DESIGN: Build a Parallel Processing Computer
ORIGIN: Greece
PAGE: 18

INSIGHT: Experiment with Multipath Fading
ORIGIN: France
PAGE: 44

PROGRAMMING: BASIC for EE Pros
ORIGIN: United States
PAGE: 60

CIRCUIT CELLAR

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

FEBRUARY 2011
ISSUE 247

WIRELESS COMMUNICATIONS

Wireless Network Card Reader

Remote Humidity Control System

Power Via Radio Waves

RF Design Review

PLUS

Battery Check

MCU-Based Parallel NiMH Cell Tester

// 8-Channel Tester Design
// Discharge Currents
// Calibration and Firmware
// Data Formatting

$7.50 U.S. ($8.50 Canada)

www.circuitcellar.com

Low-cost Industrial
Serial to **Ethernet** Solutions

NetBurner
Networking in One Day!

- Instantly network-enable any serial device
- No programming is required for serial to Ethernet application
- Customize to suit any application with a development kit

SBL2e Chip

2-port serial-to-Ethernet server with eight A/D converter inputs and optional SPI, I2C peripheral device support

SBL2e 200

2-port serial-to-Ethernet server with four A/D converter inputs, optional I2C peripheral support and 10-pin header

SBL2e 100

2-port serial-to-Ethernet server with four A/D converter inputs, optional I2C peripheral support and RJ45 connector

SBL2e

Serial-to-Ethernet server with RS-232 support

SBL2e XA

External server with up to four A/D converter inputs, up to eight digital I/O and up to two UARTS (one RS-232)

Hardware Features

Up to three serial ports, 10/100 Mbps Ethernet, up to 10 digital I/O, 12-bit A/D converters, operating temperature -40 to 85 C, 32-bit performance

Software Features

TCP/UDP/Telnet/HTTP modes, DHCP/Static IP support, web-based or AT command configuration

Low Prices

SBL2e Chip .... $10.50 (Qty. 10k, Device P/N: SBL2eCHIP-236IR)
SBL2e 200 .... $19.95 (Qty. 1000, Device P/N: SBL2e-200IR)
SBL2e 100 .... $21.95 (Qty. 1000, Device P/N: SBL2e-100IR)
SBL2e X.......................... $79.00 (Qty. 1000, Device P/N: SBL2eX-100IR)
SBL2e XA........................ $79.00 (Qty. 1000, Device P/N: SBL2eXA-100IR)

Need a custom solution?

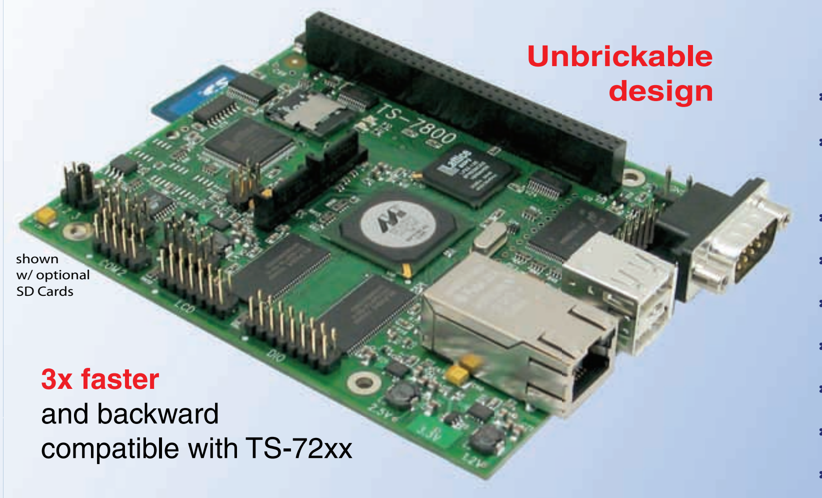
NetBurner Serial to Ethernet Development Kits are available to customize any aspect of operation including web pages, data filtering, or custom network applications. All kits include platform hardware, ANSI C/C++ compiler, TCP/IP stack, web server, e-mail protocols, RTOS, and NB Eclipse IDE.

Information and Sales | sales@netburner.com

Web | www.netburner.com **Telephone** | 1-800-695-6828

Embedded Systems

High-End Performance with Embedded Ruggedness



TS-7800
500 MHz ARM9

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

$229 qty 100

$269 qty 1

TS-TPC-7390
7" Color Touch Panel Computer

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

$449 qty 1

More Touch Panel Computers on our website

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

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

TASK MANAGER

Connectivity, Power, and Apps

How does it connect and at what speed? Does it have enough power? Can it efficiently run essential applications?

Such questions about electronics systems don't pertain only to the embedded design projects under development in corporate or university labs. Nor do such questions arise solely in conversations about technology at the embedded level. Eavesdrop on a few conversations taking place in your local smartphone shop or big-box electronics store and you'll hear similar inquiries. Why? The topics of wireless connectivity, system power, and data/app processing are vital to modern techies, from the hardware and software engineers who read *Circuit Cellar* to plugged-in students to small business owners with growing client lists. This isn't a necessarily novel observation on my part; it's been this way for some time now. What intrigues me is that many of our readers were the forward thinkers who first began asking these questions a few decades ago. (Take a quick look at the article titles in our issues dating back to the 1980s to see what I mean.) And what truly excites me is that our readers are today asking the questions that will change the tech landscape for the next decade.

In this issue, our authors and columnists address three main topics from different angles: wireless connectivity, system power, and application/data processing.

On page 28, Carlo Tauraso presents his "MiWi" network card reader design. Another innovative application—a contest-winning application, if fact—is David Penrose's remote humidity control system (p. 36). The design can calculate water vapor pressure, control a ventilation system, and more. Users have remote access. And speaking of system access, you could apply the tips Jeff Bachiochi provides in "Serving Up BASIC" to develop a webpage for operating a control or monitoring system (p. 60).

On the topic of RF technology, columnists George Novacek and Robert Lacoste weigh in with need-to-know information for R&D project managers and hands-on designers alike. In "RF Design Review," George reflects on the decades of R&D that went into developing the wireless technologies we use today (p. 16). Robert focuses on the potentially ruinous phenomenon of multipath fading (p. 44). Be sure to read this article before starting your next RF system.

Want to develop a computation-intensive application? Check out Antonios Chorevas's parallel processing computer (p. 18). The PIC-based design is used to fine-tune parallel algorithms.

And then there's power. On page 52, Ed Nisley explains how he built an eight-channel parallel cell tester with an Arduino Mega microcontroller board. If you're interested in an alternative to battery power, check out Tom Cantrell's article about technology for delivering power via radio waves (p. 66). This exciting, next-generation technology is sure to provide your future designs with the boost they need to succeed in the field.

cj@circuitcellar.com

C. Abate

CIRCUIT CELLAR®

THE WORLD'S SOURCE FOR EMBEDDED ELECTRONICS ENGINEERING INFORMATION

Circuit Cellar's mission is to collect, select, and disseminate need-to-know information around the world in the fields of embedded hardware, embedded software, and computer applications. Circuit Cellar uses an assortment of print and electronic content-delivery platforms to reach a diverse international readership of professionals, academics, and electronics specialists who work with embedded, MCU-related technologies on a regular basis. Our aim is to help each reader become a well-rounded, multidisciplinary practitioner who can confidently bring innovative, cutting-edge engineering ideas to bear on any number of relevant tasks, problems, and technologies.

