of the form `datastream,value`. Several samples can be uploaded this way, but I decided to make an update per sample since they will not be too frequent.

The variable that holds the CSV values is called `XbeeSensorPtr->csv`. The Xively API key required to update a feed ID is a fixed value in a `#define API_KEY_VAR`, so you need to change this value based on your own feed ID.

On the SensorNetwork.pde file, the XBee library will check for any transmissions and call my Sensor library methods, depending on the situation. The two common cases are where I receive a ZigBee Node Identification response (i.e., `ZB_IO_NODE_IDENTIFIER_RESPONSE`). In such a case I am interested in instantiating the sensor as an `XbeeDeviceBase` class and calling the `CreateSensor Method`. When a sensor I/O sample is received (i.e., `ZB_IO_SAMPLE_RESPONSE`), I call the `ShowSample` method.

LESSONS LEARNED

As you have seen, this project has both hardware and software content. (Information on the web server interface and framework is available on *Circuit Cellar*'s FTP site.) DesignSpark software was an easy-to-use tool that covered all aspects my PCB design. Routing the power management board signals required closely following all manufacturer recommendations and application notes. I have to thank Linear Technology engineers for providing support.

Hardware prototyping was also a demanding task requiring both soldering experience and several tools. In the future, I may try to release a commercial version for those who want to get their hands on it without doing the dirty work.

MPIDE proved to be a very useful Arduino-based IDE. If you are entering the Arduino arena with some PIC knowledge, as I was, there is an inevitable learning curve. It helps to turn to Arduino libraries and try to compile them under MPIDE. Even so, I think most readers can handle it quickly and benefit from a fast development time and code reuse.

I hope that some of the main ideas of this project survive and part of the code can be reused by readers. I would like to continue working on the sensor library. Perhaps someone with more C++ experience can join the venture to enrich it. I haven't found any public library with similar goals.

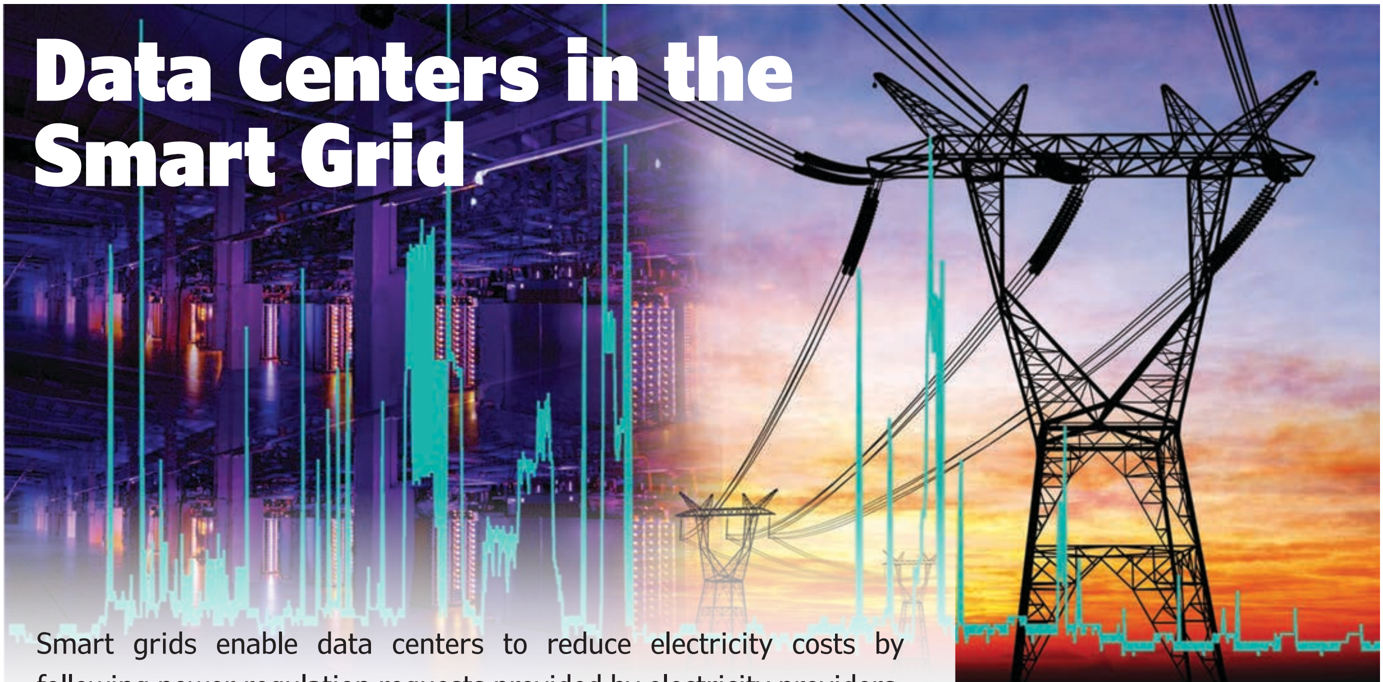
Editor's Note: For other projects from the 2012 DesignSpark chipKIT Challenge, visit http://circuitcellar.com/contests/chipkit2012.

ABOUT THE AUTHOR

Manuel Iglesias Abbatemarco (mhanuel@ieee.org) lives in Caracas, Venezuela. He graduated as an Electronics Engineer from the Universidad Simón Bolívar in 2004. Manuel began his career as a field engineer and is moving toward embedded design.

GREEN COMPUTING

COLUMNS



Data Centers in the Smart Grid

Smart grids enable data centers to reduce electricity costs by following power regulation requests provided by electricity providers. This article discusses these emerging opportunities and explains how data centers can be designed to participate in smart grid programs.

By Ayse K. Coskun (US)

There has been a lot of recent focus on managing data center energy consumption. This is because US data centers now consume more than 3% of the nation's electricity and the consumption is on a rising trend globally. As society moves more of its essential services to the cloud, data centers' capacities and corresponding power consumption will continue to grow.

Data center energy management policies have so far focused on reducing the energy consumption and, more recently, on maximizing the performance under a maximum power budget (i.e., a "data center power cap"). My article "Application-Aware Power Capping" (*Circuit Cellar* 280, 2013) discussed how to maximize application performance on a server while abiding by a dynamically changing server-level power cap. Dynamic power capping strategies are evolving across servers and data centers as ways of managing electricity costs.

A recently emerging topic is related to how a data center can act in accordance with the needs of the electricity provider (i.e., independent service operator, or ISO) rather than solely aiming to minimize its energy consumption. ISOs need to match electricity supply with demand in the power grid in real time. This is a rather difficult problem considering the variations in electricity generation and consumption, and also because of the lack of large-scale energy storage solutions. There have been a lot of developments in building more efficient batteries; however, designing economical and green batteries to store large amounts of energy (as would be needed in a data center) is still a largely unsolved problem.

ISOs' supply-demand matching problem proves to be even more difficult as many countries are moving toward integrating a larger portion of renewable energy generation into their grids. Renewables (e.g., wind or solar power) are intermittent (i.e., there is significant variation in how much energy is available at a given time). Thus, ISOs face significant challenges in maintaining stable grid operation while introducing more renewables into the power grid.

As a result, ISOs are starting to offer incentives for the "demand side" (e.g., a data center) to participate in power regulation programs. One example is the regulation services in the PJM Interconnection wholesale market.[1] In this program, the demand side first qualifies to participate by demonstrating capabilities of increasing/decreasing power consumption at a given rate and of tracking a power consumption value for a given duration.

Upon qualifying, the demand side bids in the hour-ahead market by providing an average power consumption estimate ($P_{AVERAGE}$) and a regulation amount (R). After the bidding is complete and the prices are decided for all participants, the ISO broadcasts a regulation signal every few seconds, which is followed by the demand side. Specifically, at a given time t, the demand side should consume $P_{AVERAGE} + R \times z(t)$, where z(t) is the broadcasted regulation signal with a value between [–1,1].

If the tracking error is above a given threshold, then the demand side loses its contract with the ISO. Thus, a data center should closely "track" a dynamically changing power cap to continue participating in the program. Upon successful tracking with a small error margin, the demand side pays the cost of consuming $P_{AVERAGE}$ but is reimbursed for the regulation amount, R.

For example, a data center with a 10-MW average power consumption and ±2 MW of regulation capability would receive around a 20% reduction in its electricity cost (assuming similar pricing for $P_{AVERAGE}$ and R). Another important feature is that the reimbursed credit is a statistical function of the error: the lower the tracking error, the higher the credit. Thus, there is a considerable cost reduction motivation for designing a data center that can follow an ISO's regulation signal as closely as possible.

How can a data center participate in such programs? First, the data center servers should be able to support fast power regulation. However, hardware alone cannot address all the requirements because a data center's real-time power regulation indicates many levels of integrated control. For example, the power needs to be budgeted between computational and cooling units in the data center, the computational power budget needs to be allocated onto individual servers, and the user quality-of-service (QoS) constraints need to be met in tandem with the power tracking. The remainder of this article discusses how data centers can be designed to participate in regulation service programs in the smart grid.

USING SERVERS FOR POWER TRACKING

As a preliminary step in solving the data center power tracking problem, let's assume the cooling power is always set at a level to maintain safe temperatures in the data center and the center is performing power tracking using only the servers—ignoring the switches, uninterruptible power supply (UPS), and other elements in the data center.

In this case, the servers' goal is to track the power as closely as possible while maximizing the performance. This is indeed similar to the application-aware power capping strategy I described in the previously mentioned article. The server can leverage control knobs such as applying dynamic voltage and frequency scaling (DVFS) and/or adjusting the CPU resource limits (for virtualized servers).

Without loss of generality, let's assume a single server is the data center, for the sake of simplicity. An initial step is to determine the average power consumption ($P_{AVERAGE}$) and the regulation amount the server can provide (R). $P_{AVERAGE}$ should be set depending on an estimate of the workload arrivals and the QoS requirements. For example, one would need to set $P_{AVERAGE}$ at a higher value for a busier server with higher QoS needs.

R, on the other hand, is mostly dependent on the hardware constraints. This is because $P_{AVERAGE}$ has a dominant impact on the overall throughput and the regulation signal z(t) has a mean value of 0 over a 1-h period (so the data center indeed consumes a power consumption value close to $P_{AVERAGE}$). If $P_{AVERAGE}$ is estimated as 100 W and the maximum power this server can achieve at the highest DVFS or resource limits setting is 130 W, then R is going to be maximum 30 W, as the server cannot exceed 130 W in any case.

My students and I computed the $P_{AVERAGE}$ and R using exhaustive search for a server with a 12-core AMD Magny-Cours processor that runs a heterogeneous set of applications selected from the PARSEC 2.1 benchmark suite (one application is processed at a time). Our initial results for various target QoS rates demonstrated close to an R value of 30% of $P_{AVERAGE}$, which leads to a 30% reduction in the electricity cost.[2] Once we determined $P_{AVERAGE}$ and R, we used a sample PJM regulation signal and applied dynamic power

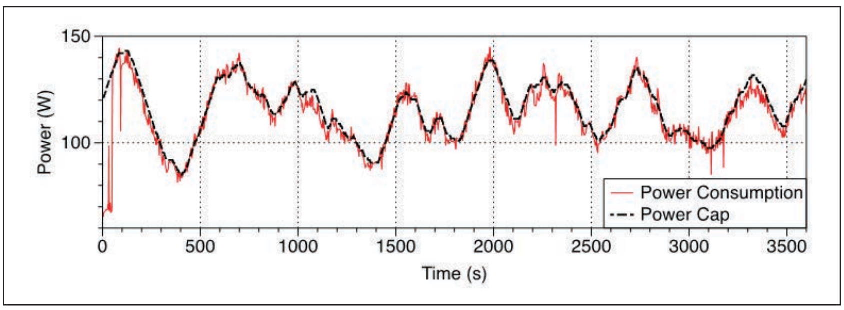


FIGURE 1
These power capping results are from a single server with an AMD Magny-Cours processor that is running a selection of PARSEC multi-threaded applications (one application at a time, but applications vary over time). Over a large set of experiments using various workload arrival rates, my students and I observed around 7% average tracking error, which is well below the error thresholds provided in PJM regulation service programs.

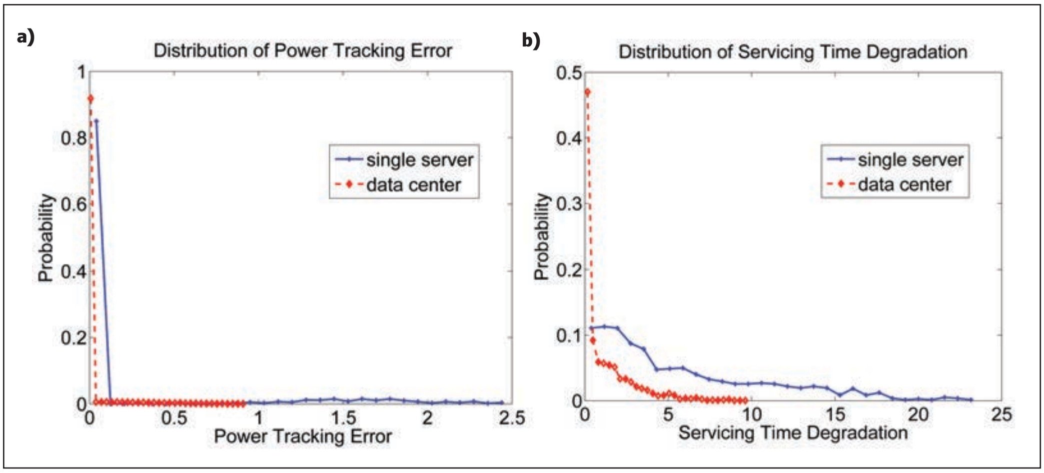


FIGURE 2
These graphs show the distribution of power tracking error **(a)** and workload servicing time degradation **(b)** for single-server and 100-server data center scenarios. Both the error and the servicing time degradation drop significantly as the number of servers are increased, indicating larger electricity cost savings and higher application performance.

capping (by adjusting the CPU resource limits in the hypervisor accordingly) on the server. **Figure 1** shows that the server can track the dynamically changing power cap with high accuracy.

SERVER PROVISIONING

Different from a single-server scenario, a data center has a number of servers, which could be at active (running an application), idle (on but not running any loads), or low-power sleep states at a given time. While these server states make the problem more complex and challenging, they also provide additional degrees of freedom to more accurately solve the power tracking problem, to increase the regulation amount R, and to achieve higher QoS.

Servers' sleep states are especially useful for saving energy, as a sleeping server consumes dramatically less power compared to an idle server. However, going in and out of sleep states has energy and performance costs. In particular, the wake-up period could take tens of seconds or minutes, depending on the specific server, during which time the server consumes close to its peak power value.

Considering the benefits and costs of putting servers into Sleep mode, one would ideally want to keep active servers closer to their peak power and process as much workload as possible. It would also be beneficial to keep most idle servers in a Sleep state, but at the same time minimize the number of transitions in and out of sleep states. In this way, it would be possible to maximize the data center's efficiency.

In my research lab, we recently put together a linear optimization problem to solve the "server provisioning" problem, where we decide on the number of active, idle, and sleep states while closely tracking the regulation signal provided by the ISO.[3] To improve the data center's QoS, we designed some rules of thumb related to performance. For example, we assigned a minimum power consumption level to an active server so the server always continues processing its workload at a minimum rate.

By solving the server provisioning problem in conjunction with applying server-level power capping for a cluster of 100 servers, we were able to improve the center-level regulation amount $R_{DATACENTER}$ to more than 50% of $P_{AVERAGE-DATACENTER}$ (compared to our earlier results that provided up to 30% of cost reduction).

Figure 2 demonstrates how the tracking error probability and servicing time degradation (i.e., the opposite of QoS) drop going from a single-server power-tracking scenario to a 100-server power tracking scenario. I expect that as we scale up the number of servers, the potential cost savings and the overall QoS would improve even further.

circuitcellar.com/ccmaterials

REFERENCES

[1] PJM, "Market-Based Regulation," www.pjm.com.

[2] H. Chen, C. Hankendi, M. C. Caramanis, and A. K. Coskun, "Dynamic Server Power Capping for Enabling Data Center Participation in Power Markets," Proceedings of the International Conference on Computer Aided Design (ICCAD), 2013.

[3] H. Chen, M. Caramanis, and A. K. Coskun, "The Data Center as a Grid Load Stabilizer," Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), 2014.

[4] C. Isci, S. McIntosh, J. Kephart, et al., "Agile, Efficient Virtualization Power Management with Low-Latency Server Power States," Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA), 2013.

RESOURCE

A. K. Coskun, "Application-Aware Power Capping," *Circuit Cellar* 280, 2013.

A factor that would enhance the energy efficiency as well as the power tracking accuracy in data centers is to build servers that can transition in and out of sleep states as fast as possible. Unlike embedded systems, many servers in today's data centers do not have fast transition capabilities to and from sleep states. These servers can be put in a deeper sleep mode and rebooted if needed, but the reboot process takes several minutes. We used a fast sleep state such as IBM's low-latency S3 states.[4] With a 10-s reboot time, we achieved more than 50% electricity cost savings in our experiments. Using a deep sleep state with a 200-s reboot time caused the savings to drop to 35% because of the larger tracking error.

COMPUTATIONAL VS COOLING ENERGY

The power tracking approaches I have described so far focus on regulation via modulating only the computational power without any control of the cooling power. However, a data center's cooling power typically consumes 30% to 50% of the total power.

There are several challenges related to regulating the cooling power. First, the thermal time constants associated with the servers are large (i.e., on the order of tens of seconds or minutes) and there are physical constraints related to how often the computer room air conditioners (CRACs) can be modulated. Therefore, it may not be possible to throttle cooling power every few seconds in a meaningful way.

Second, while regulating the cooling power, the servers' "redline" temperatures should not be exceeded. That means for every computational power consumption profile across the servers, one needs to compute the minimum cooling power needed to maintain reliable temperatures and budget the cooling power accordingly.

One strategy for integrating cooling power into the regulation services is budgeting cooling power every few minutes to maintain stable temperatures below critical levels and bidding in the power market considering the contribution of the cooling, but performing the required fine-grained power tracking only using the servers. Another strategy is to participate in slower-timescale markets with the cooling power component, while participating in the regulation service program with the computational power.

FUTURE DATA CENTERS

Figure 3 shows various components in a data center that is integrated into the smart grid. Load forecasting is necessary for accurate estimation of the average power consumption and the regulation capabilities. At runtime, the data center's power consumption should be budgeted across various computational and cooling elements to meet the given instantaneous power consumption requirements. This budgeting requires collecting telemetry from the servers and cooling elements to be able to adapt to changes in workload dynamics.

An obvious challenge is designing scalable measurement and control approaches in the data center, since data centers have thousands of servers. For example, it could be necessary to implement distributed budgeting schemes and/or multiple levels of centralized power budgeters to ensure timely load servicing.

So far, my students and I used homogeneous servers in our work on data center regulation services. This assumption is realistic for specific clusters in a data center because a data center administrator would often purchase a set of the same type of servers from the same vendor.

However, the data center as a whole includes servers from many vendors with significant differences in the servers' power and performance characteristics. Therefore, job allocation and scheduling of heterogeneous software on a large set of heterogeneous hardware are essential problems to be solved for automating the data center power control. In fact, software-defined data centers, where data center services are managed by intelligent software with minimal human involvement, are major innovation spaces and reducing the energy cost is one of the central aspects of the data center research.

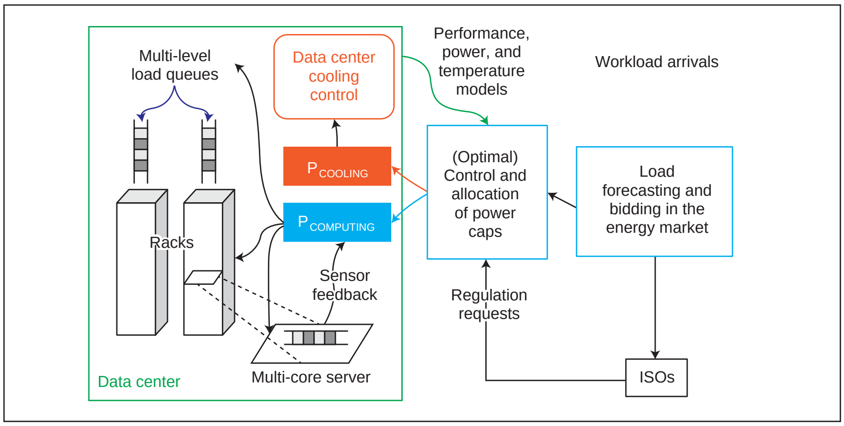
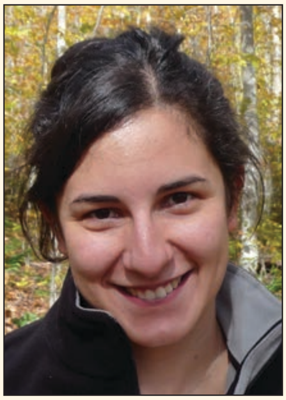


FIGURE 3
This is an overview of data center components and mechanisms that would enable integration of the center with the emerging power market programs.



ABOUT THE AUTHOR

Ayse K. Coskun (acoskun@bu.edu) is an assistant professor in the Electrical and Computer Engineering Department at Boston University. She received MS and PhD degrees in Computer Science and Engineering from the University of California, San Diego. Ayse's research interests include temperature and energy management, 3-D stack architectures, computer architecture, and embedded systems. She worked at Sun Microsystems (now Oracle) in San Diego, CA, prior to her current position at BU. Ayse serves as an associate editor of the IEEE *Embedded Systems Letters*.

COLUMNS

THE CONSUMMATE ENGINEER

COLUMNS

Wireless Data Links (Part 3)

Receivers and Recovery

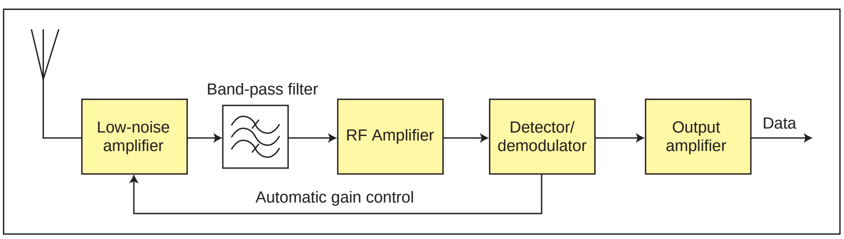
The first two parts of this article series introduced the topic of wireless communications and discussed radio communications and low-power transmitters. This article focuses on low-power data receivers, noise figures, and data bit recovery.

By George Novacek (Canada)

The antenna reciprocity theorem I discussed in Part 2 of this article series ("Transmitters and Antennas," *Circuit Cellar* 284, 2014) applies to receiver and transmitter antennas. However, there may be different challenges for each.

To a receiver, an antenna must deliver a sufficiently strong signal for it to be processed without error. There is an old ham radio maxim: A good antenna is the best amplifier; for no amplifier or processing in the world can help if the antenna does not deliver a signal with the signal-to-noise ratio (SNR) a receiver needs.

FIGURE 1
Homodyne receivers feature direction amplification.



SIGNAL-TO-NOISE RATIO

The minimum required SNR depends on many factors, including the receiver characteristics, the signal modulation and encoding, bandwidth, and the maximum baseband frequency or baud rate. This then determines the receiver sensitivity. For example, Micrel specifies the sensitivity of its MICRF218 receiver with its recommended antenna as –108 dBm for a 1-kilobaud-per-second (KBps) data exchange.

A signal delivered by an antenna contains noise. The signal plus noise is amplified by a number of amplifier stages and then processed. All electronic components and circuits generate noise that is added to the noise appearing on the antenna terminals. A receiver system's total noise figure F is calculated as:

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 \times G_2} + \ldots + \frac{F_N - 1}{G_1 \times G_2 \ldots \times G_{N-1}}$$

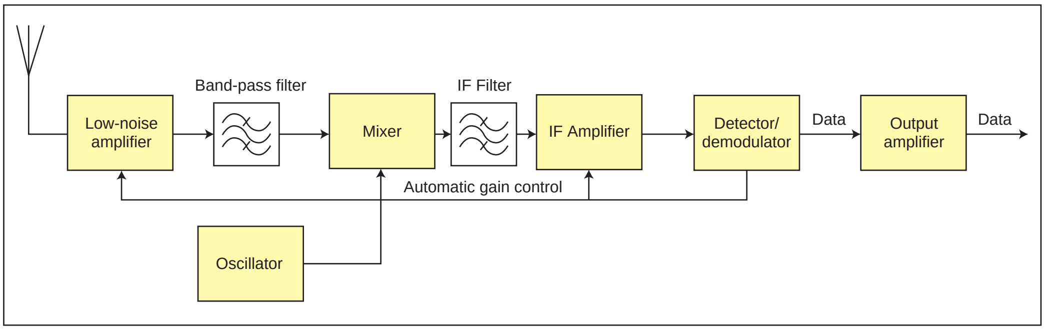


FIGURE 2
Superheterodyne receivers convert to intermediate frequencies.

where F_1 is the noise appearing on the receiver's antenna terminals and F_2 is the noise arriving at the input of the second stage from the first stage, which is the noise arriving from the antenna plus the noise generated by the first stage, and so on. G_1 is the first stage gain, G_2 is the second stage gain, and so forth.

The equation makes it evident that for the lowest overall noise, the first stage is the most important. It should have as low a noise figure and as high a gain as possible. With that, the noise generated by the second and the following stages can become negligible. This principle applies to all weak signal amplifiers regardless of their bandwidth, not just to RF receivers.

RECEIVERS

Rather than building one, you will likely purchase a ready-made receiver module. In this case, the manufacturer should have taken care of its optimal noise figure. In some cases, you may be able to improve the receiver performance by adding a preamplifier to boost the antenna signal or to counter the signal-to-noise degradation caused by its long lead. A lead from the antenna to the receiver attenuates the signal and also adds its own noise.

Because the 300-to-900-MHz low-power data transmission frequencies fall into the spectrum covered by commercially available TV antenna amplifiers, you can purchase one for placement directly on the antenna (this is best) or as a preamplifier connected to the receiver's antenna terminals. Those preamplifiers were popular when most people received television with their own antennas. The preamplifiers became a commodity available from many discount stores.

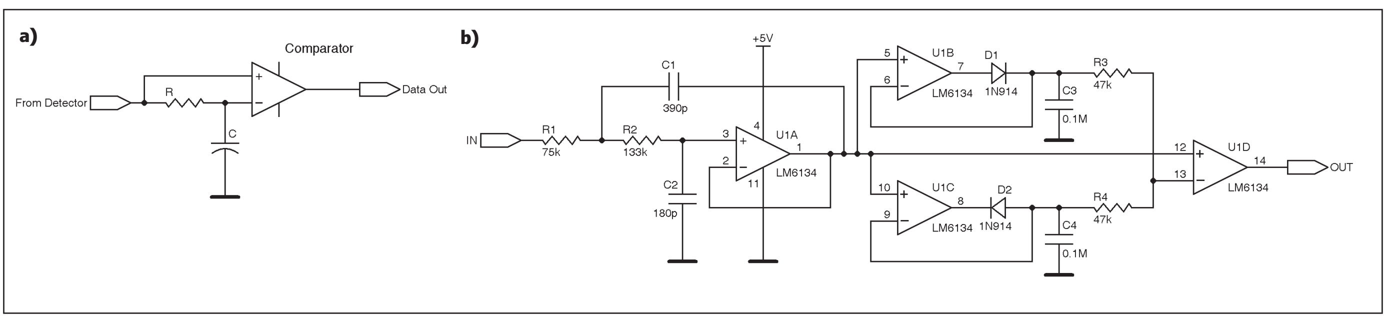
When purchasing a preamplifier, keep the previously mentioned noise figure equation in mind. Unless a preamplifier has better noise figure than the receiver front end, it will do no good when located at the receiver. However, it may offset antenna lead losses when mounted on the antenna. Some commodity preamplifiers are cheaply made and will degrade a modern receiver's performance.

Multipath distortion or multipath fading is prominent in very high frequency (VHF) and ultra-high frequency (UHF) bands. It degrades and even destroys your wireless data link's signal integrity. It is caused by the antenna directly receiving not just the transmitted signal, but also its reflections off the ground and other objects in its path.

The reflected signals arrive at the antenna with different phases and all combine with the direct signal. As a result, the signal seen by the receiver antenna has distinctive peaks and valleys of strength. A strong signal in one place may disappear just a few inches away. When the reflective surfaces move (e.g., people, cars, and airplanes), the peaks and valleys move accordingly. You may be able to rectify the problem by relocating the antenna or using a more directional one (see my article "Impedance Matching," *Circuit Cellar* 281, 2013). Other solutions include frequency hopping or sweeping, or antenna diversity. I'll provide more information about this in Part 4 of this article series.

There are two basic types of receivers:

FIGURE 3
The principles of a data slicer **(a)** and a high-performance data slicer **(b)** are shown.



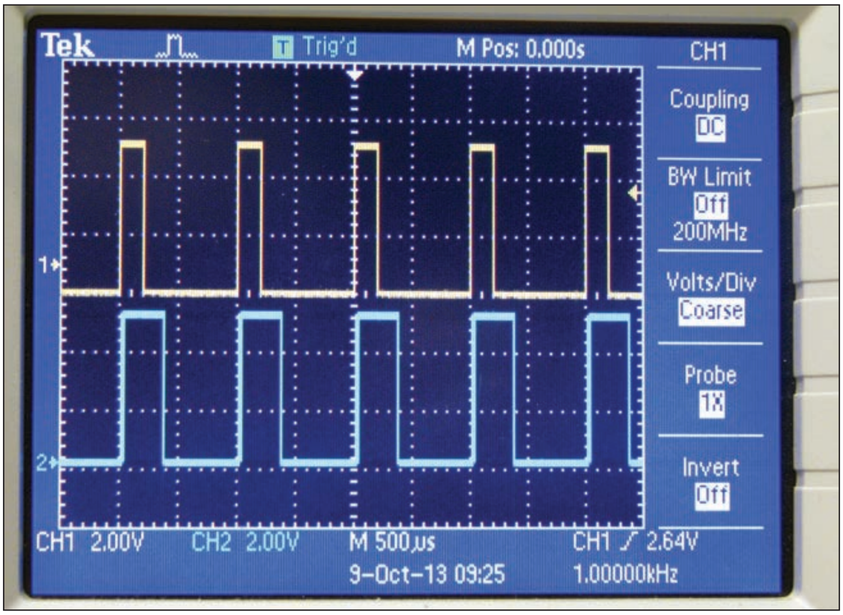


PHOTO 1
The transmitted signal is the yellow trace. The receiver output is the blue trace.

those with direct amplification (i.e., homodynes) and those with conversion to intermediate frequencies (i.e., heterodynes or superheterodynes, or, more commonly, superhets).

HOMODYNE RECEIVERS

Figure 1 shows a typical homodyne receiver's block diagram. One is C-MAX Time Solutions's CMMR-6P IC receiver for WWVB time signal. It works with a ferrite stick antenna resonating at 60 kHz feeding a low-noise amplifier (LNA) followed by a 60-kHz ceramic resonator high-Q band-pass filter to give the receiver high selectivity. Another amplifier stage comes next feeding a peak detector followed by a low-pass filter and the output signal buffer. This monolithic receiver needs only one ceramic filter, the ferrite antenna stick, and a couple of capacitors external to the IC.

Homodyne receivers, some of which exploit negative resistance, work well in the VHF and UHF bands. The ubiquitous super-regenerative receiver is one of them. It can be built with two to three transistors and a comparator, which makes it a popular choice for inexpensive systems.

The super-regenerative receiver has some shortcomings, but its sensitivity reaches the noise floor and works quite well with data in simple systems (e.g., keyless entry). SparkFun Electronics's WRL-10532 receiver is an example. More details about this are available on SparkFun Electronics's website and in my article "The Super-Regenerative Receiver" (*Circuit Cellar* 248, 2011). Monolithic superhet receivers (e.g., Micrel's QwikRadio RF IC) now rival super-regenerative receivers in simplicity, better performance, and competitive cost.

SUPERHETERODYNE RECEIVERS

Figure 2 shows a superhet receiver, which is a bit more complicated than a homodyne receiver. The antenna's RF signal, which is usually impedance-matched by a tuned circuit, is amplified by an LNA that is band-pass filtered and fed into a mixer. The mixer combines the received signal with a local oscillator frequency to generate their sum and difference. One, which is often the difference, becomes an intermediate frequency (IF).

For example, a mixer of a broadcast FM receiver tuned to a 100.1-MHz station subtracts this signal from a 110.8-MHz local oscillator's frequency to generate a 10.7-MHz IF. The IF is amplified and the bandwidth and amplitude are limited and then demodulated and buffered. The mixer also produces the 210.9-MHz sum, which is way out of the IF bandwidth and can be ignored. For economical reasons, it is not unusual to combine some stages into one.

Superhet receivers have advantages over homodyne receivers, although some are not needed for fixed-tuned datastream receivers. Due to the IF processing, superhet receivers provide high sensitivity and well-defined selectivity, which may be hard to achieve with homodyne receivers.

Modern monolithic superhet receivers simplify the design while significantly reducing the component count. Some even enable the antenna to be directly connected to the IC, but adding a tuned LC network improves their characteristics. These receivers' sensitivity usually reaches the noise floor.

In general, superhet receivers convert the IF to the baseband (the transmitted data) and homodyne receivers convert the RF to it. A low-pass filter that follows the

circuitcellar.com/ccmaterials

RESOURCES

G. Novacek, "RF Design Review," *Circuit Cellar* 247, 2011.

———, "The Super-Regenerative Receiver," *Circuit Cellar* 248, 2011.

———, "Impedance Matching," *Circuit Cellar* 281, 2013.

———, "Wireless Data Links (Part 2): Transmitters and Antennas," *Circuit Cellar* 284, 2014.