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

ADVERTISING REPRESENTATIVE
Peter Wostrel
Strategic Media Marketing, Inc.
1187 Washington St., Gloucester, MA 01930 USA
800.454.3741 • 978.281.7708
peter@smmarketing.us • www.smmarketing.us
Fax: 978.281.7706

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

Cover photography by Chris Rakoczy—Rakoczy Photography
www.rakoczyphoto.com
PRINTED IN THE UNITED STATES

CONTACTS

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

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

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

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

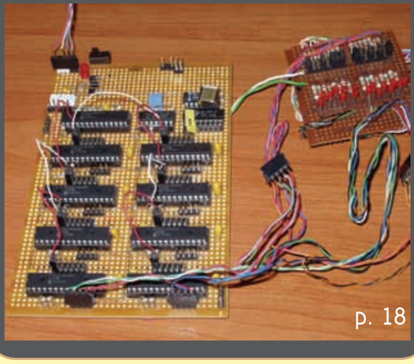
CIRCUIT CELLAR®, THE MAGAZINE FOR COMPUTER APPLICATIONS (ISSN 1528-0608) is published monthly by Circuit Cellar Incorporated, 4 Park Street, Vernon, CT 06066. Periodical rates paid at Vernon, CT and additional offices. **One-year (12 issues) subscription rate USA and possessions $45, Canada/Mexico $60, all other countries $63. Two-year (24 issues) subscription rate USA and possessions $80, Canada/Mexico $110, all other countries $116.** All subscription orders payable in U.S. funds only via Visa, MasterCard, international postal money order, or check drawn on U.S. bank. **Direct subscription orders and subscription-related questions to Circuit Cellar Subscriptions, P.O. Box 5650, Hanover, NH 03755-5650 or call 800.269.6301.**

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

INSIDE ISSUE 247

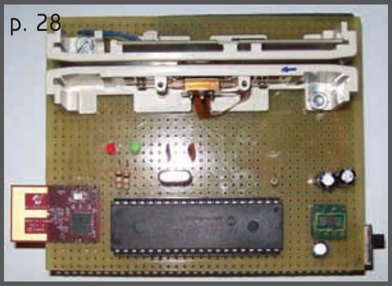
February 2011 • Wireless Communications

Parallel Computing



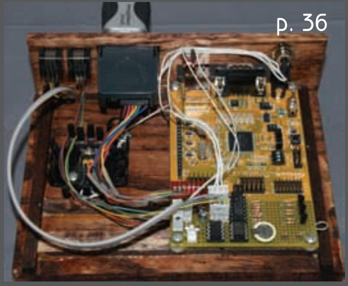
p. 18

Wireless Access



p. 28

Remote Humidity Controller

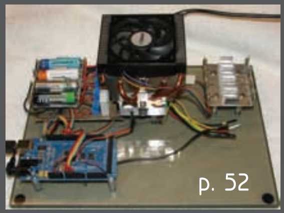


p. 36

Multipath Fading Experiment

p. 28

Build a Cell Tester



p. 52

MULTIFUNCTION WIRELESS PROGRAMMABLE TERMINAL SERVER

The **ConnectPort LTS** is the industry's first multifunction, wireless, programmable terminal server. The server provides serial-to-IP connectivity and optionally ZigBee wireless communications. This minimizes equipment space requirements and reduces cost by combining a terminal server and wireless gateway in one device. The server is ideal for utilities, security, building automation, retail, industrial automation, healthcare, data center management, and other applications.

ConnectPort LTS serial servers deliver secure, reliable, flexible, and cost-effective RS-232/422/485 serial-to-Ethernet connectivity and are available in eight-, 16-, and 32-port options. The wireless version features ZigBee communications, an internal modem, and USB 2.0 ports for expandability. All ConnectPort LTS serial servers include dual GB Ethernet for extreme reliability, an SD memory slot for data storage, and a large LCD screen for easy configuration, monitoring and diagnostics.

The Linux-based ConnectPort LTS is programmable with Python support and a software development kit. Advanced software features include an IPv4/v6 dual stack, user port sharing, port logging, SSHv2, and SSL.

The ConnectPort LTS is available now starting at **$1,283**. The wireless ConnectPort LTS W is available now starting at **$1,511**.

Digi International
www.digi.com

WINDOW/DOOR SENSOR SAVES ENERGY

The ILLUMRA solar-powered, wireless **Door/Window Sensor** maximizes the energy savings of heating and air conditioning systems by providing status information about windows and doors. Energy waste can be reduced by 20% to 60% by disabling blowers and adjusting temperature set points in HVAC systems when windows and doors are left open. The wireless Door/Window Sensor is a key component to reducing energy waste in hotels, condominiums, and dormitory buildings.

The wireless, battery-free sensor communicates using the EnOcean protocol to a variety of compatible actuator and controller products: thermostats, relays, room controllers, and BACnet and Ethernet gateways for integration with energy-management systems. The sensor also includes on-board energy storage that can operate the device in darkness for several days.

The sensor is small—measuring only 3.86″ × 0.62″ × 0.81″—and attaches easily to doors and windows (peel and stick). The RoHS-compliant sensor is made of recyclable plastic, qualifies for LEED credits, and helps buildings comply with International Green Construction Code standard 189.1.

Please contact an ILLUMRA distributor for pricing.

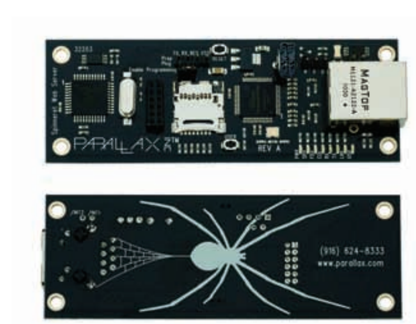
ILLUMRA
www.illumra.com

TINY WEB SERVER

The **Spinneret Web Server** may be small—at less than 1.5″ × 4″—but it is a feature-packed development platform. As an open-source hardware design, the layout, schematics, and firmware are available under licenses that allow free distribution and reuse. This means that the Spinneret Web Server's design can be incorporated into new applications royalty free and without a nondisclosure agreement.

The Spinneret Web Server is an Ethernet-based development board for the Propeller microcontroller. Webpage content, files, and logs can be stored on a MicroSD card. The serial EEPROM has 32 KB for storing a Propeller program and 32 KB for non-volatile data storage, independent of the MicroSD card. There is a real-time clock controller for time stamping files and events and a back-up capacitor that will keep the clock running through extended power outages. There is a serial programming header and two auxiliary I/O connections, one for level-shifted open collector communications over a three-pin data/power/ground cable, and a the second is a 12-pin socket for direct 3.3-V I/O connections. There are eight status LEDs on the PCB, plus two that are repeated on the Ethernet jack. One of the status LEDs is user-controllable and shares a line with a button that can be read under user control. A second button resets the Propeller to reload the firmware from the EEPROM.

The Spinneret Web Server costs **$49.99**.



Parallax, Inc.
www.parallax.com

NEW PRODUCT NEWS

Edited by John Gorsky

DATA ACQUISITION MODULES FOR DISTRIBUTED CONTROL

Sealevel has announced two additions to the popular Seal/O family of modular I/O, the **Seal/O-530** and **Seal/O-540** digital I/O modules. Offering a powerful selection of open-collector outputs and optically isolated inputs, the modules are designed to interface to a variety of real-world I/O. Ordering options allow control from a host device via wireless, Ethernet, RS-485, USB, or RS-232.

The Seal/O-540's 32 open-collector outputs are well-suited to controlling common industrial peripherals, and each output circuit includes a flyback diode for protection when interfacing highly inductive loads such as DC motors. The Seal/O-530 combines 16 optically isolated inputs with 16 open collector outputs. The nonpolarized inputs can monitor 5 to 30 VDC. Removable terminal blocks are standard on both modules.

Both modules are housed in a rugged, metal enclosure and are also available in board only versions, which allow easy integration into OEM systems. The modules feature a standard operating temperature range of –25° to 85°C and an extended temperature range of –40° to 85°C is available.

As a result of a technology refresh, pricing has been recently reduced on all Seal/O products. Pricing for the Seal/O-530 products starts at **$359** while the Seal/O-540 series starts at **$379**.

Sealevel Systems, Inc.
www.sealevel.com

NPN

EASY-TO-IMPLEMENT USB 2.0 MODULE

The **QuickUSB QUSB2** module makes adding Hi-Speed USB 2.0 to new or existing products fast and easy by integrating all the hardware, firmware, and software needed to implement a general-purpose USB endpoint into a simple plug-in module and development library. Under ordinary circumstances, implementing a USB peripheral requires a functional understanding of the USB protocol as well as a considerable amount of firmware and software development and stringent compliance testing. The QuickUSB Module provides a desirable alternative for adding high-speed USB 2.0 for speeds as much as 40 times faster than USB 1.1.

The module may be used as a development station when combined with the QuickUSB Adapter Board or QuickUSB Evaluation Board. It may also be designed as a plug-in module for new products, or designed directly into new products and licensed using the QuickUSB iChipPack or QuickUSB EEPROMs.

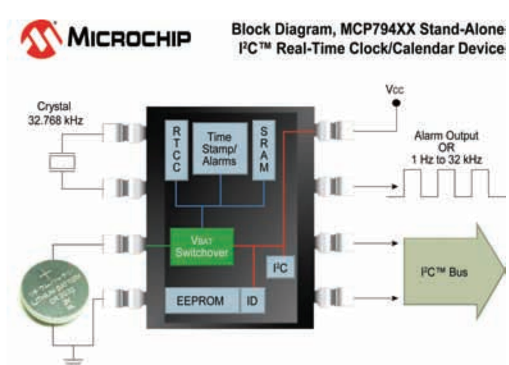
The QuickUSB QUSB2 Module is 2" × 1.5" and implements a bus-powered high-speed USB 2.0 endpoint terminating in a single 80-pin target interface connector. The target interface consists of one 8- or 16-bit high-speed parallel port, up to five general-purpose 8-bit parallel I/O ports, two RS-232 ports, one I²C master port, and one SPI master port. Also included is one FPGA configuration port (Altera PS or Xilinx SS), 2 KB of nonvolatile memory, software, libraries, and drivers for Windows 32/64, Linux, and Mac OS X.

The QuickUSB QUSB2 module costs **$149**.

Bitwise Systems
www.quickusb.com

STAND-ALONE RTCC FAMILY

The Microchip Technology **MCP794xx** is a new stand-alone I²C Real-Time Clock/Calendar (RTCC) family. All six MCP794xx devices are highly integrated at a low cost, including ample amounts of on-chip EEPROM and SRAM, as well as a user-lockable section of EEPROM available for a 64-bit reprogrammable unique ID that can be factory-programmed with a MAC address. The devices include digital trimming for time-of-day calibration, and a battery-switchover feature that supports back-up power at very low voltage and current levels.



With their on-chip battery-switchover circuit and power-fail timestamp, the MCP794xx RTCC devices help to address system health, safety, and security concerns in applications involving the storage of perishable goods, or monitoring access to secure rooms. The devices can be ordered with a pre-programmed MAC address, which eliminates a time-consuming step in the production flow, and the digital trimming feature can support software temperature compensation, which lowers costs in comparison to devices where temperature compensation takes place in hardware.

All six MCP794xx RTCC devices are available in eight-pin MSOP, SOIC, TSSOP, and TDFN packages. The MCP79400 is priced at **$0.64** each. The MCP79401, MCP79402, and MCP79410 each cost **$0.70**. The MCP79411 and MCP79412 cost **$0.76** each. All are prices are for 10,000-unit quantities.

Microchip Technology, Inc.
www.microchip.com

HIGH-PERFORMANCE 2.3-TO-2.7-GHz POWER AMPLIFIER

The **RF5632** is a new power amplifier IC optimized specifically for WiMAX systems. The amplifier can be designed into multiple applications, including customer premises equipment, gateways, access points, LTE wireless infrastructure, and Wi-Fi-based wireless high definition interface for wireless video distribution networks.

The RF5632 integrates a three-stage PA and power detector into an industry-leading 4 mm × 4 mm QFN package, significantly minimizing customer design footprint requirements. Additionally, the RF5632 operates from a standard 5-V supply, eliminating additional power supply requirements, enhancing design flexibility, and lowering bill-of-material costs. The RF5632 is also fully DC and RF tested—including EVM performance—at the rated output power, maximizing application yields and accelerating time to market.

The RF5632 delivers an EVM of 2.5% and meets or exceeds WiMAX and LTE spectral mask requirements with an output power of 28 dBm in the 2.3 to 2.4 GHz, 2.4 to 2.5 GHz, and 2.5 to 2.7 GHz frequency ranges. The bias of the PA may be controlled to accommodate a 22-dB gain step to increase the dynamic range of the system. The RF5632 offers high gain of 34 dB and high linear output power, with best-in-class efficiency. The RF5632 maintains linearity over a wide range of temperatures and power outputs while the external match enables tuning for output power over multiple bands. The RF5632 also features internal input and inter-stage matching, a power-down mode and power detection.

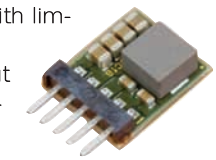
The RF5632 starts at **$3.10** per 10,000 units.

RF Micro Devices, Inc.
www.rfmd.com

LOW-INPUT VOLTAGE SIP MINI POL CONVERTERS

FDK extends its DK Series of nonisolated, point-of-load (POL) DC-to-DC converters with the addition of three new converters that operate from a 3.0-to-5.5-VDC input bus. These small, low-profile SIP devices deliver up to 3, 4, and 6 A of output current with efficiencies up to 94%. They provide a tightly regulated programmable output voltage in the range of 0.8 to 3.465 V. The 3- and 4-A models deliver the full rated output current with no de-rating in ambient temperatures up to 85° and 75°C, respectively, under natural convection. The 6-A converter delivers the full rated output current with no de-rating up to 60°C with 400 LFM (2 m/s) airflow.

Occupying a footprint of less than 0.08 square inches for the 3- and 4-A models, and 0.09 square inches for the 6-A model, they provide significant space savings over discrete on-board solutions. All models have a low height of 0.472”, making them particularly suitable for high density racks, blade systems, and low-profile consumer applications with limited board-to-board and board-to-case clearances.



The DK Series converters provide complete power conversion solutions without the need for external input and output capacitors. They are true single input converters in that they do not require an external bias supply for operation. They feature programmable output voltage, remote on/off, over-current and over-temperature protection, and start up into a pre-biased output.

Pricing for the FPDK05SR8003PSV model starts at $5 in 1,000-piece quantities.

FDK America, Inc.
www.fdk.com

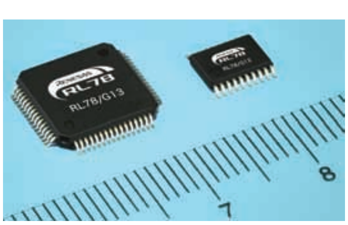
NEW RL78 MICRO FAMILY

The Renesas RL78 family is a new series of extremely low-power microcontrollers. The RL78 integrates advanced features from both the R8C and the 78K (78K0, 78K0R) families to deliver lower power, enhanced performance, and higher integration, providing a robust migration path. The new products are built around the company’s new RL78 CPU core, which is based on the low-power, high-performance 78K0R CPU core, and incorporates an extensive range of powerful peripheral functions from the R8C and 78K families, making them ideal for a number of applications including battery-operated devices and household applications.

The new MCUs achieve industry-leading low-power consumption, with operating current of only 70 µA/MHz and standby current of 0.7 µA with the RTC and low-voltage detection functions active. The new RL78 MCUs’ advanced process technology enables higher integration of on-chip components such as a 32-MHz ±1% on-chip oscillator, data flash with background operation supporting up to 1 million rewrite cycles, and 1.6- to 5.5-V operation.

Pricing will differ according to memory capacity, package type, and pin count. As an example, the RL78/G12 Group MCUs with 2 KB of flash memory and 256 bytes (B) of RAM in a 20-pin shrink small-outline package (SSOP) cost $0.45 per unit in 10,000-piece lots.

Renesas Electronics America, Inc.
www.renesas.com



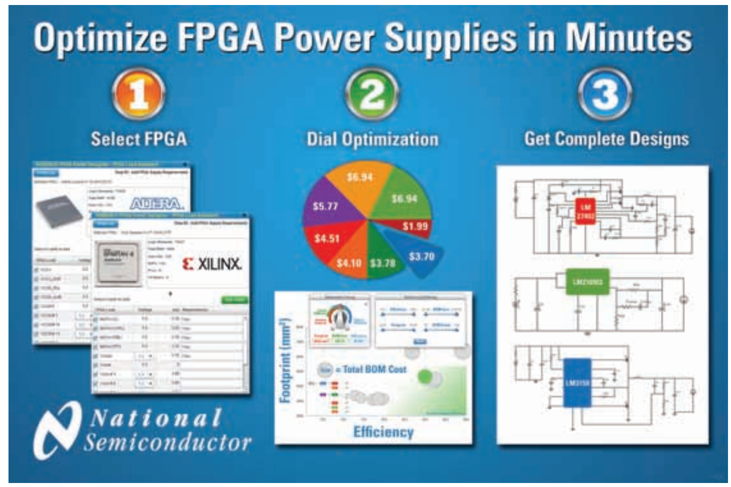
NPN

NEW TOOL ACCELERATES DESIGN OF MULTIPLE LOAD POWER SUPPLIES FOR FPGAs

WEBENCH FPGA Power Architect is the industry's first design tool to model and optimize power supplies for FPGAs in minutes. WEBENCH FPGA Power Architect incorporates the detailed supply requirements of more than 130 of the newest FPGA devices from Altera and Xilinx.