SOURCES

CMMR-6P IC Receiver
C-MAX Time Solutions GmbH | www.c-max-time.com

MICRF218 Receiver and QwikRadio RF ICs
Micrel, Inc. | www.micrel.com

WRL-10532 Receiver
SparkFun Electronics | www.sparkfun.com

ABOUT THE AUTHOR

George Novacek 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. Contact him at gnovacek@nexicom.net with "Circuit Cellar" in the subject line.

detector removes the unwanted noise. In data transmission, the binary datastream needs to be recovered from the baseband, usually by a data slicer (a.k.a., a bit slicer). **Figure 3a** shows the principle.

DATA SLICERS

The data slicer's RC network maintains the average signal value on a comparator's inverting node. The noninverting node follows the incoming signal and produces a rectangular output compatible with digital logic. The RC time constant is critical.

Correct data slicer operation depends on the incoming signal averaging to zero. Herein lies the problem for reception of non-return-to-zero (NRZ) asynchronous serial datastreams. NRZ data decoding depends on precise timing. This, unfortunately, is affected by the data slicer's time constant and the varying signal average.

SparkFun Electronics's WRL-10532 receiver module, which I mentioned before in this series, is inexpensive, but it's a good super-regenerative receiver. A simple software example to set up a data link using Arduino, which is available from SparkFun Electronics's website, didn't work for me. Arduino drives the transmitter with NRZ data (see **Photo 1**). The signal timing is too distorted by the data slicer built into the WRL-10532 module to correctly decode the data.

The WRL-10532 receiver's "analog" pin provides access to the baseband signal. Using the data slicer shown in **Figure 3b**, I was able to fine-tune its timing to obtain a short range, which provided acceptable performance. For reliable maximum-distance operation, the data needs to be encoded so it is largely insensitive to its timing. I will explore this in Part 4 of this article series.

COLUMNS

DesignSpark Tips & Tricks

Day #10: Custom Outline Models

By **Neil Gruending**
(Canada)

Let's import our PCB back into our 3-D model and have a little fun.

Over the last couple of days (which mysteriously lengthened to months in Elektor), we designed a circuit board to fit in a Hammond 1551N enclosure using DesignSpark PCB and DesignSpark Mechanical. Today we'll import our board back into DesignSpark Mechanical and have a little fun along the way.

Import the PCB into DesignSpark Mechanical

We had finished the board outline in the last installment, so now it's time to export an IDF file from DesignSpark PCB by using the "Output->DesignSpark Mechanical (IDF)" menu. This will open the DesignSpark Mechanical IDF export window where you can specify a name for the file and the board thickness. Note that the export will use a default component height of 1 mm unless the components specify a different height in the library. We'll see an example of that later.

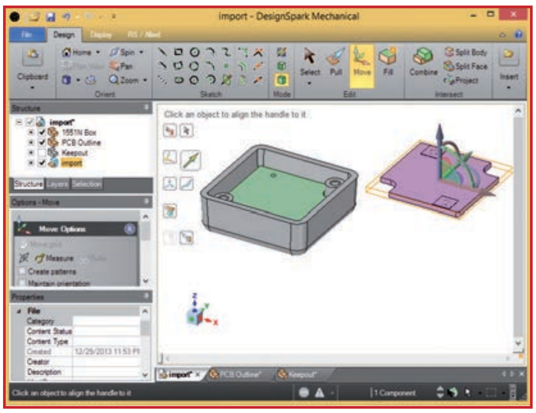
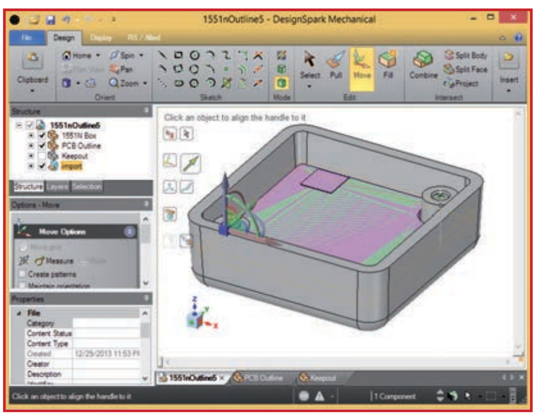


Figure 1. Imported board outline.

Figure 2. Corrected board placement.



Now let's import the IDF into our DesignSpark Mechanical model using the file import tool (Insert tab->File) and you should get something like **Figure 1**. DesignSpark has already selected the board outline for us so now we just need to move it to the right position in the enclosure. This is really easy in our case because we just need to line up the new board outline to our original model. The first step is to move the move anchor point to a board corner using the Anchor tool. Then select the Up To tool and select the same corner on the original PCB outline. Both of these tools are available in move mode in the left side of the drawing window.

The updated model should now look like **Figure 2**. Everything looks fine except that our mounting holes have squares over them. That's because our mounting holes are components on the PCB and DesignSpark PCB applied the default 1mm component height rule to them which gives us the square boxes over the holes. A quick edit to remove the square will actually update both of the holes automatically because DesignSpark Mechanical has also imported the component structure from the PCB. I expanded the model structure in **Figure 3** to show the mounting hole components. The figure also shows what the imported PCB looks like after being cleaned up. Now let's add some components to the board to get a complete rendering of our design.

Adding PCB components to our model

Now let's add some components to our board and see what happens. I chose to add a few SOT23 transistors and a couple 0603 resistors as in **Figure 4** and then imported the board into DesignSpark Mechanical which looks like **Figure 5**. I had problems getting the PCB components to import correctly into DesignSpark Mechanical while I was playing with the playing with the component heights though. The solution ended up being to delete all of the files in the IDF export directory.

Figure 5 also shows an example of what importing a PCB with different component heights looks like in DesignSpark Mechanical. The mounting holes have the small box drawn around them like before and the 0603 and SOT23 patterns are 0.5 mm and 1.12 mm, tall respectively. The trick is that you have to specify the component heights in DesignSpark PCB by adding a value named “Height” to the component properties. I recommend doing that in the component libraries, so that you don’t forget later. My board was already set to metric units so I set the height to the desired value without any units, i.e. I entered 0.5 mm as 0.5.

But what if you wanted to make our imported 3-D PCB look more realistic? The first thing we’ll need are the 3-D models for our components. It’s possible to draw them yourself in DesignSpark Mechanical but in this case I will use STEP models downloaded from 3-D Content Central [1]. DesignSpark also has many 3-D models available as part of Modelsource [2]. The only real requirement for the models is that they use the same coordinate origin and orientation as the PCB components so that the new 3-D models will line up properly. For example, I like to use the center my PCB footprints as the component origin, so I used 3-D models that also used their bottom center point as the origin. Note that DesignSpark Mechanical won’t let you edit STEP models, so sometimes you might have to try several different models before finding one that will work.

Now we’re ready to update the 3-D PCB model. Open the imported PCB with the “Open Component” command, which will open the PCB in its own viewing window. Now you can select the components to change in the Structure window. Right click on it and select “Source->Replace Component” and then choose the 3-D model file you want to use. DesignSpark Mechanical will then exchange the model for the new one and it will also rotate it as necessary. This is why it was important to use models that correspond to the PCB footprint. The final result will look like **Figure 6** after a little bit of editing. Make sure you doublecheck the component 3-D model position if its placement is critical.

Conclusion

Today we used DesignSpark Mechanical to make a 3-D model of our finished PCB. Next time we will focus on using DesignSpark PCB for more complex designs.

Web Links

[1] www.3dcontentcentral.com

[2] www.tracepartsonline.net/(S(rhrqx-sieinz4vp45g4eluh55))/content.aspx?fwsid=DESIGNSPARK&Lang=&P=

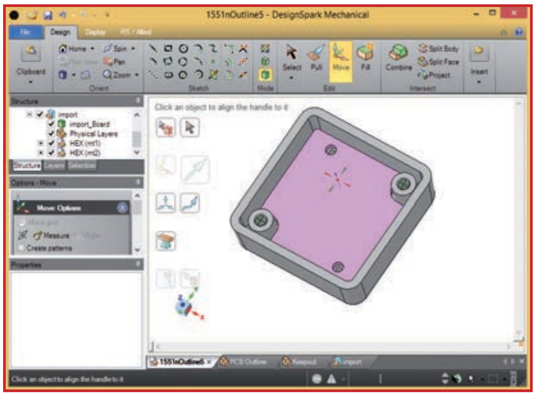


Figure 3.
Finished board import.

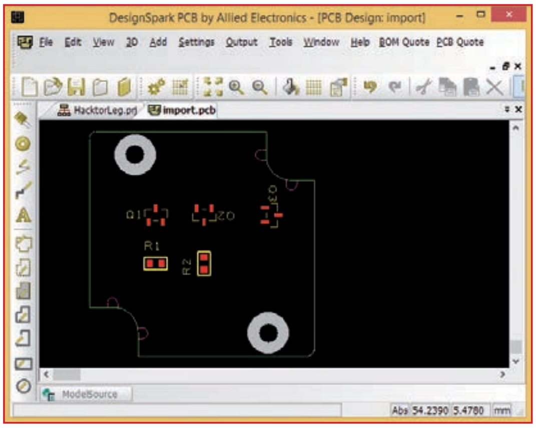


Figure 4.
Components added to PCB.

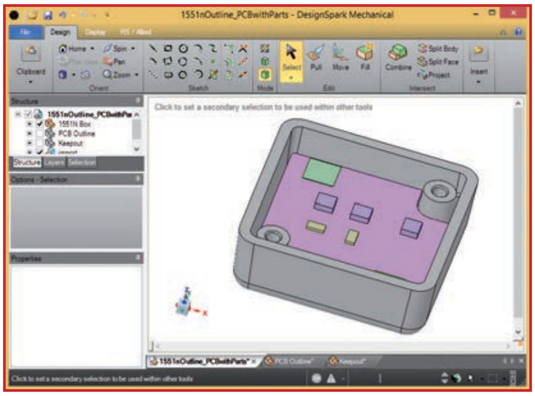


Figure 5.
3-D PCB model.

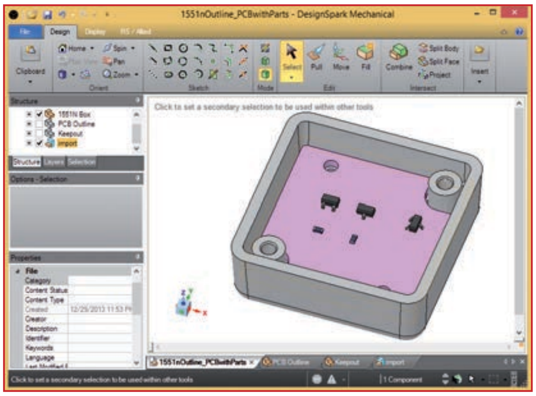


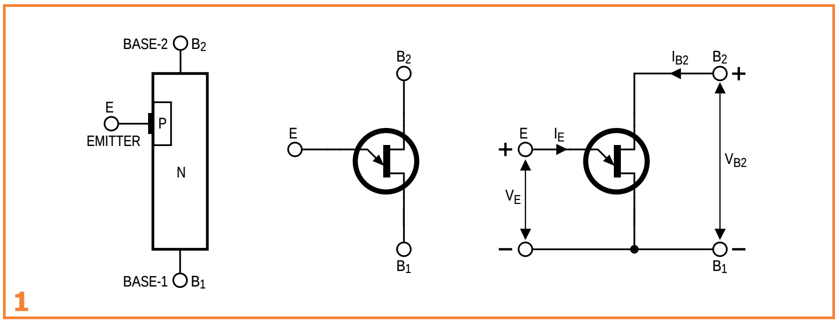
Figure 6.
Final PCB model.

Unijunction Transistors

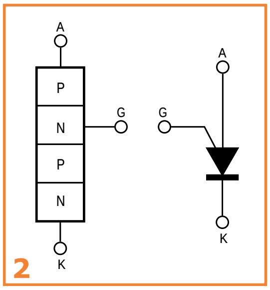
Weird Component #4

By **Neil Gruending**
(Canada)

Mention the word transistor and I bet most people immediately think of the usual bipolar variety. Unijunction transistors (UJTs) aren't common anymore, but a few decades ago they achieved widespread use in low-frequency oscillators and silicon-controlled rectifier (SCR) firing circuits. Let's take a look at how these devices work—and at a modern replacement.



1



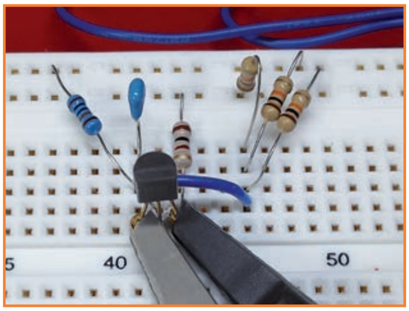
2

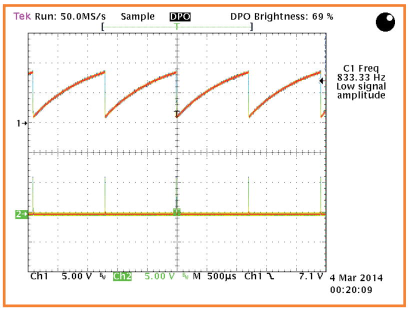
A traditional UJT is a three-pin device with a single (!) PN junction inside. Its construction, circuit symbol and basic circuit arrangement are shown in **Figure 1**. Two of the pins are used for the base connection and are labeled B1 and B2. They connect to either side of a bar of N-type silicon with a well of P-type silicon in it for the third connection called the emitter (E). When the UJT is off, there's a resistance between the base pins, and the emitter acts as a diode. The base construction acts as a voltage divider for the diode so that no current will flow into the emitter until the voltage exceeds the internal base voltage. Once the emitter voltage increases enough to start conducting, the UJT will switch on and create a low-resistance path between the emitter and B1. This switching point is called the peak voltage and the UJT will continue conducting until the emitter voltage drops below the valley voltage threshold. The valley voltage is always less than the peak voltage, which gives UJTs their negative-resistance characteristic and makes them great for triggering from short pulses.

A modern replacement for UJTs are programmable unijunction transistors (PUTs). They operate in a similar manner as a UJT but internally they are an SCR with a four-layer P-N structure. This construction means that PUTs have an anode, cathode and gate connections (**Figure 2**) instead of the usual UJT connections. Just like a UJT, a PUT has a negative-resistance characteristic when the anode voltage exceeds the gate voltage which is programmed with a resistor voltage divider.

I couldn't find UJTs to play with, but I did find some 2N6027s which are a PUT. So I put together a simple relaxation oscillator (**Figure 3**) and measured its output with an oscilloscope (**Figure 4**). Channel 1 in the oscilloscope trace shows the anode voltage and Channel 2 is the cathode voltage. The anode is charged by a RC circuit. When the threshold voltage is reached, the SCR kicks in and discharges the capacitor very quickly, as shown by the cathode voltage pulses. This circuit isn't terribly useful by itself. But if you used it to trigger an SCR, then you could start controlling much larger loads.

UJTs may be almost obsolete, but the fact that they are being replaced by PUTs shows just how useful their functionality can be.





COLUMNS

ABOVE THE GROUND PLANE

Battery Capacity and Aging

Despite the precise numbers printed on the case, a battery's actual performance depends on everything from manufacturer integrity to end-user care. Ed tests several new and old batteries in his collection, presents the sometimes surprising results, and concludes that you might get what you pay for.

By Ed Nisley (US)

PHOTO 1
A 3-D printed fixture connects a Canon NB-5L battery to the West Mountain Radio CBA-II tester.

Many of the battery-powered devices in my collection have outlived their batteries and, in some cases, even the original battery supplier. Despite relentless improvements, batteries remain the weakest and bulkiest part of portable devices, to the extent that they may account for half the volume and most of the weight.

In this article, I'll examine the test results for several batteries in my collection, with data that spans nearly four years. I certainly can't claim that these results justify any grand conclusions, but the numbers provide an interesting counterpoint to the usual vague claims about battery performance.

BATTERY TESTING

The CBA-II (Computerized Battery Analyzer) from West Mountain Radio shown in **Photo 1** has been a key piece of test equipment in my shop for many years. The firmware upgrade required for use with Windows 7 gave it some of the capabilities of the current CBA-IV and improved its voltage and current resolution. It connects to the battery-under-test through Anderson Power Products's Powerpoles, with provision for a separate Kelvin sense lead at the load. The large aluminum heatsink and fan atop the CBA can dissipate up to 150 W at 40 A from a 55-V source.

Although ordinary cylindrical cell holders suffice for AA and AAA NiMH cells, **Photo 1** shows the 3-D printed holder required for the prismatic Canon NB-5L batteries in my SX230HS camera. It appears that every camera uses a different type of battery, so my first "battery test" seems to be designing and building Yet Another Holder with dependable connections to the battery terminals. Those thin wires suffice for low test currents, but obviously can't handle the maximum current through the 13-AWG wires on the CBA side of those Powerpoles.

The CBA draws a constant load current from the battery, regardless of its terminal voltage, so that the product of the current in amps and the elapsed time in hours gives the energy in ampere hours supplied by the battery. Conversely, dividing an energy value in A·h by the load current gives the elapsed time.

Cameras and other contemporary devices using switching power supplies act as constant-power loads that draw increasing current as the battery voltage decreases. The CBA cannot mimic a constant-power load, but, for most batteries, the difference doesn't create much error, because a good battery produces a relatively constant voltage during most of its discharge. As a result, you can estimate the energy capacity with reasonable accuracy by multiplying the median voltage by the load current and test duration.