Modern power supply systems, including advanced FPGAs, are complex to design, often incorporating multiple unique loads to drive precisely specified voltages. In addition to the required voltages and currents, each load may have specific limitations for ripple, noise filtering, synchronization, and separation of supplies, as well as start-up definitions. The power supply designs delivered by WEBENCH incorporate comprehensive power requirements published by the FPGA manufacturers, giving the designer confidence that their power supplies will meet these constraints while saving critical time in the process.

To begin a design, the designer selects an advanced FPGA from Altera or Xilinx and the tool automatically populates the unique power requirements, identifying core and I/O options for every potential load of that array. Once tuned by the designer, WEBENCH collects all the loads and creates multiple power supply architectures as options for driving the FPGA and the total system.



The power supply may involve one or more intermediate voltage rails between the input supply and the point-of-load regulators. The designer can tune the recommended power supply with the turn of a dial, reducing size, increasing efficiency, or lowering complete system cost in seconds. The designer can also order components for prototyping, share the complete system with others, or easily print a complete project report including schematics, bill of materials, and performance characteristics.

Go to www.national.com/FPGAarchitect to start a design.

National Semiconductor Corp.
www.national.com

NEW INDUSTRY STANDARD FOR LOW-COST TOOLS FOR 32-BIT MICROCONTROLLERS

The **STM32 Discovery Kit** is a hardware development platform priced at less than $10. It's supported by either free or low-cost downloadable development tools from major third-party vendors Atollic, IAR, and Keil.

As an ultra-low-cost and convenient starter platform, the STM32 Discovery Kit is particularly suited to the STM32 Value Line microcontrollers. The USB-powered board is straightforward to set up and use by plugging into a PC. It is populated with a 24-MHz, 64-pin STM32F100RBT6B Value Line device featuring a 128-KB flash memory plus multiple timers, analog peripherals, and industry-standard serial interfaces. The included ST-LINK in-circuit debugger allows work to start without further costs. Extension connectors provided on the board enable easy connection to other boards or devices for deeper testing of the microcontroller peripherals. In addition, 15 ready-to-run applications are available to help developers evaluate microcontroller features.

The kit is a fully integrated development environment including board, debugger, programmer, and compiler. Developers can benefit from the kit's extensive capabilities as a low-risk and low-investment prototyping platform.

The STM32 Discovery Kit is now available.

STMicroelectronics
www.st.com

FIRST TRUE 0.9-V MICROCONTROLLER

The Texas Instruments **MSP430L092** —the newest addition to the ultra-low-power MSP430 MCU portfolio—is the industry's first true 0.9-V MCU. The MCU enables multiple applications to run from a single-cell battery, and it offers developers increased flexibility with programmable analog building blocks.

Unlike existing MCUs that claim 0.9-V technology, the MSP430L092 inherently operates at 0.9-V, including the entire analog and digital logic. Because it operates at 0.9-V, the MSP430 MCU does not require an on-board boost converter, lowering the entire system's power consumption. It also reduces the need for external circuitry required by traditional solutions. This allows developers to run applications such as electric toothbrushes, razors, toys, and security devices off a single-cell battery, ranging from an AAA battery to a coin cell. MSP430L092 also offers programmable analog building blocks that can be configured as five different peripherals. In addition to free software, comprehensive application notes, code libraries, and community support, new tool kits are available to help developers fully leverage the benefits of this industry-first device.

Key features and benefits of the MSP430L092 include three variations with up to 2-KB RAM and 2-KB ROM. An integrated analog functions pool (A-POOL) can serve as an ADC, DAC, system voltage supervisor, temperature sensor, or comparator eliminating the need for external components. The programmable A-POOL enables the various analog peripheral configurations to run sequentially without user interaction, providing increased flexibility.

The new MSP430L092 MCUs cost **$0.85** at 10,000 units. The supporting MSP-TS430L092 (**$99**) and MSP-FET430U092 (**$149**) development tools are immediately available to customers.

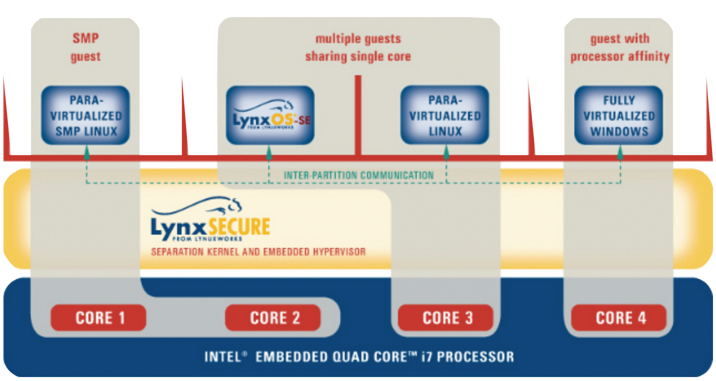
Texas Instruments
www.ti.com

LynxSecure OFFERS MULTI-OPERATING SYSTEM SUPPORT

LynuxWorks recently announced the availability of **LynxSecure 4.0** on the latest hardware platforms from Intel. The Core i7 family has been widely used in the desktop and laptop market, and now the quad-core versions of the processor are available for embedded designs.

LynxSecure 4.0 provides the most flexible virtualization solution for use in embedded systems. Built from the ground up, and now in its fourth generation, LynxSecure offers the combination of security with functionality, allowing embedded designers to use the latest software and hardware technologies to build complex multi-operating systems.

The availability of LynxSecure on the latest quad-core processors from Intel allows large GUI-based operating systems, such as Windows or Linux to securely co-reside with more traditional embedded real-time OS solutions such as the LynxOS family of RTOSes from LynuxWorks. The advanced software virtualization in LynxSecure is integrated with the hardware virtualization technologies—such as vt-x and vt-d—on the Intel processors to give native performance and functionality of all the OSes that are running as "guests."



Another key feature that LynxSecure offers is the ability to run guest OSes that have symmetric multiprocessing capabilities. For embedded systems that require a sophisticated user interface coupled with networked connectivity, and also hard real-time data response, the combination of LynxSecure with the quad-core Intel core-i7 processor allows all of this functionality to be easily developed, or migrated from existing systems, giving an unprecedented development window for the next generation of complex embedded systems.

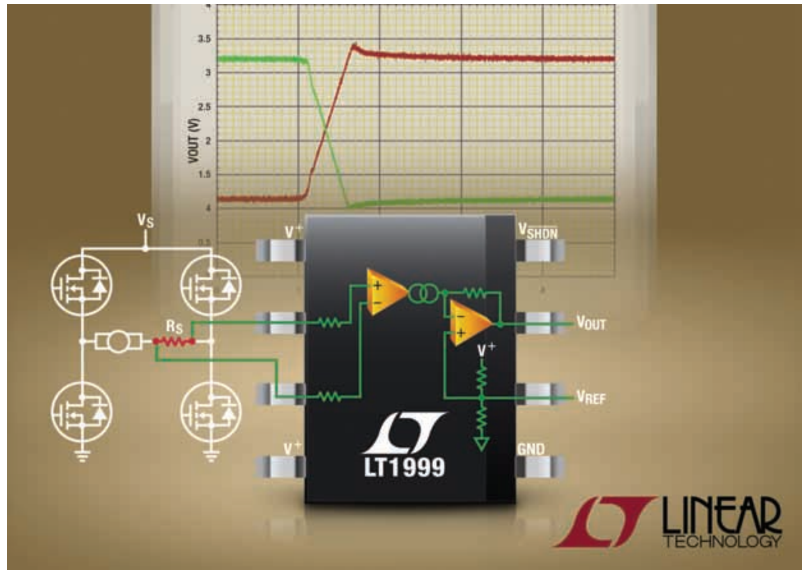
You can contact LynuxWorks for pricing and additional information.