Because a battery's ability to deliver energy decreases as the load current increases, its nominal capacity may have little relation to the actual performance in a real device. The usual "20-h capacity" comes from the current that discharges the battery to a specific voltage after 20 h. For example, a 3.7-V lithium-ion (Li-ion) battery with a nominal 1.0-A·h capacity can supply a constant 50 mA during those 20 h:

$$50 \text{ mA} = 0.050 \text{ A} = \frac{1.0 \text{ A} \cdot \text{h}}{20 \text{ h}}$$

Using that battery in a camera drawing a continuous 250 mA will produce a *much* shorter run time than the 4 h predicted by the formula:

$$4 \text{ h} = \frac{1.0 \text{ A} \cdot \text{h}}{0.25 \text{ A}}$$

The actual capacity reduction depends on the battery chemistry, cell structure, internal temperature, and a myriad other factors in addition to the current. Classic lead-acid cells have a roughly inverse-exponential relationship between discharge current and total run time, as defined by Peukert's law, but finding the coefficients for a similar equation applicable to other battery chemistries isn't practical for most applications.

As a result, measuring the battery capacity by applying a current roughly equal to the expected load produces the most useful results. The plots you'll see in this article generally show a battery capacity less (in some cases, much less) than the nominal value, but that value serves as a better prediction of actual performance.

Because the data behind these plots represents batteries bought years ago, I haven't included many brand names. You may safely assume major manufacturers, the ones producing batteries bearing brand names you recognize, have consistent quality that's missing from the no-name offerings.

NIMH AA CELLS

My Sony DSC-H5 camera places a heavy demand on its pair of nickel-metal hydride (NiMH) cells, with the inevitable result that the OEM cells failed all too soon. Random NiMH cells from my collection also fared poorly, so I bought eight Sanyo Eneloop cells that combine good capacity with low self-discharge. The red curve in **Figure 1** shows the as-received results for a single cell at 500 mA before its first charge; all eight cells produced essentially identical results. The remaining "factory charge" held about 0.75 of the rated capacity and the cells were truly

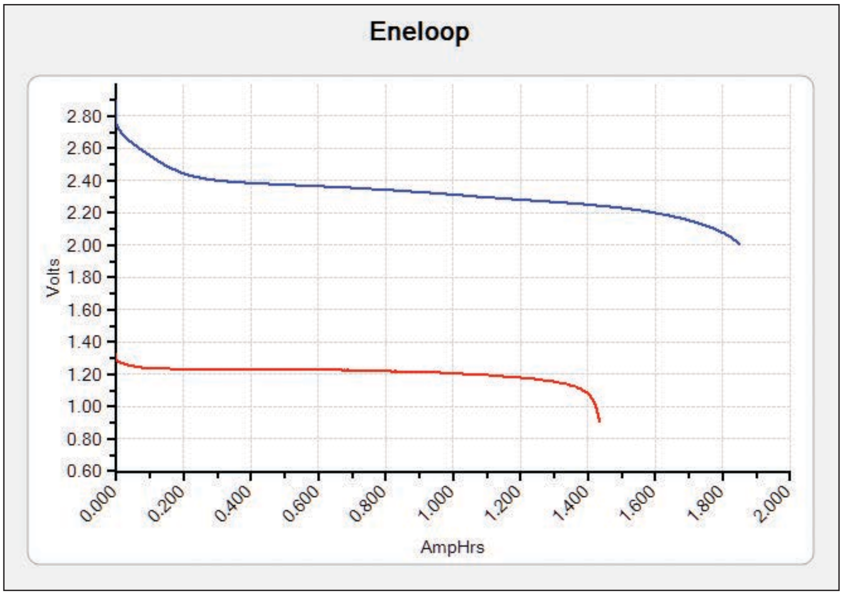


FIGURE 1

A set of eight Sanyo Eneloop 2.0-A·h NiMH AA cells has delivered dependable performance. The red trace shows a single cell's as-delivered capacity before its first charge and the blue trace shows a freshly charged pair of cells three years later, both at 500 mA. The run time to 1 V/cell remains near the rated 2.0-A·h capacity, but the voltage declines more rapidly during discharge.

COLUMNS

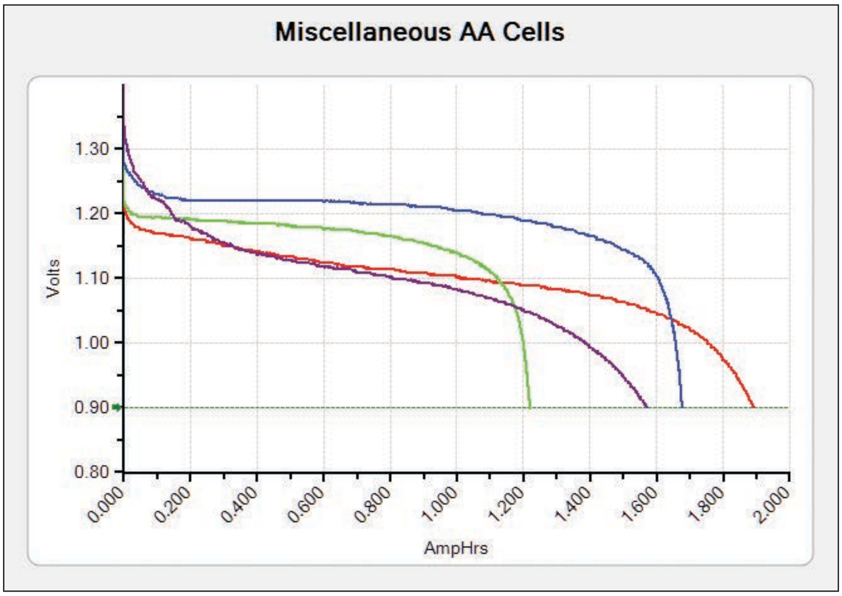


FIGURE 2
The test results for a random assortment of NiMH AA cells, all rated between 2.0 and 2.4 A·h, suggest how little the nominal ratings mean. Although the cells producing the purple and blue traces have similar capacity, the former would produce the dreaded "Low Battery!" warning much sooner. All tests are at 500 mA.

ready to use.

After the first charge, all eight cells together produced 1.9 A·h at 500 mA, very close to their 2.0-A·h rating for 100 mA over 20 h. That data came from an eight-cell pack and would be off-scale high on this plot, but shows the same endpoint.

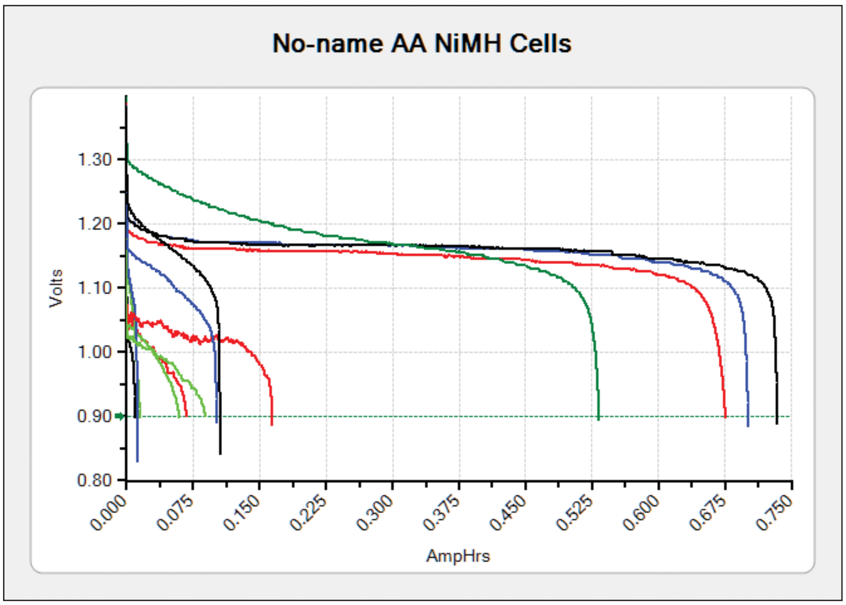


FIGURE 3
The performance of this quartet of exceedingly cheap NiMH AA cells also improved during their first four charges, but they deliver less than a third of their rated 2.3- A·h capacity at 1 A. Name-brand cells performed much better at this relatively heavy load.

As a general rule, I use a pair of cells until the camera first warns of a low battery, and then install the next pair in line, which means those Eneloop cells have marched through the camera and charger in an orderly stream. The blue trace in **Figure 1** shows the latest results for one pair at 500 mA; the curves for all four pairs are indistinguishable.

Their original 1.85 A·h to 1.1 V/cell capacity has declined to 1.6 A·h, but, because the camera determines the charge state by measuring the voltage, they don't provide as much run time as before. After three years, that's understandable!

Figure 2 compares the widely varying results for four cells from second-rank companies, the ones with less-familiar names, all tested at 500 mA. Despite bearing ratings from 2.0 to 2.6 A·h, the actual run times would be far less than that. For example, the cells corresponding to the purple and blue traces end with 1.55 and 1.8 A·h at 0.9 V/cell, but the purple cell reaches 1.0 V/cell at 1.4 A·h; it would indicate "low battery" long before the blue cell. The purple cell would last half as long as the blue cell in a fussy device that required 1.1 V/cell.

Even those cells look good compared with the quartet in **Figure 3**, which lacked a manufacturer's name and weighted noticeably less than other AA cells. Despite proudly proclaiming "1.25 V @ 2.3 AH," their actual test results at 1.0 A show poor performance. They had essentially no capacity at all after their first charge and, after four charge-discharge cycles; three of them had improved to less than a third of their rated capacity. Perhaps they would fare better in a low drain device, but they went directly to my recycling heap.

The era of NiMH AA cells seems to be drawing to a close, as nearly all new devices use batteries based on Li-ion technology.

LITHIUM-ION BATTERIES

Lithium-based cells have about twice the energy density of NiMH cells and, because the cells can be both thin and rectangular, lithium batteries fit neatly into currently fashionable rounded-rectangle devices. Unfortunately, that means there are nearly as many different lithium battery types as there are devices, with essentially no standardization between manufacturers.

In fact, there's little standardization within each manufacturer, as exemplified by the NB-5L and NB-6LH lithium batteries used in several Canon cameras. Despite having nearly identical 1.1-A·h capacity, the two batteries have slightly different dimensions and are not interchangeable.

I recently purchased a Canon S120 camera that uses NB-6LH batteries. The blue curve in

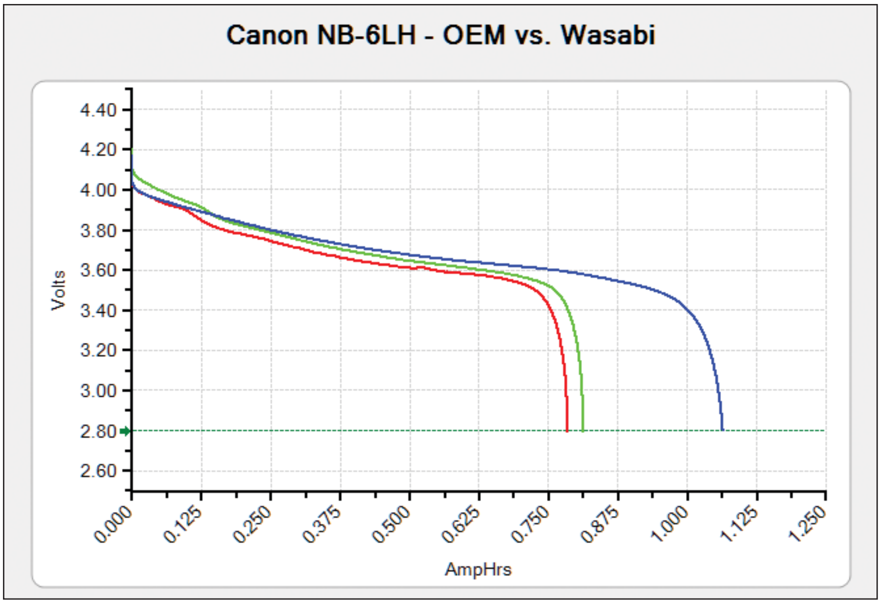


FIGURE 4
Measured at 3.6 V, the Wasabi lithium-ion battery packs have about 80% of the capacity of the genuine Canon NB-6LH pack. All three batteries were fresh from their first charge and bear a 1-A·h rating.

Figure 4 shows that the fully charged Canon battery delivers almost exactly its rated 1.1-A·h capacity when discharged to 2.8 V at 250 mA. Unfortunately, the camera won't function when the battery voltage drops below 3.6 V, making the nominal rating irrelevant: the battery delivers only 750 mA·h to the 3.6-V cutoff.

Although the 250-mA test current may seem high when compared to the 20 h rate of 55 mA, that's the idle current of my older Canon SX230HS camera, so it's in the right ballpark. Building a dummy battery that fits into the camera's compartment poses some challenges that aren't relevant here, but that's the best way to measure the actual current.

Along with the S120 camera, I bought a pair of Wasabi Power BTR-NB6L-JWP batteries that carried an "ultra high" 1.3-A·h rating. The red and green curves in **Figure 4** show that the pair actually deliver 800 mA·h down to 2.8 V

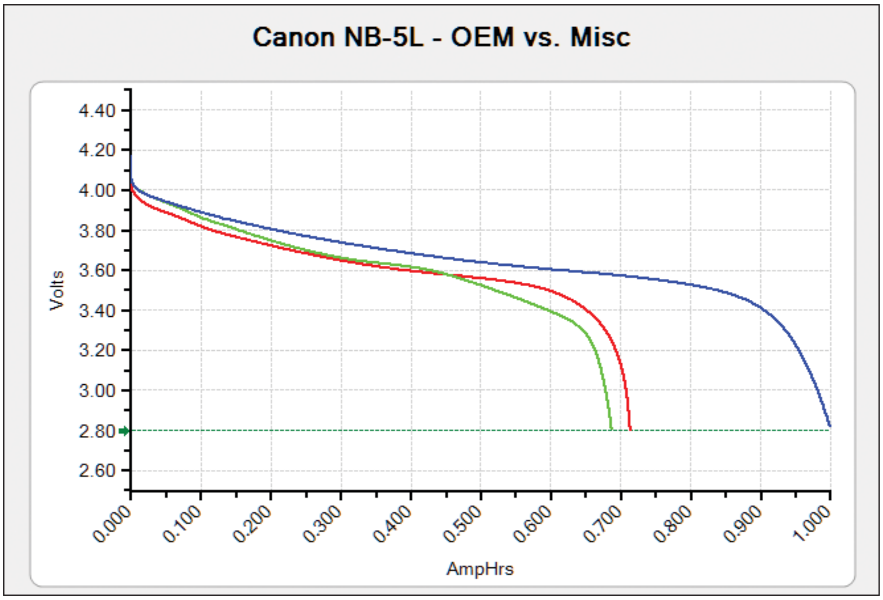


FIGURE 5
After two years, two surviving generic batteries provide barely two thirds of the capacity of a genuine Canon NB-5L pack at the 3.6-V level. Two other generic batteries completely failed after a few months.

COLUMNS

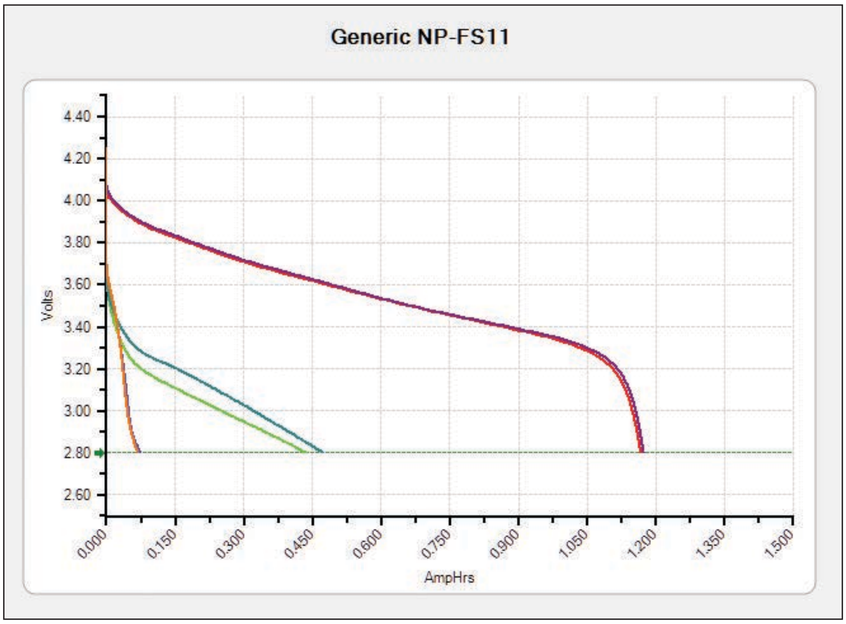


FIGURE 6
The first two charges on three cheap generic NP-FS11 batteries showed extremely poor performance: two are effectively dead and the third has a steep voltage drop that flags it as empty in short order.

at 250 mA and barely 500 mA·h to 3.6 V. In round numbers, the usable capacity under a realistic load amounts to 40% of the 20-h rating.

While it's certain that both battery brands would deliver more energy at the lower 20-h discharge current, a factor of two improvement isn't likely. The Canon rating seems much more forthright.

However, the direct-from-Canon price for a genuine NB-6LH battery is $60 and the Wasabi Power BTR-NB6L-JWP batteries cost $9 each. Under my test conditions, the Canon battery delivers 13 mA·h/$ and the Wasabi delivers 56 mA·h/$ when discharged to 3.6 V, so, even though the Wasabi batteries offer only 67% of the Canon battery's performance, they cost 15% of its price.

You'll find warnings about using non-OEM batteries in your devices, generally issued by the OEM. I cannot attest to the safety or compatibility of generic batteries, so you must make those decisions on your own.

With that in mind, **Figure 5** shows some interesting lifetime data for the NB-5L batteries in my Canon SX230HS camera. The blue curve indicates that, after two years, the Canon battery still delivers 1.0 A·h, essentially identical to its 1.030-A·h rating, when discharged at 250 mA to 2.8 V. At the camera's 3.6-V minimum level, it delivers 600 mA·h, 80% of the 750 mA·h from the nearly identical and new NB-6LH in **Figure 4**.

That's definitely untrue of the four no-name "generic" batteries I bought at the same time, choosing four eBay vendors with the lowest buy-it-now prices. Those batteries cost about $3 each, delivered halfway around the planet, and initially had usable capacities under 50% of the genuine Canon battery. Two of those batteries failed after a few months, so the red and green traces in **Figure 5** represent the survivors. Both of those batteries now deliver 67% of the genuine Canon battery's capacity at the 3.6-V endpoint.

However, a 50% early failure rate doubles the effective price and provides far more aggravation than seems justified for what should be a simple retail transaction. I autopsied the dead batteries and found their internal build quality to be about as bad as you'd expect, so I strongly recommend avoiding bottom-dollar batteries from unknown suppliers.

An earlier experience with low-end batteries ended somewhat differently, however.

LITHIUM-ION BATTERY REBUILDS

At the turn of the millennium, I bought a Sony DSC-F505V camera that used Sony NP-FS11 batteries with an embedded chip to supply a unique ID to the camera. That camera turned out to be extremely hard on its batteries, perhaps because they consisted of a pair of lithium-ion cells in parallel, and, after Sony stopped selling those batteries, I turned to generic replacements.

Figure 6 shows the results for the first two charge-discharge cycles of three generic NP-FS11 batteries from a single bottom-dollar

circuitcellar.com/ccmaterials

RESOURCES

AA Portable Power Corp., "Li-ion 14430 Cylindrical Rechargeable Cell," www.batteryspace.com.

Blue Nook Corp., www.bluenook.com

E. Nisley, "Battery Power: Feeding the Z3801A," *Circuit Cellar* 155, 2003.

———, "Battery Capacity: Discharge," *Circuit Cellar* 199, 2007.

———, "Battery Capacity: Charge," *Circuit Cellar* 201, 2007.

———, "Real-World NiMH Charging," *Circuit Cellar* 221, 2008.

———, "Parallel NiMH Cell Measurement," *Circuit Cellar* 247, 2011.

———, Softsolder.com, "NB-5L Holder: Coil Springs," http://softsolder.com.

———, Softsolder.com, "Canon NB-6LH Battery Test Fixture," http://softsolder.com.

Wikipedia, "Peukert's law."

SOURCE

CBA-II Computerized battery analyzer
West Mountain Radio | www.westmountainradio.com

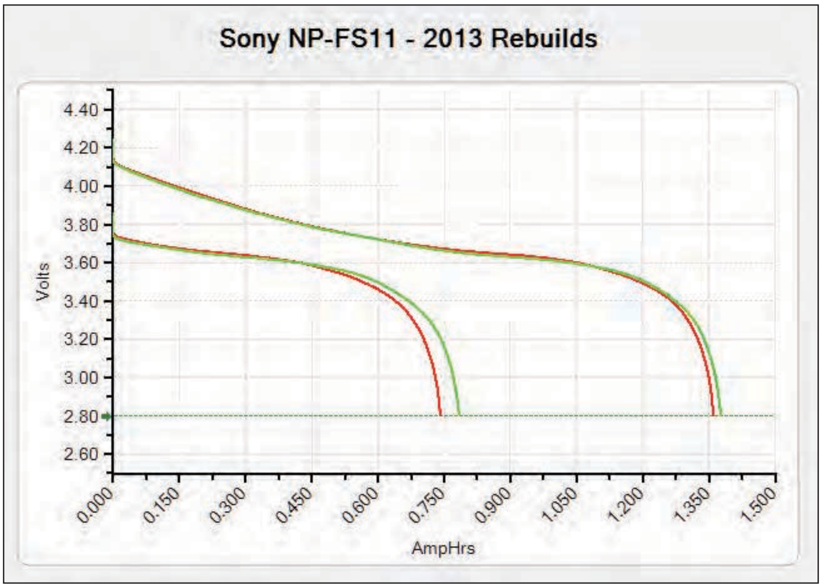


FIGURE 7
The lower pair of curves shows the charge remaining in a pair of then-new 14430 lithium-ion cells after two years in storage, the upper curves show those cells assembled into an NP-FS11 battery after charging. Although these cells did discharge in storage, they maintained about 40% of their energy.

eBay supplier; they sent these as replacements when I complained about the poor performance of the first set. Of six no-name batteries, only one had acceptable performance, two were marginal, and three were dead on arrival. The voltage of the best battery declined so rapidly during discharge that it was essentially useless.

I was willing to try bottom-dollar batteries, because more expensive generic batteries had also failed in short order. After that experience, I took a different tack by buying replacement cylindrical 14430 Li-ion cells from a known-good supplier, delicately sawing the failed battery cases open, and rebuilding the batteries with new cells. That worked quite well, with the rebuilt batteries delivering about 1 A·h at 3.6 V and surviving for the better part of two years: about as good as the original Sony-branded NP-FS11 batteries.

I bought a dozen 650 mA·h 14430 cells and used six to rebuild three batteries, because that's as many as I can keep track of: one battery in the camera, a spare in the carrying case, and one in the charger. I charged the remaining six loose cells, insulated their positive terminals, and put them in my box of battery parts and supplies.

Fast forward two years, when the pairs of curves in **Figure 7** show the results of tests on two rebuilt batteries before and after their first charge. The lower curves show that the stored cells maintained just over 40% of their initial capacity at the 3.6-V level after two years in storage. The upper curves show that the recharged cells have essentially their original 1-A·h capacity at 2.8 V.

The final pair of cells remains in the box, specifically so I can repeat that test when the remaining battery wears out.

Although I cannot recommend rebuilding battery packs from bare cells, sometimes it's the only way to get dependable performance at a reasonable cost. Make sure you understand the risks of working with bare Li-ion cells before you start, then take precautions to avoid shorting the cells while reassembling the battery!

CONTACT RELEASE

The positive reviews you'll find on retail sites describe battery performance in glowing terms and the negative reviews seem to describe total duds, but I suspect both you and I prefer actual measurements under realistic conditions. Wouldn't it be nice if the vendors paid as much attention to the numbers as we do?

ABOUT THE AUTHOR

Ed Nisley is an EE and author in Poughkeepsie, NY. Contact him at ed.nisley@pobox.com with "Circuit Cellar" in the subject line to avoid spam filters.

COLUMNS

BOARDS, BOOKS, DVDs AND MORE AT WWW.ELEKTOR.COM/STORE

Elektor.STORE

The world of electronics at your fingertips!

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

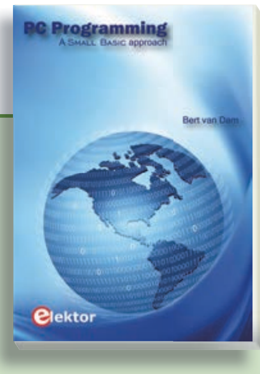
Books

Explore the RPi in 45 Electronics Projects

Raspberry Pi

At around 30 dollars the Raspberry Pi is a complete computer system that allows all sorts of electronics parts and extensions to be connected. This book addresses one of the strongest aspects of the Raspberry Pi: the ability to combine hands-on electronics and programming. No fewer than 45 exciting and compelling projects are discussed and elaborated in detail. From a flashing lights to driving an electromotor; from processing and generating analog signals to a lux meter and a temperature control. We also move to more complex projects like a motor speed controller, a webserver with CGI, client-server applications and Xwindows programs. Each project has details of the way it got designed that way. The process of reading, building and programming not only provides insight into the Raspberry Pi, Python, and the electronic parts used, but also enables you to modify or extend the projects any way you like.

288 pages • ISBN 978-1-907920-27-1 • $56



A Small Basic approach

PC Programming

In this book we take a practical approach to programming with Small Basic. You will have an application up and running in a matter of minutes. You will understand exactly how it works and be able to write text programs, graphical user interfaces, and advanced drivers. It is so simple; you don't even need to be an adult!

194 pages • ISBN 978-1-1-907920-26-4 • $48

Circuits & Projects Guide

Arduino

The Arduino user is supported by an array of software libraries. In many cases, detailed descriptions are missing, and poorly described projects tend to confuse rather than elucidate. This book represents a different approach. All projects are presented in a systematical manner, guiding into various theme areas. In the coverage of must-know theory great attention is given to practical directions users can absorb, including essential programming techniques like A/D conversion, timers and interrupts— all contained in the hands-on projects. In this way readers of the book create running lights, a wakeup light, fully functional voltmeters, precision digital thermometers, clocks of many varieties, reaction speed meters, or mouse controlled robotic arms.

260 pages • ISBN 978-1-907920-25-7 • $56

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

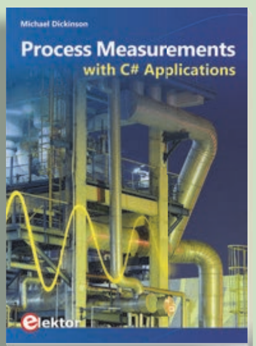
Books

Very Original Applications!

Mastering Microcontrollers
Helped by Arduino

This book not only introduces you to the World of Arduino, but also demonstrates the tools & skills necessary for continuing your adventure into microcontroller programming independently. In this book theory is put into practice on an Arduino (uno) board using the Arduino programming environment. Having completed this fun and playful course, you will be able to program any microcontroller, tackling and mastering I/O, memory, interrupts, communication, A/D converter, and more.

348 pages • ISBN 978-1-907920-23-3 • $56



A Highly Practical Guide

Process Measurements with C# Applications

This book introduces PC based measurement systems and software tools for those needing to understand the underlying principles or apply such techniques. C# was chosen to give readers immediate, practical experience with a prominent programming language. C-Sharp has a wide support base and is a popular choice for engineering solutions. The basics of measurement and data capture systems are presented, and examples of software post-processing

144 pages • ISBN 978-1-907920-24-0 • $39

All Articles In Elektor Volume 2013

DVD Elektor 2013

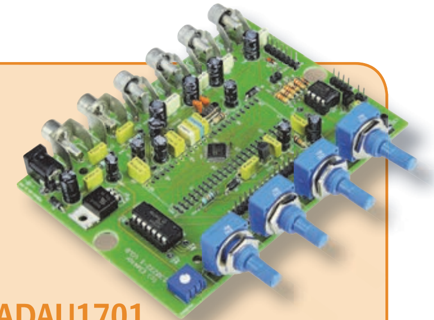
The year volume DVD/CD-ROMs are among the most popular items in Elektor's product range. This DVD-ROM contains all editorial articles published in Volume 2013 of the English, American, Spanish, Dutch, French and German editions of Elektor. Using the supplied Adobe Reader program, articles are presented in the same layout as originally found in the magazine. An extensive search machine is available to locate keywords in any article. With this DVD you can also produce hard copy of PCB layouts at printer resolution, adapt PCB layouts using your favorite graphics program, zoom in / out on selected PCB areas and export circuit diagrams and illustrations to other programs.

ISBN 978-90-5381-277-8 • $39

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 US
111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA
Phone: 860.289.0800
Fax: 860.461.0450
E-mail: order@elektor.com

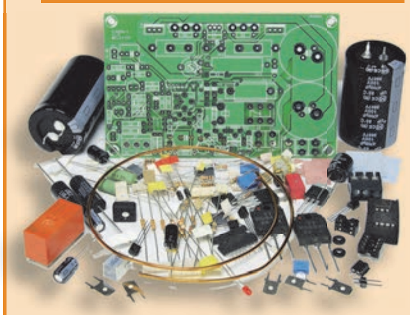


ADAU1701 Universal Audio DSP Board

Always wanted to start out with DSPs, but afraid of the SMD? Elektor is making Digital Signal Processing more fun and accessible with the release of a Universal Audio DSP Board. Based on the Analog Devices ADAU1701 DSP chip and drag-and-drop software, this board will ease your way to becoming a sound craftsman, guaranteed maths-free.

Semi-kit: all TH components, PCB with DSP preassembled

Art.# 130232-71 • $102



Q-Watt Audio Power Amplifier

Elektor has a long history with audio power amplifiers. We are proud to present yet another fully analog circuit developed entirely in house. Despite the simple design of this audio power amp with just one pair of transistors in the output stage, Q-Watt can deliver over 200 quality (Q!) watts into 4 ohms with exceptionally low distortion thanks to the use of a special audio driver IC. Due to popular demand, this project is now supported by a semi-kit containing everything you need to build a stereo amplifier.

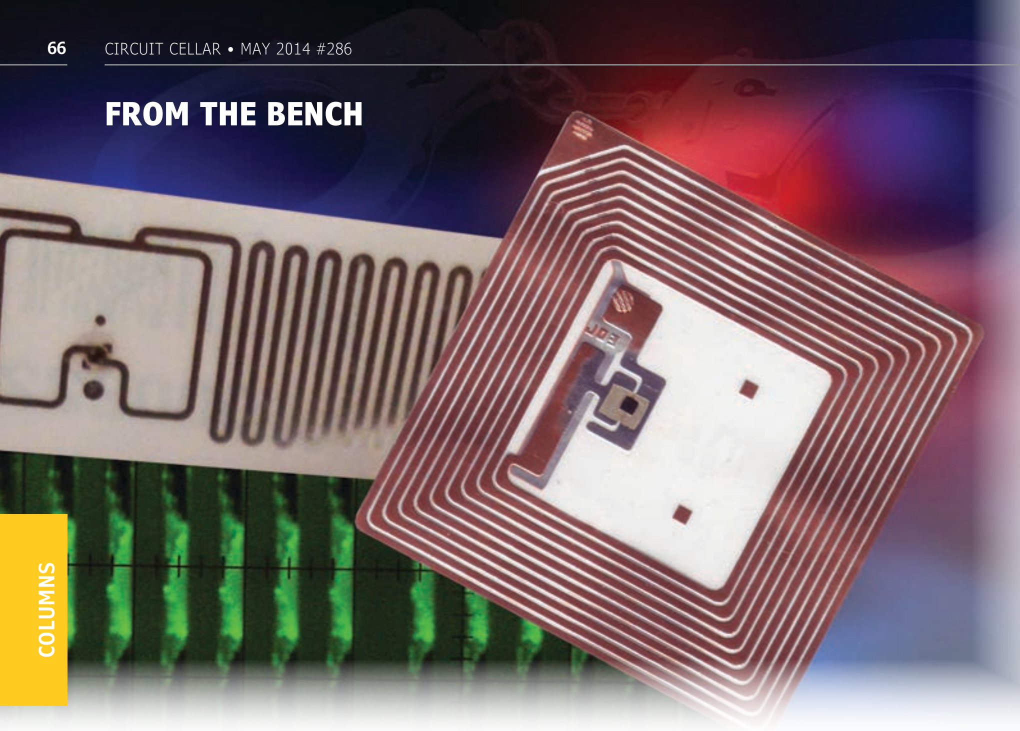
2 PCBs and all electronic parts

Art.#110656-72 • $236

Kits & Modules

FROM THE BENCH

COLUMNS



Passive RFID Tagging (Part 1)

Read-Only Tags

By Jeff Bachiochi (US)

RFID uses wireless RF electromagnetic fields to transfer data that identifies and tracks transponders. This article discusses commonly used read-only tags.

RFID technology is changing the way we transfer data and track objects. This article series will cover the history of the technology and delve into the need-to-know technical details about how it works.

In this article, I will focus on passive 125-kHz tags. I'll use the term "tag" for any shell with transponder circuitry. Tags come in many forms (e.g., fobs, cards, and labels). Most are read-only devices, which supply a 5-byte serial number identification. Read/write tags are becoming hot and can provide multiple 4-byte pages of user-supplied data, optionally protected by 4-byte passwords. Let's begin with read-only tags.

RFID (WWII TO PRESENT)

In a 2005 *RFID Journal* article, "The History of RFID Technology," Bob Violino wrote about how the Luftwaffe German air force identified itself. Upon returning from a mission, a simple barrel roll altered a radar reflection, which was recognized as friendly by radar supervisors avoiding anti-aircraft fire. This may have been the first use of passive RFID.

The British took this a step further by installing transponders in their aircraft, which were capable of transmitting an identifying response to radar receptions. This was active RFID.