LynuxWorks, Inc.
www.lynuxworks.com

BIDIRECTIONAL CURRENT SENSE AMPLIFIER

The **LT1999** is a high-voltage amplifier designed for bidirectional current sensing in the presence of fast-switching, common-mode voltages from –5 to 80 V. This wide-input common-mode range is obtained with standard low-voltage op-amp supplies. A 2-MHz bandwidth enables the LT1999 to monitor currents in H-bridge motor controls, switching power supplies, solenoid, and battery chargers. To maintain outstanding accuracy, even when monitoring motor and solenoid current, the LT1999 guarantees more than 80 dB of common-mode rejection at 100 kHz. This ensures that the LT1999 remains insensitive to the large square wave input common-mode voltages present in these applications.

The LT1999 features a buffered output with three fixed-gain options for 10, 20, and 50 V/V. By default, the LT1999 output voltage is referenced halfway between the supply voltage and ground, enabling the output to indicate both the magnitude and direction of the sensed current. In addition, the output bias level can be set via a separate input. Gain error is guaranteed at less than 0.5% and input offset voltage is guaranteed at less than 1.5 mV over the full temperature range of –40° to 125°C. The LT1999 operates with an independent 5-V supply voltage, drawing only 1.5 mA while active and only 10 µA in shutdown mode.



Offered in an eight-lead MSOP package and eight-lead SOP package, the LT1999 family is fully specified over four temperature ranges: 0° to 70°C, –40° to 85°C, –40° to 125°C, and –55° to 150°C. The LT1999 is in full production, with prices starting at **$1.92** each in 1,000-piece quantities.

Linear Technology Corp.
www.linear.com

NPN

CIRCUIT CELLAR

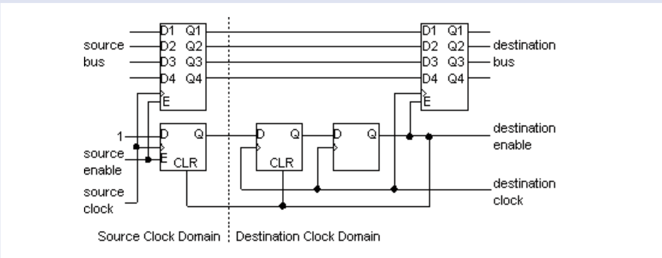
Test Your EQ

Edited by David Tweed

ANSWERS for Issue 246

Answer 1—The obvious multi-bit synchronizer built with dual pipeline registers does indeed address the problem of metastability, but it leaves open the possibility that different data bits in the bus will change at different clock edges in the destination domain. This means that the bus will occasionally contain "invalid" values that are a mix of bits from the previous value and bits from the new value.

Answer 2—In systems where the data changes less often than every three to four clocks of the destination domain, you can use a single pipeline register to capture the data. The trick to avoiding metastability is to enable this register only when the data isn't changing. For this, you create a separate "update" signal that gets passed in the usual way from the source domain to the destination domain. One possible implementation is shown here.



This implementation presumes that there is an "enable" or "valid" signal that flows in parallel with the data in each clock domain anyway, and this circuit creates an enable pulse in the destination domain for each one that occurs in the source domain. Note that two of the flip-flops have an asynchronous clear function.

Answer 3—Ethernet relies on message-level error checks to detect collisions, which means that the two messages that collided are both lost and must eventually be retransmitted. Since there's a nonzero chance that a collision can happen on every (re-)transmission, there's no guaranteed upper bound on the delivery time of any particular message.

On the other hand, CAN transmitters monitor the physical medium on a bit-by-bit basis, and if two of them happen to start sending at the same time, the first one to send a "recessive" bit at the same time the other is sending a "dominant" bit will recognize this at once and abort its message. The other transmitter continues; none of the other devices on the network are aware of the collision, and that message experiences no delay. After that, the transmitter that experienced the collision is free to try again, and it won't risk a collision with that same message again.

Even if it experiences a collision with a different higher-priority message, there is a definite limit on the number of those that can occur, and therefore, there's a definite upper bound on the delivery time of the message. This makes CAN suitable for "hard" real-time applications for which ordinary Ethernet cannot be used—although there are now higher-level protocols that have been invented that can be used with Ethernet to mitigate this problem.

Answer 4—Since there's no requirement for collisions to be detectable within one bit time, the raw Ethernet bit rate can be much higher, which means that the average network throughput can be correspondingly higher. CAN typically maxes out at about 1 Mbps (on small networks), while Ethernet starts at 10 Mbps and goes up from there.

Contributed by David Tweed

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

by George Novacek (Canada)

RF Design Review

Wireless technology is all the rage. But it took decades of research and development to usher in this "wireless era." As you'll see, the uses of wireless technology might have changed, but many of the fundamental technologies and theories remain the same.

The other day I finished transferring many of my paper archives into soft copies, a task that took me a few years. Soft archives need no space, and they're searchable, but they take forever to build. Not only is my scanner slow, but quite often, as I pull those yellowish papers out of the box, I read, reminisce, and the time flies.

This time, I ran across my old designs of wireless security devices. It struck me how the technology has progressed and yet the fundamental issues have remained the same. The old designs are valuable for understanding the underlying principles which haven't changed over the time.

WIRELESS EVERYWHERE

The proliferation of wireless devices as we see it today has been driven, in my mind, by the success of the cell phone. It provided the volume, fueled research and development into the new technology, lowered the cost, and put pressure on the governing bodies to open up access to the radio waves. Authorities have always been very protective of the RF spectrum, but public demand has made several frequency bands, especially when very limited radiated power is used, available for many useful applications, with minimum bureaucracy to contend with.

The major difficulty with designing RF devices was the need for expensive test equipment, although there always were ways around it, as ham radio amateurs can confirm. I have fond memories of tuning transmitters by having a guy hold an ordinary 4′ fluorescent light bulb in hand, standing a few feet away from the antenna, while I was tweaking the output stages to obtain the maximum brightness of the bulb. But that's all in the past. Monolithic modules now widely available make life simpler. Why design RF modules when they can be purchased, at much lower price than development, over the counter?

WHY DESIGN?

A typical transmitter used for security devices, garage door openers, car locks, and remote controls is shown in Figure 1. At one point in time, the operating frequency was in the 27-MHz citizen band (CB). It was largely by necessity, as that band was available for use and transistors capable of operation at these frequencies were available at a reasonable price. The disadvantage of the low frequency was its 11-m (36.4′) wavelength λ. A 1/4 λ whip antenna would have been 2.75 m (9.1′) long—quite impractical for the intended purpose. Anything shorter than that meant a significant loss of antenna efficiency.

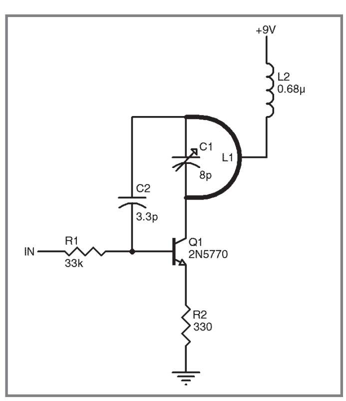


Figure 1—This is a typical 315-MHz transmitter.

Eventually, the 315-MHz band became popular. With λ =

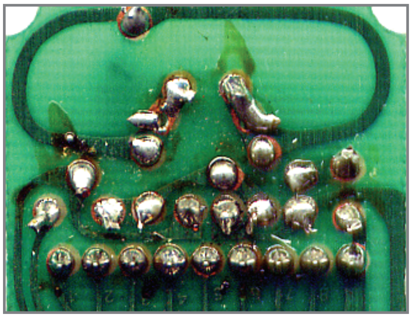


Photo 1—Transmitter PCB trace

0.95 m (3.1′), reasonably sized resonant circuits became practical, often allowing the coils to be a PCB trace. This simplified their construction and ensured manufacturing consistency and reduced the cost. Refer to Photo 1. Sometimes an antenna was also etched on the PCB, but usually, the resonant coil loop area radiated enough energy to exceed the FCC Part 15 limit. During 9-V operation, for instance, R2 was needed to reduce the radiated power.

For remote control, the transmitter operated as a continuous wave (CW) device, its carrier being keyed on and off (called OOK mode) by the input signal. The Motorola MC145026 encoder was a popular IC to generate a train of trinary pulses to drive the transmitter Q1. Today, microcontrollers are often used instead.