Naturally, this technology found its way into the commercial market for electronic article surveillance (EAS), which prevents shoplifting or the unlawful removal of property. EAS tags are passive and normally carry only a single bit of data. While activated, the tags can be sensed by equipment installed at exit points. Some EAS tags are destroyed in the deactivation process at the sales counter (e.g., clothing tags) while those used in libraries (e.g., book tags) can be reactivated when the borrowed item has been returned.

RFID systems consist of a base device that sends out a signal and looks for some response to that signal by a transponder (tag) within its operating range. In the mid-1970s, the first patents were issued for active and passive systems. Passive systems use transponder circuitry that harvests energy from the transmitted signal to power its response. Active systems use transponder circuitry that carries its own power source for a response.

The first passive systems were used by the US Department of Agriculture (USDA) to identify cows. Since most cows look the same, it was vital that each one was correctly identified so each cow could receive the correct dosage of hormones and/or medicine. Therefore, the cows were implanted with a small glass passive RFID tag for tracking purposes.

What began with systems focused on 125 kHz (low frequency, LF) has expanded to cover the spectrum including 13.56 MHz (high frequency, HF), 433 MHz (ultra-low frequency, UHF), and higher frequencies. Today RFID is used for mobile payments, asset management, access, public transportation, passports, animal identification, and in the health-care system.

The Swiss company EM Microelectronic-Marin (a.k.a. EM-Marin, EM Micro, or EM) began designing miniaturized, ultra-low-power ICs for watches (e.g., Swatch). Now EM produces ICs for various applications (e.g., communication, computer peripherals, logistics, security, access control, industrial, animal identification, consumer markets, and the automotive sectors). Its EM4*xxx* family of passive and active RFID circuits and reader ICs has identified the company as a major player in the worldwide LF, HF, and UHF RFID market.

READ-ONLY TAGS

Read-only passive RFID tags must obtain their own power from their surroundings. Tags or transponders use a coil and a capacitor (e.g., tank circuit) to capture a 125-kHz carrier transmitted by a nearby tag reader. This tank is used by three separate circuits within the tag.

The first circuit harnesses energy from the signal received, AC/DC conversion, and regulation to (temporarily) power the rest of the circuitry. A second circuit converts the carrier to a clock that will provide a means of synchronization for the access and data delivery. The third circuit modulates a load on the tank circuit according to the data's Logic state. This modulation not only affects the coil's magnetic field on the transponder, but is also reflected back to the reader. (The reader will be looking for these field changes and attempt to interpret them as data.)

Animal identification tag data comes from a 128-bit memory array that is laser programmed at the time of manufacture. This provides more than 3.4 × 10^{38} unique codes (incidentally, the same as IPv6). The data structure for these 128 bits is sent least-significant bit (LSB) first:

- 11 bits as a header (10000000000)
- 72 bits as eight identification bytes, each with a trailing bit (1)
- 18 bits as two CRC bytes, each with a trailing bit (1)
- 27 bits as three extended bytes, each with a trailing bit (1)
- 128 bits total

Because most of the data for this use is predefined, only 38 bits are actually set aside for unique animal codes. The non-animal version (for whatever reason) has 5 bytes of unique code. The trillion or so unique codes (in these 40 bits) are fit into a 64-bit structure. This data structure is the basis of most read-only tags:

- 9 bits as a header (111111111)
- 50 bits as nibbles, each with a trailing row parity bit
- 5 bits as a column parity nibble with a trailing bit (0)
- 64 bits total

Each nibble's even-parity bit provides a level of confidence, while the final parity nibble is a vertical even-parity indicator for each of the 10 data nibbles' bit positions. After all the checking is done, you end up with 10 nibbles or a 5-byte hexadecimal code.

Beyond this depth of memory, two other format choices will determine how a transponder reacts to a carrier receipt: data rate and data coding. The data bit rate is a function of the carrier frequency divider and is preprogrammed into each transponder. The choices are divide by 64, divide by 32, or divide by 16. Divide by 32 is the standard of choice. Data coding is how each data bit (0/1) will modulate the carrier.

Most are familiar with non-return-to-zero (NRZ) coding. In this type of scenario, a data bit = 0 would not affect the carrier (let's call this carrier 100%) and a data bit = 1 would affect the carrier (let's call this carrier 80%). Note: The difference between the two levels (100% and 80%) will determine how well the reader can interpret the tag's data. **Figure 1** shows two data coding techniques available for the tags to use. Manchester encoding seems to be the standard of choice.

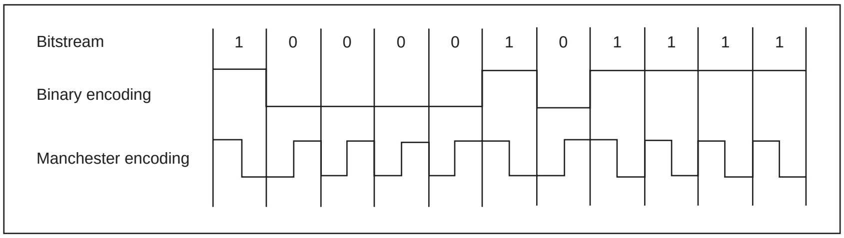


FIGURE 1
Many RFID tags use Manchester encoding. While you can use any encoding technique, one advantage of Manchester encoding is that it has at least one transition for every bit. Therefore, it has no DC component.

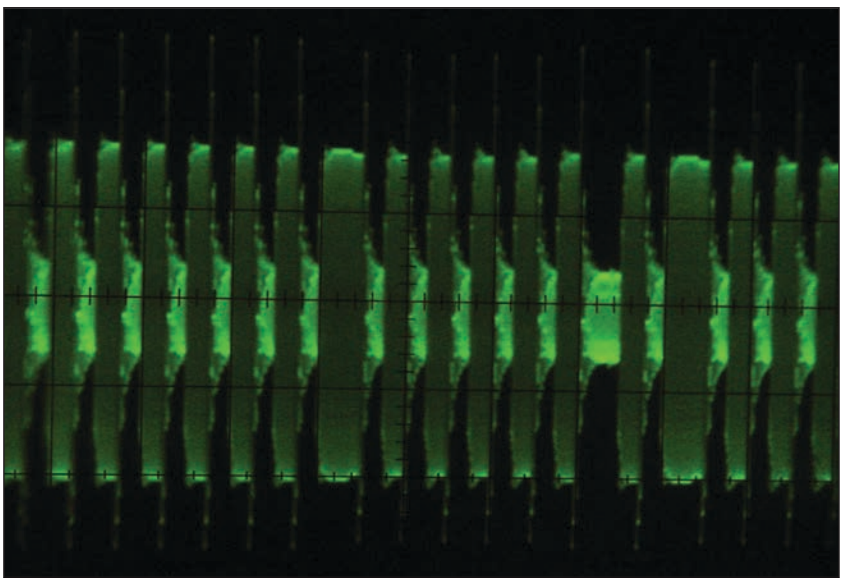


PHOTO 1
This oscilloscope trace shows how data from a tag in field (TIF) affects the carrier by loading down the tag's resonant circuit. The carrier's modulation is how the reader will detect a tag's data.

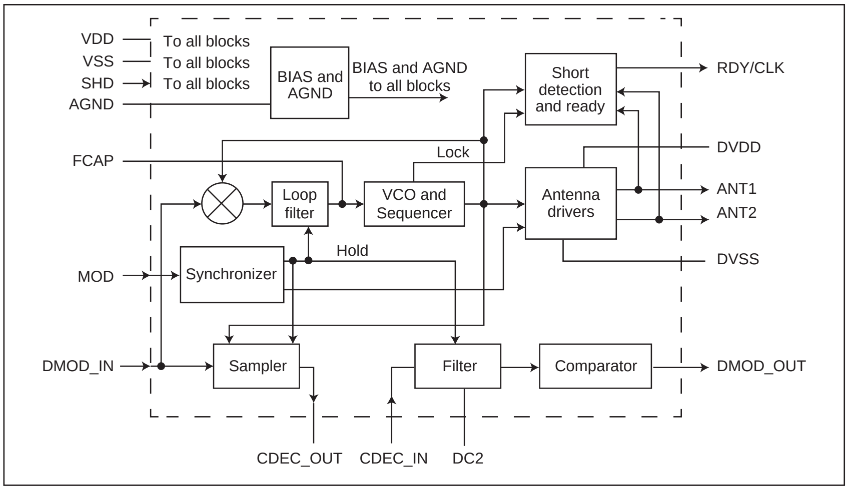


FIGURE 2
A block diagram of the EM Microelectronic-Marin EM4095 125-kHz RFID front end is shown. This device will create its own carrier and detect modulation (data) on it.

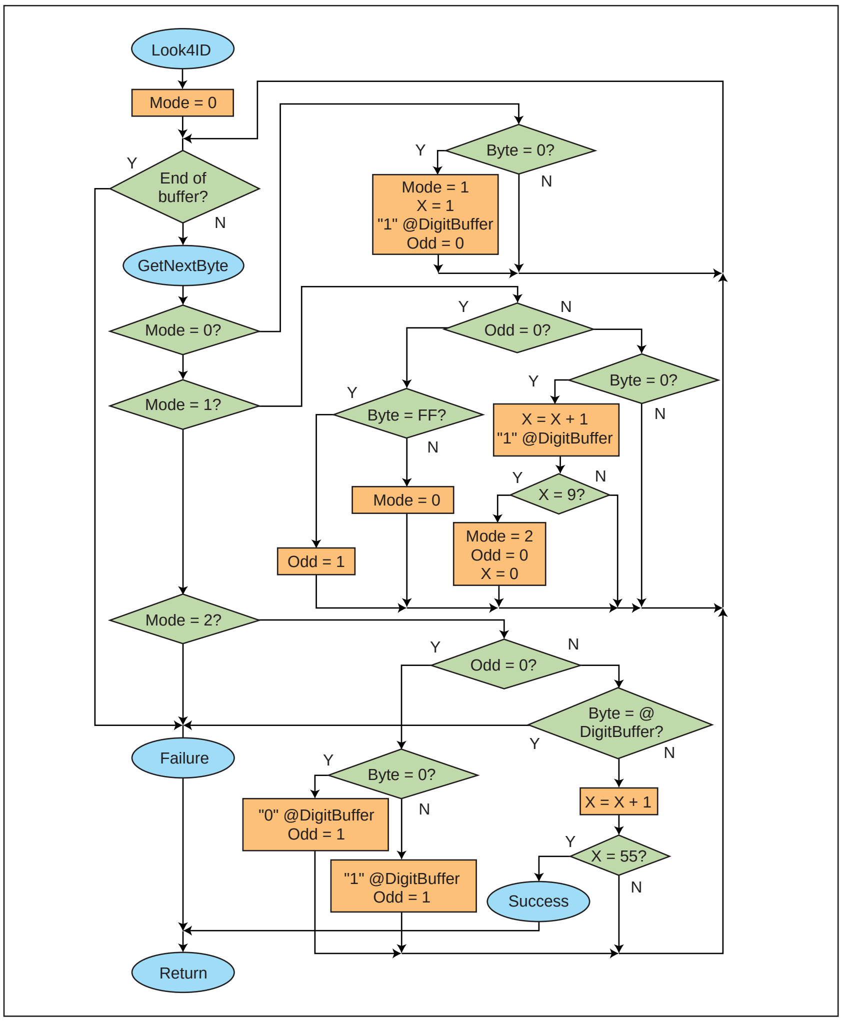


FIGURE 3
This flowchart shows the algorithm used to decode the device ID from the 64-bit data packet. Each dual-symbol data bit is sent at a 32-clocks-per-symbol rate. That's about a 2-kbps transmission rate.

TAG READERS

A tag is happy to sit quietly and do nothing until it is called into action by a reader. Passive tags need energy, which must come from the reader. So, I guess a reader's most important attribute is to create a 125-kHz carrier on its tank circuit (i.e., antenna). The tank (i.e., LC circuit) is made from a capacitor in parallel with an inductor. These are both frequency-dependent devices, but they react in opposite directions.

As a capacitor's reactance goes up, an inductor's reactance goes down. For any given component values, there is one frequency where the reactance of both devices is equal: the resonant frequency.

Energy momentarily added across the tank circuit charges the capacitor to one polarity. The capacitor discharges through the coil, building up a magnetic field that reaches its maximum at the moment the capacitor voltage reaches zero. The magnetic field then collapses, forcing the current to continue until the capacitor is charged with the opposite polarity and the current has decayed to zero.

At this point, the capacitor begins to discharge in the other direction through the coil, building up a magnetic field of the opposite polarity. This magnetic field then collapses, forcing the current to continue until the capacitor is once again charged with its original polarity. And so it goes, over and over, back and forth, until circuit resistance causes the current to be turned into wasted heat.

You can think of a tank circuit as a gong that has been struck. It will continue to vibrate at its resonant frequency until all its energy has been dissipated. If you were to touch the gong while it was vibrating, you would remove some energy and shorten its vibration period. Most inductors have a resistive component. Their physical makeup (i.e., the wire used to make the inductor's coils) has some measurable resistance. Resistance in series with the L and C will act like touching a gong and will tend to reduce the tank circuit's resonant period.

The function of resistance to reactance describes the circuit's damping factor (i.e., Q). A higher Q (i.e., smaller resistance) circuit will tend to "ring" longer (i.e., be able to hold onto its energy).

Periodically adding energy to the circuit at the resonant frequency will enable energy in the tank to accumulate. The tank oscillations grow in voltage and current. The magnetic field that builds and collapses with each cycle radiates further and further from the physical inductor. Thus the inductor is acting as an antenna and any external circuit within

	Inductance @ 125 kHz	Wire Resistance	Wire Size	Wire Gauge	Wire Length	Coil Diameter	Q
Coil A	2.8 mH	53.8 Ω	0.09 mm	35	161′	~20 mm	41
Coil B	2.8 mH	63 Ω	0.09 mm	35	189′	~48 mm	35
Coil C	2.8 mH	42 Ω	0.127 mm	36	101′	~101 mm	53
Coil D	2.7 mH	19 Ω	0.2 mm	32	114′	~162 mm	112
Coil E	2.7 mH	8.7 Ω	0.4 mm	28	131′	~255 mm	244

TABLE 1
Here are some comparisons for a few coils I purchased from Priority 1 Design. This gives you an idea of how they vary based on configuration. You could also use PCB coils, but calculating the reactance can be tricky.

range can capture some of that energy, like a tag in field (TIF). In case it isn't obvious, the tag's tank circuit begins resonating, since it is tuned to the same resonant frequency and its coil is within the changing magnetic field of the reader's tank.

The reader's functions can be boiled down to just two: providing an energy source for the tags (radiated emissions) and monitoring its tank circuit for modulation (tag data).

I just discussed the first function. The second function is where the work gets done. After all, the whole idea here is to be able to gather some data from any TIF.

Earlier I discussed how tags place a load on their tank circuits to indicate data = 1. When the tag's tank is loaded (to indicate a data = 1 state) this extra load is reflected back to the reader's tank circuit as a decrease in energy. The reader's work is to recognize this energy reduction to decode the datastream. **Photo 1** shows data from a TIF affecting the carrier being transmitted by the reader circuitry.

This would be more difficult if the reader didn't know how the tag was encoding the data. If you receive the data as transmitted first, you could then spend some computing power to analyze it by trying different algorithms until you figure out which is being used. In this case, the tags are using a divide (the carrier) by 32 for their bit timing with Manchester encoding on the data.

Manchester encoded data uses symbol pairs transmitted as a "01" symbol pair or a "10" symbol pair to represent one data bit. That's two bit times to transmit one data bit. If the carrier is supplied or accessible, the bit time is already known. However, using Manchester encoding enables the bit time to be pulled from the data.

The tag response's header will begin with nine bits of data = 1, or nine symbol pairs, 101010101010101010. According to the data format, this pattern will not be repeated anywhere within the remaining data packet. So, this will be easy to detect. There won't be any TIFs without data. In essence, it will look like all 1s (illegal symbol pair 11). The first 0 you see should be 1 bit time long. This should be the second symbol in the first symbol pair "10" of the header data.

Using an EM Microelectronic-Marin EM4095 RFID base station simplifies the design by providing an analog front end to handle driving the tank circuitry, an AM demodulator to recognize data being sent by a tag, and an AM modulator, which can be used to send data to a tag. So far I haven't discussed this aspect, as the read-only tags do not require it. I'll provide more information about this later.

Figure 2 shows the EM4095's block diagram. It may not be obvious, but a PLL in the carrier frequency circuit enables the carrier frequency to be shifted slightly to best match the resonant frequency of the actual external components used in the tank. This may be off from the calculated frequency due to component tolerances.

The EM4095 communicates to a Microchip Technology PIC18F25K22 microcontroller via two input and two output interfacing signals. The two digital inputs to this analog front-end device are SHD (shutdown) and MOD (modulation).

A logic high on the SHD input will place the device in a low-power mode (disabling the carrier and the demodulator). A logic high on the MOD input will remove the drivers from the tank circuitry, interrupting the flow of energy to it and providing an amplitude modulation (data = 1) state. When used in the read-only mode, the MOD input is tied low. The EM4095's two digital outputs are DMOD (demodulation) and RDY/CLK (ready/clock).

The DEMOD output will normally be high until it senses a drop in carrier energy (i.e., the tag applying a load on its coil indicating a data = 1 state). The DEMOD output is disabled for 41 carrier cycles to enable the tank to build its energy and the demodulator to stabilize its references. If the external circuitry's operation falls outside the normal

PHOTO 2
Can you recognize the object in this photo? I first saw it used at my local bank. Any thoughts to where I'm going with this project in Part 2?

characteristics, the RDY/CLK output will remain low. Once the SHD input goes low and the operational parameters have stabilized, the RDY/CLK output will provide the reference carrier back to the microcontroller.

My first inkling was to measure the bit time of the first 0 I saw and use half of it to offset any following samples of the remaining data. Instead, I chose to take advantage of the fact that the carrier divisor (for the demodulated bit times) is divisible by 8, in this case 32.

I used the EM4095 as a SPI master with the RDY/CLK output as a serial clock out and the DEMOD output as serial data output (SDO) providing data to the microcontroller (SPI slave device). In this case, 4 bytes will equal a 1-bit time and all will stay perfectly synchronized.

In reality, I only needed to take one sample in the middle of each bit time (e.g., after the 16th of 32 bits). However, since I was using this project as a learning experience, I wanted to see exactly what was happening, so I filled a buffer with some raw data and then mashed it around after the fact.

First, I just looked for the reception of 64 bits of data with each bit being 64 clocks long: 32 clocks for symbol 1 and 32 clocks for symbol 2. Remember, there should only be two legal symbol pairs 01 or 10 for each bit. These are received as eight SPI bytes, four for symbol 1 and four for symbol 2.

The 32 data bits for each symbol received are analyzed and a 00 or an FF is stored for each symbol depending on if the majority of bits are 0s or 1s (they should be all one or the other). Without a TIF, all data received will be 1s. Since 11 is an illegal symbol pair, there is no sense in saving anything until the first legal symbol pair is received.

Figure 3 shows a flowchart of the algorithm I used to pull apart the data. There are three things to look for: the first 0 (and it will be the second symbol of symbol pair 10), the rest of the header (a total of nine consecutive 1s, which are 10 symbol pairs), and finally 55 additional symbol pairs (data and parity bits).

Except for looking for the header, no check is done on the symbol pairs. Once all the data has been received and stored, symbol checking can be done while preparing a matrix (second table) of 64 ASC bytes. This new table consists of an ASC representation (either 0x30 or 0x31) of each symbol pair received. Using an ASCII representation makes it easy to display the 64-bit matrix (if required). From this table, a third table is made consisting of five HEX values representing the actual 5-byte hexadecimal tag ID number.

ANTENNA

I previously mentioned that the EM4095 can adjust the carrier frequency to best fit the external components used. This is true for about ±20% of the 125-kHz design specification. For this to work, the right components must be selected to achieve a natural resonance as close as possible to 125 kHz. The important component here is the inductor or coil used as the antenna and part of the tank circuit. I purchased several preformed coils to use with the EM4095. These are interesting as they all have approximately the same inductive reactance. Referring to

circuitcellar.com/ccmaterials

RESOURCES

Priority 1 Design, www.priority 1design.com.au.

B. Violino, "The History of RFID Technology," *RFID Journal*, 2005.

Wikipedia, "Tuned circuit animation3.gif."

SOURCES

EM4095 RFID Base station
EM Microelectronic-Marin | www.emmicroelectronic.com

PIC18F25K22 Microcontroller
Microchip Technology, Inc. | www.microchip.com

SchmartBoard
SchmartBoard | www.schmartboard.com

Table 1, I used the following formula:

$$Q = 2 \times \pi \times \frac{L}{R}$$

to calculate the Q for each coil. Note Coil A and Coil B both have the same reactance, but Coil B has a lower Q. At twice the diameter of Coil A, Coil B required more wire to achieve the same reactance, which lowered the Q. So how does the Q play into the design?

To answer this, I'll step back and start with the tank (coil/capacitor) values. Using Coil B, I have a selected inductor (2.8 mH) and a selected frequency (125 kHz), so I can use this formula to find the necessary capacitor value:

$$C_{RES} = \frac{1}{(2\pi f)^2 L}$$

$$C_{RES} = \frac{1}{6.2 \times 10^{11} \times 0.0028}$$

$$C_{RES} = 5.8 \times 10^{-10}\ \text{F} = 580\ \text{pF}$$

This capacitor in parallel with the selected inductor will resonate at the required frequency. The peak tank current is based on the drive voltage and the total resistance across the drive:

$$I_{ANT(PEAK)} = \frac{4}{\pi} \times \frac{V_{DD} - V_{ss}}{R_{ANT} + R_{SER} + 2 \times R_{AD}}$$

Using this formula, the current is:

$$I_{ant(PEAK)} = \frac{4}{\pi} \times \frac{5\ \text{V}}{63\ \Omega + 0\ \Omega + 2 \times 3\ \Omega} = 92\ \text{mA}$$

and the peak voltage developed across the tank will be:

$$V_{PK} = \frac{I_{PK}}{2\pi F C_{RES}}$$

$$V_{PK} = \frac{92\ \text{mA}}{2\pi \times 125\ \text{kHz} \times 580\ \text{pF}}$$

$$V_{PK} = 202\ \text{V}$$

Wow! Potentially, the circuit could produce a pretty high peak voltage. From **Table 1** you can see that an inductor's Q goes up as its resistance goes down (based on a fixed inductive reactance). Low Q tank circuits are well damped, that is the larger resistance causes the circuit to lose energy quickly (as wasted heat). As the Q goes up (resistance goes down), they hold onto any energy and continue to oscillate longer over time. Since no tank has zero resistance, energy must be added to keep it oscillating.

ABOUT THE AUTHOR

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.

Adding energy at the resonate frequency enables the tank circuit's oscillations to increase since it is now gaining energy faster than it can lose it. This is just what is needed to radiate the RF carrier. However, the large voltage and current can create issues by exceeding some specifications.

First the tank circuit drive must be current limited. The EM4095's specifications suggest a maximum of 250 mA. In this case, my drive is well below that limit. If the peak driver current's calculation was too high, some R could be added to the circuit, which would reduce the I_{PK}. Note that adding R will also alter the V_{PK}, which is the next area of interest.

The next task at hand is to monitor the carrier for any modulation placed on it by a TIF. The input to the demodulator has a maximum voltage specification of less than 4.5 V (i.e., V_{DD} – 0.5). With a potential of 430 V_{PK-PK} (2 × V_{PK}) on the tank circuit, the 4.5-V limit is far exceeded and a voltage divider must be used. The division factor is 96 (i.e., 430/4.5). With a recommended Cdv2 of 0.001 µf, Cdv1 (across the input) would be approximately 10 pF. You should take this series combination of Cdv1 + Cdv2 (less than 10 pF) in parallel with C_{RES} (580 p) to get the actual value of C_{RES} required for a 125-kHz resonant frequency.

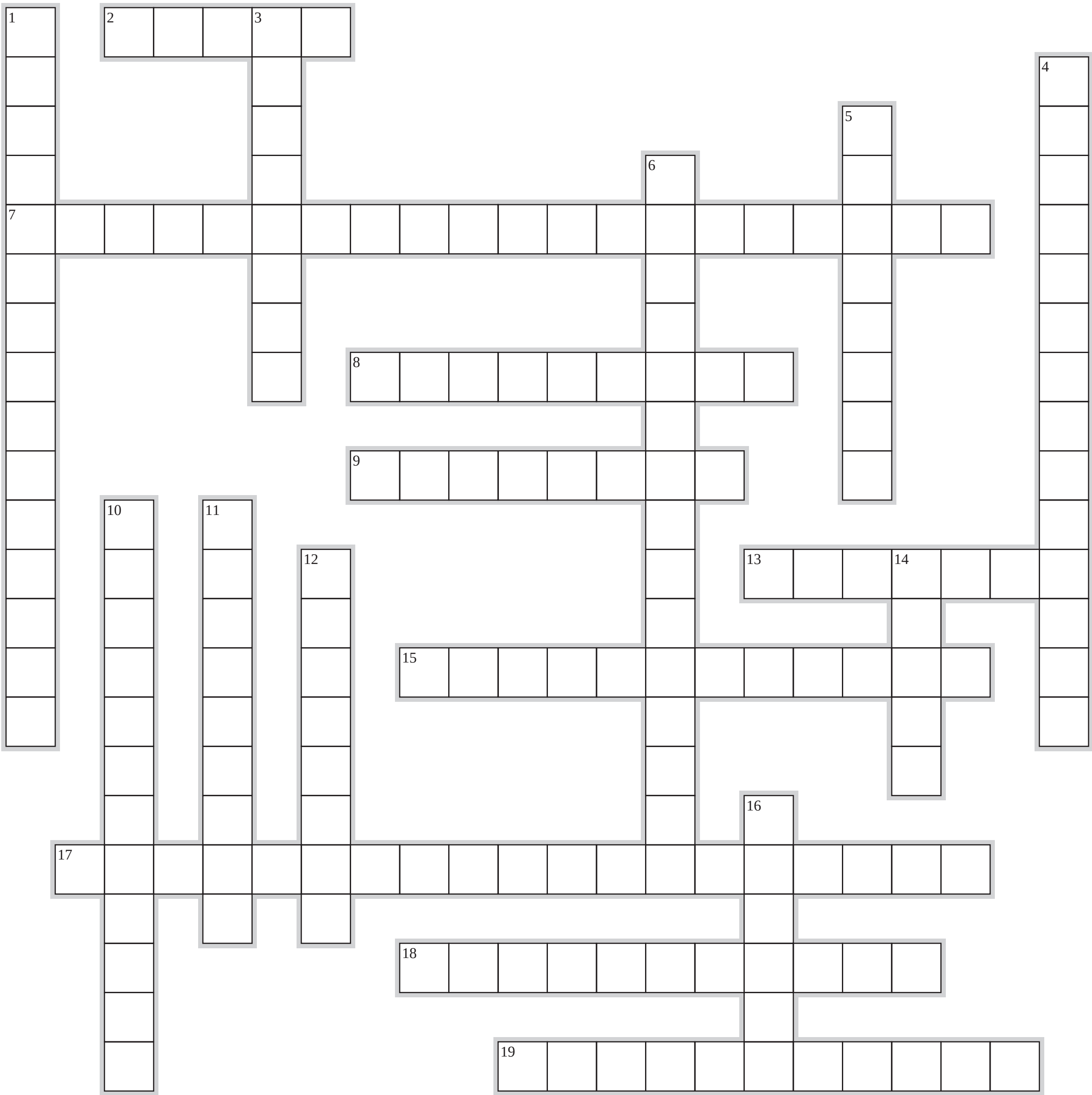
SMT CIRCUITRY

With the calculations for key external components done, a front-end module can be fabricated. The EM4095 is only available in a surface-mount technology (SMT) package, so I used a SchmartBoard with 1.27-mm pin spacing. This is an SMT-to-DIP PCB. It will enable the chip to be easily mounted on a prototyping board and only have to handle a single SMT part. Therefore, a special PCB doesn't have to be fabricated.

In my next article, I'll examine some actual data transmitted via this carrier-suppressing method. Then I'll finish off this project based on reading RFID tags. In the meantime, look at **Photo 2**. This may give you a hint about what this project aims to accomplish.

CROSSWORD MAY 2014

The answers will be available at circuitcellar.com/crossword.



ACROSS

2. This networking protocol enables you to write to an embedded file system from a Windows PC
7. Used for data transfer [two words]
8. These types of microphones were commonly called "condensers" until about 1970
9. A receiver with direct amplification
13. Contains several modeling features and an integrated game engine
15. Used to improve wired or wireless communication link performance [two words]
17. Occurs when certain types of light change their frequency and route [two words]
18. Used to coordinate circuits' actions [two words]
19. Designed to encourage scholastic computer science lessons [two words]

DOWN

1. Typically 1s are a positive voltage and 0s are a negative voltage [four words]
3. Provides compact storage for computing and digital communications [two words]
4. Coordinates the rising and falling edges of an [18 Across] to transfer data [three words]
5. Absent of sound
6. Oldest requests receive priority [four words]
10. The "I" in SQUID
11. Occurs when accidental coupling causes unwanted signals
12. An electronic oscillator component
14. Receiver that intercepts or demodulates IR radiation
16. A good breed of analyzer for I^2C and SPI designs

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

TEST YOUR EQ

Contributed by David Tweed

PROBLEM 1

A divider is a logic module that takes two binary numbers and produces their numerical quotient (and optionally, the remainder). The basic structure is a series of subtractions and multiplexers, where the multiplexer uses the result of the subtraction to select the value that gets passed to the next step. The quotient is formed from the bits used to control the multiplexers and the remainder is the result of the last subtraction.

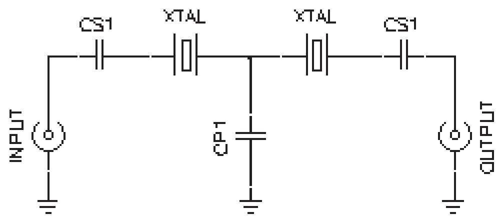
If it is implemented purely combinatorially, then the critical path through all of this logic is quite long (even with carry-lookahead in the subtractors) and the clock cycle must be very slow. What could be done to shorten the clock period without losing the ability to get a result on every clock?

PROBLEM 2

On the other hand, what could be done to reduce the amount of logic required for the divider, giving up the ability to have a result on every clock?

PROBLEM 3

An engineer wanted to build an 8-MHz filter with a very narrow bandwidth, so he used a crystal lattice filter like what is shown in **Figure 1**:



However, when he built and tested his filter, he discovered that while it worked fine around 8 MHz, the attenuation at very high frequencies (e.g., more than 80 MHz) was very much reduced. What caused this?

PROBLEM 4

Suppose you know that a nominal 10.000-MHz crystal has a 9.996490-MHz series-resonant frequency and a 10.017730-MHz parallel-resonant frequency. You also know that its equivalent series capacitance is 27.1 fF. How can you calculate the value of its parallel capacitance?

CC SHOP

1 CC 2013 CD

2013 was an exciting year for electronics engineers! The continued success of open-source solutions, IoT revolutions, and green energy consciousness have changed the face of embedded design indefinitely. In *Circuit Cellar*'s 2013 Archive CD, you can find all of these hot topics and gain insight into how experts, as well as your peers, are putting the newest technologies to the test. You'll have access to every article, schematic, and source code published from January to December 2013.

Item #: CD-018-CC2013

2 ANDROID APPS: PROGRAMMING STEP-BY-STEP

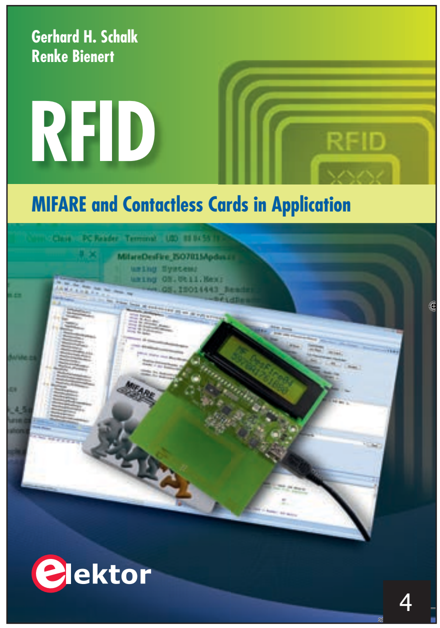
Many smartphones and tablet computers are powered by an Android OS. These portable devices' speed and computing power enable them to run applications that would have previously required a desktop PC or custom-designed hardware. *Android Apps* introduces you to the programming required to design apps for Android devices. Operating the Android system is explained step-by-step to show how personal applications can be easily programmed.

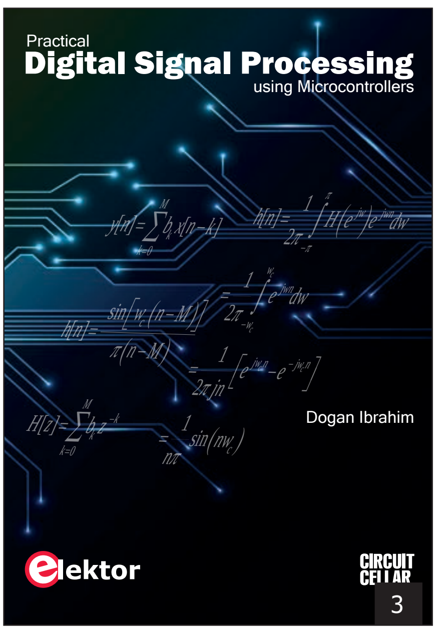
Author: Stefan Schwark
Item #: BK-ELNL-978-1-907920-15-8

4 RFID MIFARE AND CONTACTLESS CARDS IN APPLICATION

RFID technology is now being used in many areas in which barcodes, magnetic strips, and contact smart-cards were previously used. This book provides a practical and comprehensive introduction to MIFARE, which is the most widely used RFID technology. The initial chapters cover physical fundamentals, relevant standards, RFID antenna design, security considerations, and cryptography.