The superregenerative receiver was the king. Figure 2 shows its typical schematic. Photo 2 shows its input coil implementation. Once again, the resonant circuit and the antenna coupling were etched on the PCB.

The deceptively simple receiver features an amazing RF sensitivity in the range of a few microvolts. Its design is quite complicated and beyond the scope of this column. In a nutshell, Q1 operates simultaneously at three frequencies: The radio frequency determined by the L1/C2 tank, a supersonic quench frequency controlling positive feedback, and the baseband data frequency (voice also could be used). The baseband data appearing at the junction of L2, R8, and C4 is fed through R7 to a low-frequency amplifier Q2, with frequency roll-off to suppress the quench. The amplified data would be cleaned up in a following Schmitt trigger and fed into an MC145027 or a microcontroller to be decoded.

The transmission range has always been a frustrating issue. In an open field, with the devices placed on 30′ posts, 200′ range was common. But in a building, it was a completely different story. Multipath and absorption, added to by the frequency drift of the LC tuned devices, could reduce the usable range to merely a few feet.

As the transmitted power was limited by the regulations, one could only improve the range by improving the reception. Reducing the data bandwidth and replacing the simple R3/C6 filter in the receiver with a sharp active filter could help a bit, but it required a trade-off of other characteristics. An RF preamplifier, when used, was primarily to isolate the antenna from the superregenerative circuits to limit their notorious back radiation to satisfy the FCC regulations. It did little for sensitivity. A superheterodyne (superhet) receiver with its narrow bandwidth could improve the reception, but its significantly higher cost aside, the transmitters' frequency drift, also caused by proximity of metal objects, was a serious concern.

The help arrived with the surface acoustic wave (SAW) resonators. These inexpensive devices provided frequency stability rivaling crystals while replacing the tuning capacitors C1 in the transmitters with minimal changes to the rest of the circuit. That enabled narrow-band receivers to be effectively used to improve the reception. New monolithic receivers, including superhets, needing just a few external parts keep the cost low. Adding to the receiver an array of antennas, usually two, such as wireless routers do, alleviated the multipath problem, making the connection a lot more reliable without the frustration of having to experiment with their location.

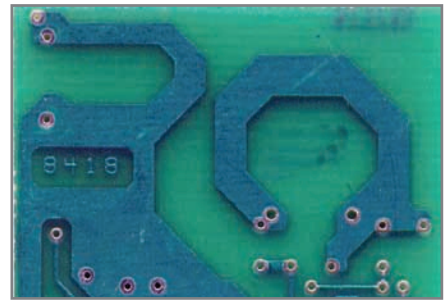


Photo 2—Receiver coils

WIRELESS TODAY

This is pretty well the situation today. Some remote controls, security devices, and industrial devices moved to higher frequencies, such as 915 MHz or 2.4 GHz. But the practical range remains limited to generally 100′ to 200′, because while higher frequencies allow higher transmitted power and more efficient antennas, they also suffer from higher path losses.

The environmental noise is the main range-limiting factor. Once the noise floor is reached by the receiver, there isn't much more that can be done to increase the range without radiating more power. Improving the noise figure of the receiver front end below the floor doesn't work. There exist correlation techniques to recover a signal buried in noise, but the customer is unlikely to pay for the necessary computing power to be able to unlock his car from a little further down the street. ◼

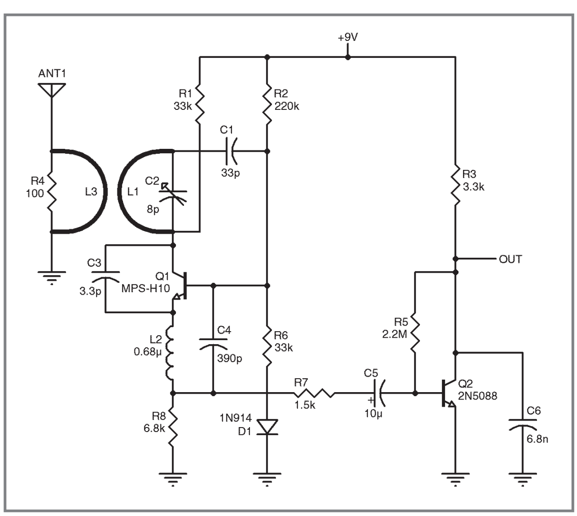


Figure 2—Superregenerative receiver

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

by Antonios Chorevas (Greece)

Parallel Processing Computer

A Design for Solving Computation-Intensive Problems

This parallel processing computer uses multiple microcontrollers for problem-solving applications. The PIC-based design can be used for the development and fine-tuning of parallel algorithms.

I constructed a computer that uses the combined computational power of several Microchip Technology microcontrollers to solve a problem (see Photo 1). The computer is defined as a "parallel processing" machine, and the algorithms that are executed on it are called "parallel algorithms." You can use the system to help develop, understand, and fine-tune many parallel algorithms. You also can use it to achieve high performance in digital signal processing (DSP) applications.

The computer is fairly easy to design. But using it efficiently requires some effort because the development and implementation of parallel algorithms is a rather difficult task. In this article, I'll briefly introduce the topic of parallel-processing computers. I'll then describe the hardware and supporting software (i.e., the software that runs on the PC plus the firmware on the "parallel computer" board). Lastly, I'll describe some parallel algorithms that run on this parallel computer.

The code for this project is posted on the *Circuit Cellar* FTP site. The source code for samples is available in the ParallelPIC_examples.zip file. The code for the supporting software that runs on a PC is in ParallelPIC_PCcode_V2.zip. The source code and the hex files for the firmware are in the ParallelPIC_firmwareV2.zip file.

PARALLEL PROCESSING

My design is a single instruction multiple data (SIMD) type of parallel computer. This means all the MCUs execute the same program, but each MCU works with different data. (As a matter of fact, there are instances when only one subset of the MCUs executes actual code while the other MCUs do nothing. This is inherent in some parallel algorithms. I'll describe this in detail later in this article.)

The MCUs communicate with other MCUs to exchange data. Thus, a "network" is developed between them. The network's topology affects the design of the algorithms that are executed in the parallel machine. The parallel machine can be represented graphically by using circles for the MCUs and arcs for the interconnections between them. (There may be one-way or two-way communication between two MCUs.) Figure 1 depicts two common topologies.

Let's look at an example of how to use the tree. Suppose the problem is to find the maximum among eight numbers. Use the tree structure. MCUs 1, 2, 3, and 4 are called "leaves." MCU 7 is called a "root." The leaves get the input data. Each MCU compares the two incoming data values, finds the maximum, and passes it to the "upper" MCU. For example, MCU 5 might get two values from MCUs 1 and 2 and pass the maximum to MCU 7. This read-compare-send algorithm is executed repeatedly. This way, the root MCU 7 holds the maximum of the eight

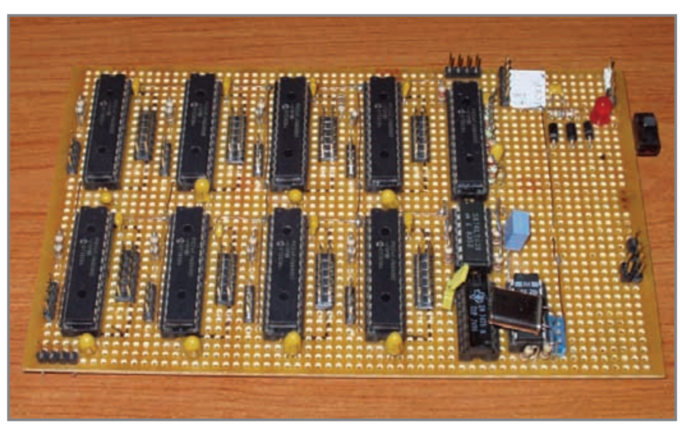


Photo 1—The parallel computer prototype featuring eight MCUs

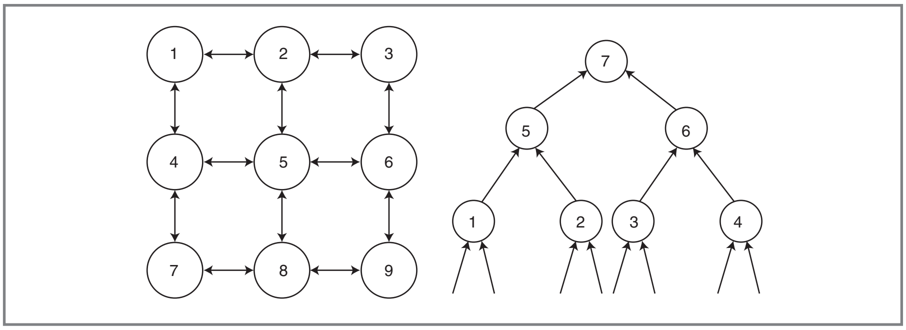


Figure 1—Mesh and tree topologies

input numbers after three executions of the read-compare-send algorithm. In the first execution, MCUs 1, 2, 3, and 4 send correct results to their upper MCUs. In the second execution, MCUs 5 and 6 send correct results. And finally, in the third execution, MCU 7 has the maximum.