Author: Gerhard H. Schalk and Renke Bienert
Item #: BK-ELNL-978-1-907920-14-1



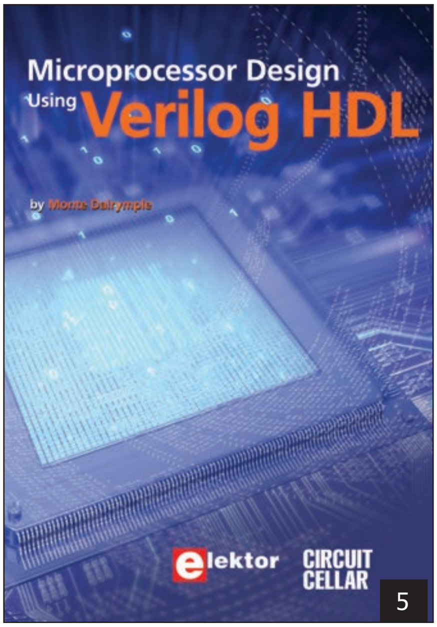


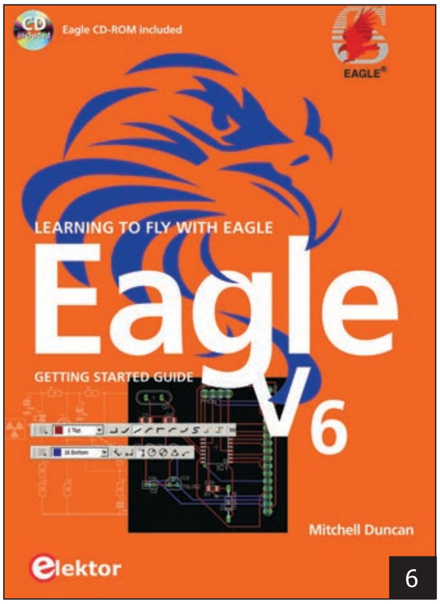
3 PRACTICAL DIGITAL SIGNAL PROCESSING USING MICROCONTROLLERS

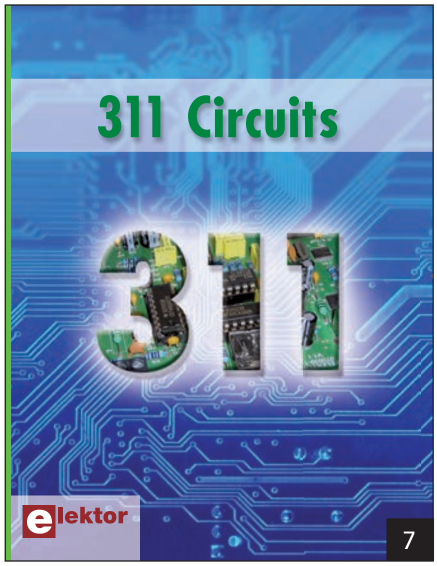
Digital signal processing (DSP) reflects the growing importance of discrete time signals and their use in everyday microcontroller-based systems. In this book, the author presents the basic theory of DSP with minimum mathematical treatment and teaches you how to design and implement DSP algorithms using popular PIC microcontrollers. You'll be up and running in no time!

Author: Dogan Ibrahim
Item #: BK-ELNL-978-1-907920-21-9

CC SHOP







5 MICROPROCESSOR DESIGN USING VERILOG HDL

After years of experience, Monte Dalrymple has compiled his knowledge of designing embedded architecture and microprocessors into one comprehensive guide for electronics engineers. *Microprocessor Design Using Verilog HDL* provides you with microarchitecture, writing in Verilog, Verilog HDL review, and coding style that enables you to depict, simulate, and synthesize an electronic design on your own.

Author: Monte Dalrymple
Item #: CC-BK-9780963013354

6 LEARNING TO FLY WITH EAGLE V6

EAGLE is a user-friendly, powerful, and affordable software package for efficient PCB design. It can be used on most main computing platforms including: Microsoft Windows, Linux, and Apple Mac OS X. *EAGLE V6* will benefit novices and professionals who are eager to learn about EAGLE or may be migrating from another CAD package. From schematic and layout editing tools to project completion, this book will help you achieve your PCB fabrication goals.

Author: Mitchell Duncan
Item #: BK-ELNL-978-1-907920-20-2

7 311 CIRCUITS

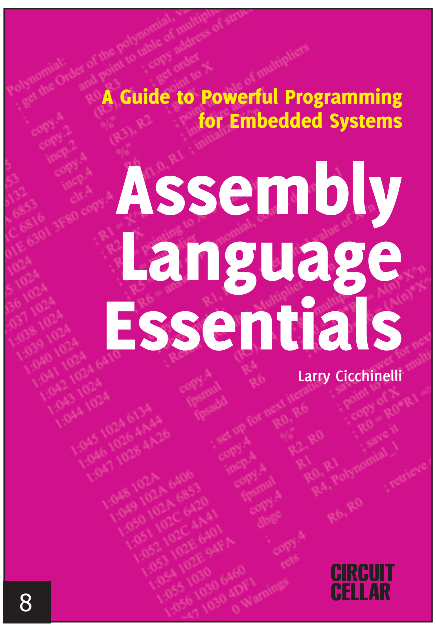
An immense source of inspiration for all electronics enthusiasts and professionals, *311 Circuits* deserves a place on your bookshelf! This book includes tips in all areas of electronics: audio and video, computers and microcontrollers, power supplies and batteries, test and measurement, and more. The 12th book in Elektor's celebrated 300 series, it presents complete solutions for numerous problems and distinct starting points for your DIY projec

Author: Elektor
Item #: BK-ELNL-978-1-907920-08-0

8 ASSEMBLY LANGUAGE ESSENTIALS

Looking to brush up your programming skills? Get back to to the basics with this matter-of-fact guide to Assembly language. Perfect for advancing students and academics, this book introduces you to a processor's most fundamental programming language. It includes essential terminology pertaining to higher-level programming, important algorithms that can be built into high-level language, a free downloadable Assembler program, and much more.

Author: Larry Cicchinelli
Item #: CC-BK-9780963013323



Further information and ordering

www.cc-webshop.com

CONTACT US:

Circuit Cellar, Inc.
111 Founders Plaza, Suite 300
East Hartford, CT 06108
USA
Phone: 860.289.0800
Fax: 860.461.0450
E-mail: custservice@circuitcellar.com

IDEA BOX

the directory of PRODUCTS & SERVICES

AD FORMAT:

Advertisers must furnish digital files that meet our specifications (circuitcellar.com/mediakit).

ALL TEXT AND OTHER ELEMENTS MUST FIT WITHIN A 2″ x 3″ FORMAT. E-mail adcopy@circuitcellar.com with your file.

For current rates, deadlines, and more information contact Peter Wostrel at 978.281.7708 or circuitcellar@smmarketing.us.

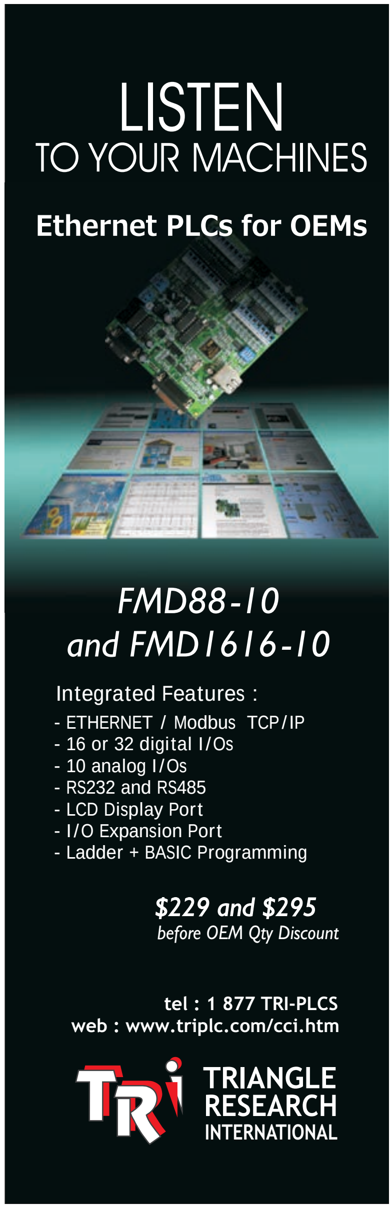
The Vendor Directory at circuitcellar.com/vendor is your guide to a variety of engineering products and services.

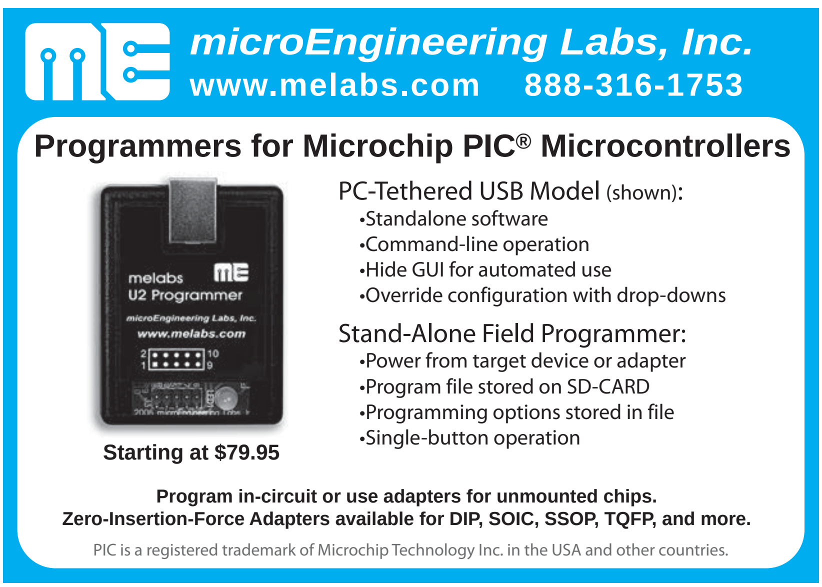

IMAGECRAFT
JumpStarter® C Tools for Cortex™-M and AVR
SIMPLIFYING EMBEDDED DEVELOPMENT
• From $249, 3-user Pro Pack just $1500
• Easy to use - be productive within hours
• Quick startup and build times; up to 90% faster
• Pro features - includes viewing IO registers while debugging
• Direct support by developers
http://imagecraft.com


MaxBotix® Inc.
High Performance Ultrasonic Rangefinders
Save 10%
Online Web-Order Code:
CIRCUIT140531
Valid thru June 1, 2014
HRXL-MaxSonar®-WR™
- High noise tolerance
- IP67 rated
- 1 mm resolution
- Multi-Sensor operation
- Calibrated beam pattern
- Starting at $109.95
XL-MaxSonar®-EZ™
- Great for UAV's and robotics
- Incredible noise immunity
- Small in size
- 1cm resolution
- Automatic calibration
- Starting at $39.95
Phone: 218-454-0766
Email: sales@maxbotix.com
www.maxbotix.com


CCS C Compiler
for Microchip PIC MCUs
NEW Project Watch
Detailed log of the working project
Daily Activity • Time Log • Comments
www.ccsinfo.com/CCpw
sales@ccsinfo.com • 262-522-6500 x 35


PIC-SERVO
MOTION CONTROL
MOTION CONTROLLERS FOR BRUSH, BRUSHLESS AND STEPPER MOTORS.
• controller chips
• controller boards
www.picservo.com
JEFFREY KERR, LLC

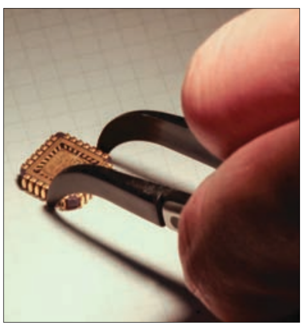

LISTEN
TO YOUR MACHINES
Ethernet PLCs for OEMs
FMD88-10
and FMD1616-10
Integrated Features :
- ETHERNET / Modbus TCP/IP
- 16 or 32 digital I/Os
- 10 analog I/Os
- RS232 and RS485
- LCD Display Port
- I/O Expansion Port
- Ladder + BASIC Programming
$229 and $295
before OEM Qty Discount
tel : 1 877 TRI-PLCS
web : www.triplc.com/cci.htm
TRi TRIANGLE RESEARCH INTERNATIONAL


microEngineering Labs, Inc.
www.melabs.com 888-316-1753
Programmers for Microchip PIC® Microcontrollers
melabs U2 Programmer
microEngineering Labs, Inc.
www.melabs.com
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.

The Future of Monolithically Integrated LED Arrays

By Vincent Lee

Vincent Lee (vincentlee@lumiode.com) is the CEO of Lumiode (www.lumiode.com), a New York City-based start-up company. Lumiode is developing LED microdisplays for head-mounted, head-up, and pico projectors enabling high brightness, low-power, and compact form factor devices based on LED and silicon integration. Prior to starting his company, Vincent graduated with his PhD in Electrical Engineering from Columbia University, where he worked with professor Ioannis Kymissis in the Columbia Lab for Unconventional Electronics. Vincent is secretary for the local chapter of the Society for Information Display, where he helps organize local speakers and events for the display community. He is also a 2013 Fellow in the New York chapter of the Startup Leadership Program.



A prototype emissive LED display chip is shown. The chip includes an emissive compass pattern ready to embed into new applications.

TECH THE FUTURE

LEDs are ubiquitous in our electronic lives. They are widely used in notification lighting, flash photography, and light bulbs, to name a few. For displays, LEDs have been commercialized as backlights in televisions and projectors. However, their use in image formation has been limited.

The developing arena of monolithically integrated LED arrays, which involves fabricating millions of LEDs with corresponding transistors on a single chip, provides many new applications not possible with current technologies, as the LEDs can simultaneously act as the backlight and the image source.

The common method of creating images is to first generate light (using LEDs) and then filter that light using a spatial light modulator. The filter could be an LCD, liquid crystal on silicon (LCoS), or a digital micromirror device (DMD) such as a Digital Light Processing (DLP) projector. The filtering processes cause significant loss of light in these systems, despite the brightness available from LEDs. For example, a typical LCD uses only 1% to 5% of the light generated.

Two pieces are essential to a display: a light source and a light controller. In most display technologies, the light source and light control functionalities are served by two separate components (e.g., an LED backlight and an LCD). However, in emissive displays, both functionalities are combined into a single component, enabling light to be directly controlled without the inherent inefficiencies and losses associated with filtering. Because each light-emitting pixel is individually controlled, light can be generated and emitted exactly where and when needed.

Emissive displays have been developed in all sizes. Very-large-format “Times Square” and stadium displays are powered by large arrays of individual conventional LEDs, while new organic LED (OLED) materials are found in televisions, mobile phones, and other micro-size applications. However, there is still a void. Emissive “Times Square” displays cannot be scaled to small sizes and emissive OLEDs do not have the brightness available for outdoor environments and newer envisioned applications. An emissive display with high brightness but in a micro format is required for applications such as embedded cell phone projectors or displays on see-through glasses.

We know that optimization by the entire LED industry has made LEDs the brightest controllable light source available. We also know that a display requires a light source and a method of controlling the light. So, why not make an array of LEDs and control individual LEDs with a matching array of transistors?

The marrying of LED materials (light source) to transistors (light control) has long been researched. There are three approaches to this problem: fabricate the LEDs and transistors separately, then bond them together; fabricate transistors first, then integrate LEDs on top; and fabricate LEDs first, then integrate transistors on top. The first method is not monolithic. Two fabricated chips are electrically and mechanically bonded, limiting integration density and thus final display resolutions. The second method, starting with transistors and then growing LEDs, offers some advantages in monolithic (single-wafer) processing, but growth of high-quality, high-efficiency LEDs on transistors has proven difficult.

My start-up company, Lumiode (www.lumiode.com), is developing the third method, starting with optimized LEDs and then fabricating silicon transistors on top. This leverages existing LED materials for efficient light output. It also requires careful fabrication of the integrated transistor layer as to not damage the underlying LED structures. The core technology uses a laser method to provide extremely local high temperatures to the silicon while preventing thermal damage to the LED. This overcomes typical process incompatibilities, which have previously held back development of monolithically integrated LED arrays. In the end, there is an array of LEDs (light source) and corresponding transistors to control each individual LED (light control), which can reach the brightness and density requirements of future microdisplays.

Regardless of the specific integration method employed, a monolithically integrated LED and transistor structure creates a new range of applications requiring higher efficiency and brightness. The brightness available from integrated LED arrays can enable projection on truly see-through glass, even in outdoor daylight environments. The efficiency of an emissive display enables extended battery lifetimes and device portability. Perhaps we can soon achieve the types of displays dreamed up in movies.


circuit cellar
cc-webshop
archive store
MISSED an issue?
LOST an issue?
Dog ATE your issue?
No problem!
Circuit Cellar issue PDFs are always available at www.cc-webshop.com. Save $2 off every issue PDF now through the end of May. Use coupon code CCPDF514.
Each issue of Circuit Cellar contains:
• Analysis of the newest embedded technologies
• Electronics engineering insight
• Hardware design
• Programming tips
• Techniques for testing
www.cc-webshop.com
PLUS!
Want the complete issue archive? Get a CC Vault USB drive today. It's packed with every Circuit Cellar issue published through date of purchase.
Ends May 31, 2014
RISE OF THE MACHINES
Robot simulation, design, and the IoT
circuit cellar VAULT