Many algorithms have been developed for solving problems in this type of parallel machine that cover many areas, from elementary operations (e.g., enumeration) to complex operations (e.g., Fourier transforms, optimization, and so on). The interconnection network plays an important role in the efficiency of the algorithm. The most powerful network is the one where each MCU communicates with all the others. However, this is not a practical network, so all the algorithms concentrate on simpler networks. A critical parameter is the number of connections for each MCU. The mesh topology requires that each MCU is connected to at most four other MCUs. The binary tree requires three connections per MCU. A very powerful network is the hypercube, which requires n connections per MCU when there are 2^n MCUs (e.g., in a hypercube with 64 MCUs, each MCU communicates with six other MCUs).

MCU LAYOUT & CIRCUITRY

The parallel computer consists of a number of MCUs. Each MCU can communicate with one MCU or more through a serial connection. (In the present configuration, each MCU can communicate with 12 other MCUs at most). You place the wires that form the connections between the MCUs.

All MCUs execute code from their internal flash memory. The MCUs are loaded with the same code (although this is not true for all the cases, as I already mentioned). A key feature for the implementation of the parallel machine is that all the MCUs have to execute the program in absolute synchronization with each other. This is achieved by using a common external clock for all the MCUs, and by forcing all MCUs to begin execution at the same time. This is something that is feasible with the PIC24FJ*xx* family.

The MCUs use their internal UART for the serial communication with each other. This inserts a great delay in the execution of the algorithm as compared to parallel communication, but the benefit is that it increases the connectivity between the MCUs, making possible the implementation of networks like the hypercube.

Figure 2 shows the main parts of the parallel computer. (The interconnection network is not shown because you form it.) The parallel machine consists of the MCUs, a clock generation and control circuit, and a "control" MCU that provides a serial interface with a PC. The clock generation and control circuit consist of a crystal oscillator plus a circuit that enables or disables the feeding of the MCUs with the oscillator's output. This is performed for two reasons. Firstly, all the MCUs must be completely synchronized in their code execution. This is achieved by using the reset (*MCLR) and clock pins of the MCUs. The procedure is as follows. All the MCUs go to the reset state (*MCLR = 0) and the clock is disabled. Next, the MCUs leave the reset state (*MCLR = 1). They still don't execute any code because there is no clock input. Then the clock is enabled and all MCUs execute their code in complete synchronization with each other. The second reason for using a clock enable/disable switch is to enable you to "freeze" the code execution at any time so the parallel computer may be used for educational purposes.

The PC interface uses another Microchip Technology PIC24FJ16GA002 microcontroller to take commands from a serial port and communicate with each MCU through their *MCLR, PGC, and PGD signals. The PC must run a special program that generates the user interface to the parallel machine. The *MCLR and PGC signals are common to all the MCUs. Each MCU is independently selected with the use of the separate PGD lines. The control microcontroller's main task is to detect the MCUs and load them with the appropriate program.

Figure 3 depicts the system's circuitry. Each MCU has a 12-pin connector (p1) for communicating with other MCUs using the UARTs. The peripheral pin select mechanism is

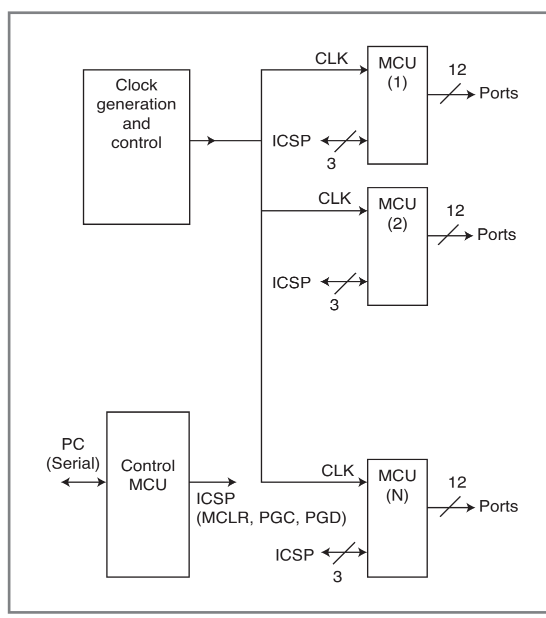


Figure 2—An overview of the parallel computer

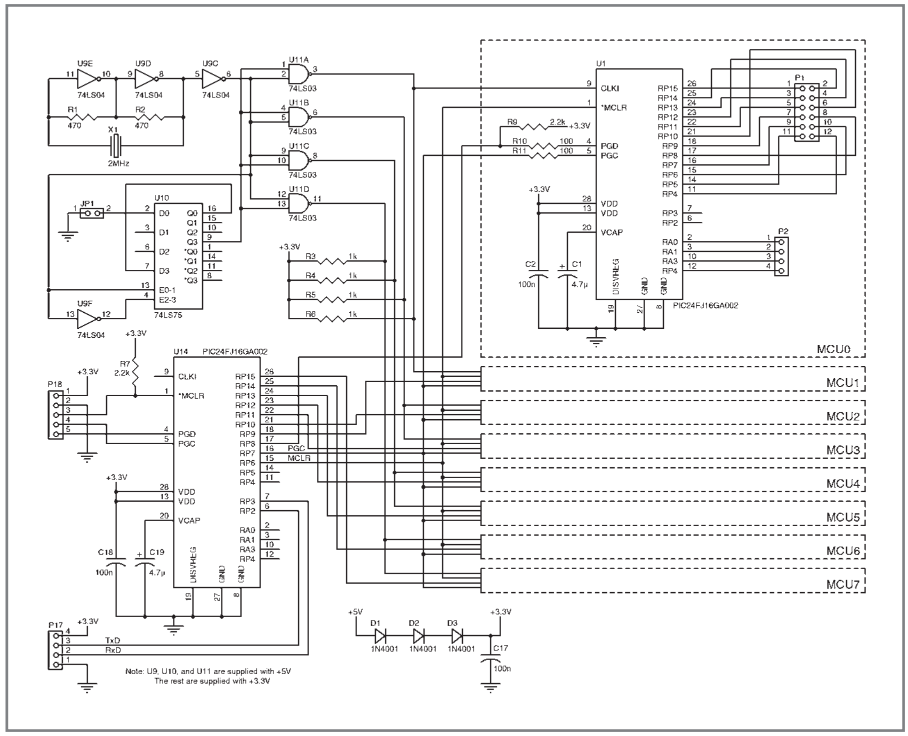


Figure 3—The parallel computer with eight MCUs

used to map each UART's transmitter and receiver to the desired pins. Each MCU has a four-pin connector (p2) for auxiliary input/output. Each MCU communicates with the control microcontroller via the *MCLR, PGC, and PGD pins. All MCUs share the same RESET and CLOCK signals.

The clock signal must be distributed with equal delay among all MCUs in order to have all MCUs completely synchronized.

A crystal oscillator is built around gates U9e, U9d, and U9c. Two of the latches inside U10 serve as a master-slave flip-flop that synchronizes the

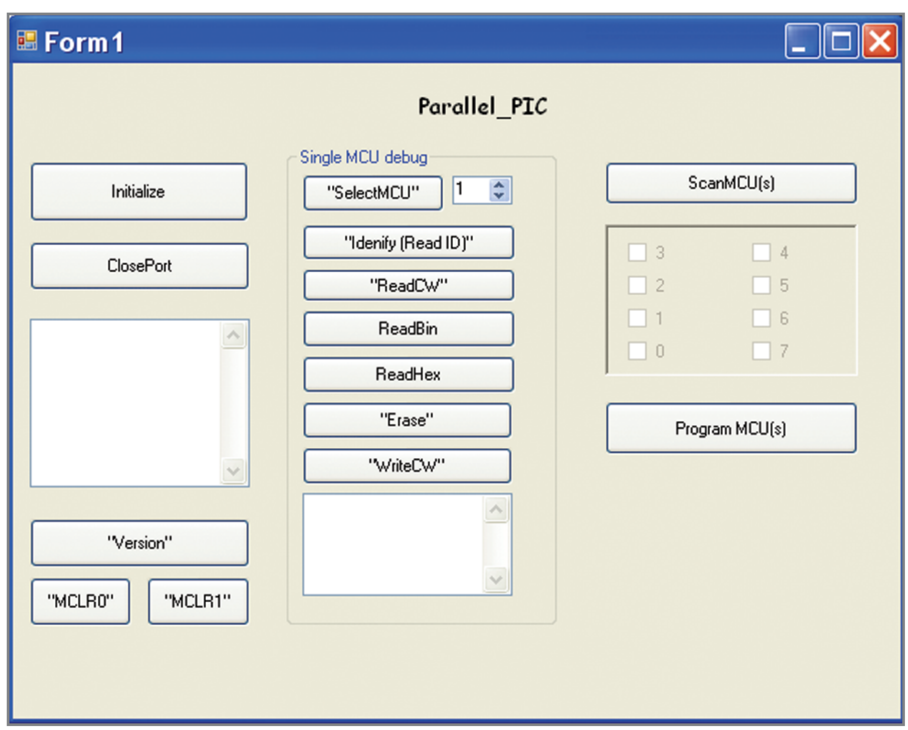


Photo 2—The user interface for the programming software

Listing 1—The code for the simple data transfer on a linear array

```
;Left

loop:
;============
; read value
MOV      PORTA,W1
AND      W1,W2,W1
;=============
; send value
MOV      W1,U1TXREG
;=============
; Wait
MOV      #100,W3
loopW:
NOP
DEC      W3,W3
BRA      NZ,loopW
;=============
; receive value
MOV      U1RXREG,W1
;=============
;
NOP
NOP
NOP
;=============
BRA loop

;Middle

loop:
;============
;
NOP
NOP
;=============
; send value
MOV      W1,U1TXREG
;=============
; Wait
MOV      #100,W3
loopW:

NOP
DEC      W3,W3
BRA      NZ,loopW
;=============
; receive value
MOV      U1RXREG,W1
;=============
;
NOP
NOP
NOP
;=============
BRA loop

;Right

loop:
;============
;
NOP
NOP
;=============
; send value
MOV      W1,U1TXREG
;=============
; Wait
MOV      #100,W3
loopW:
NOP
DEC      W3,W3
BRA      NZ,loopW
;=============
; receive value
MOV      U1RXREG,W1
;=============
; Write value
AND      W1,W2,W1
SL       W1,#3,W1
MOV      W1,PORTA
;=============
BRA loop
```

clock ENABLE/DISABLE signal with the crystal's output. The gates in U11 distribute the clock to the MCUs. U9, U10, and U11 are TTL and work with 5 V. The gates in U11 are open-collector to perform level translation for the 3-V MCUs.

The interface with the PC consists of the microcontroller U14. The microcontroller's serial port does not provide the correct levels for the RS-232 interface, so a level converter must be used. I don't show one here, but keep in mind that U14 is supplied with 3 V and not 5 V, so a MAX232 is not a good choice. The control microcontroller can be programmed "in circuit" through connector P17. Any in-circuit programmer can be used, so I don't recommend one in particular. The firmware that runs on the control microcontroller was written in assembly language and was assembled with MPLAB 8.10. The source code and the hex file are free for download and use.

The configuration words in all the MCUs must be set to disable JTAG (the respective pins are used for the MCUs' communication), select external clock, disable oscillator output, disable WDT, and enable IOLOCK to change.

The 3 V required for the MCUs is produced from the 5-V supply using three diodes in forward bias. The actual voltage may be less (around 2.8 V with full load), but it is still within the range permitted by the MCUs (2.2 V–3.6 V). All MCUs have their internal voltage regulator enabled. Photo 1 shows the parallel computer prototype with eight MCUs.

PC SOFTWARE

The programs that run on the parallel machine were written and assembled using the MPLAB IDE 8.10. I used assembly language to ensure the synchronization of the MCUs.

The code is downloaded to the MCUs using a special program that has been developed for this task. The software communicates with the Parallel_PIC machine through the RS-232 port (COM1). I developed the software with Microsoft's Visual Basic 2005 Express Edition (a free download). I suggest you create a new project and

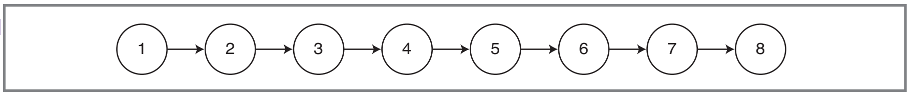


Figure 4—A linear array

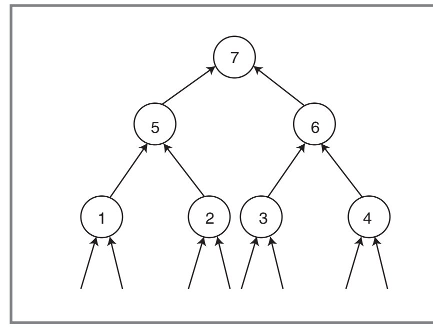


Figure 5—A binary tree with four leaves

replace the files "form1.vb" and "form1.designer.vb" with the files that you can download from my website (http://users.sch.gr/achorevas/ParallelPIC_PCcode_V2.zip).

Photo 2 shows the user interface. The Initialize button is used to start the program. (It is also done when the program is loaded.) The ClosePort button is used to end the communication through the serial port. (It is also done when the program is closed.) The Version button is used to send the respective command to Parallel_PIC. The result is displayed in the box over this button. The MLCR0 button is used to send the respective command to Parallel_PIC. All MCUs enter the reset state. The MLCR1 button is used to send the respective command to Parallel_PIC. All MCUs enter the normal operation state.

The buttons in the middle column are for controlling a single MCU. They are used mainly for debugging and problem detection on single MCUs, and not in normal operation. The Select MCU button is used in conjunction with the frame on its right to select the MCU that will be referred to by the other buttons in this column. It sends the respective command to the Parallel_PIC. The Identify (Read ID) button is used to send the respective command to Parallel_PIC. The result of this command is the ID number of the selected MCU, and is displayed in the frame at the bottom of this column. The ReadCW button is used to send the respective command to Parallel_PIC. The result of this command is the CodeWord of the selected MCU displayed in the frame at the bottom of this column. The ReadBin button is used to send the respective command to Parallel_PIC. The result of this command is to send the Parallel_PIC all the contents of the flash memory of the selected MCU. The data transmission is done in binary format. The data is then converted to ASCII hex by the PC and stored in the FlashDump.txt file.

The ReadHex button is used to send the respective command to Parallel_PIC. The result of this command is to send the Parallel_PIC all the contents of the flash memory of the selected MCU. This data transmission is done in ASCII format and the data is stored in the FlashDumpHex.txt file.

The Erase button is used to send the respective command to Parallel_PIC. The result of this command is to erase the content in the selected MCU's flash memory. The WriteCW button is used to send the respective command to Parallel_PIC. The result of this command is to read the two CodeWords of the selected MCU. (Note that the firmware does not let you define the value of the CWs. These values are preset from the firmware.) The ScanMCU(s) button is used to detect the MCUs and show what is available.

The check boxes are used to select the MCUs that will be programmed. The Program MCU(s) button is used to select the .hex file that will be loaded into all the selected MCUs.

FIRMWARE

The firmware that runs in the control MCU uses a serial connection to communicate with the PC. Most of the communication is performed in ASCII-formatted commands and responses. The control MCU executes an infinite loop where it waits for commands from the PC. When it receives a command, the MCU executes it and returns to the main loop. Although all

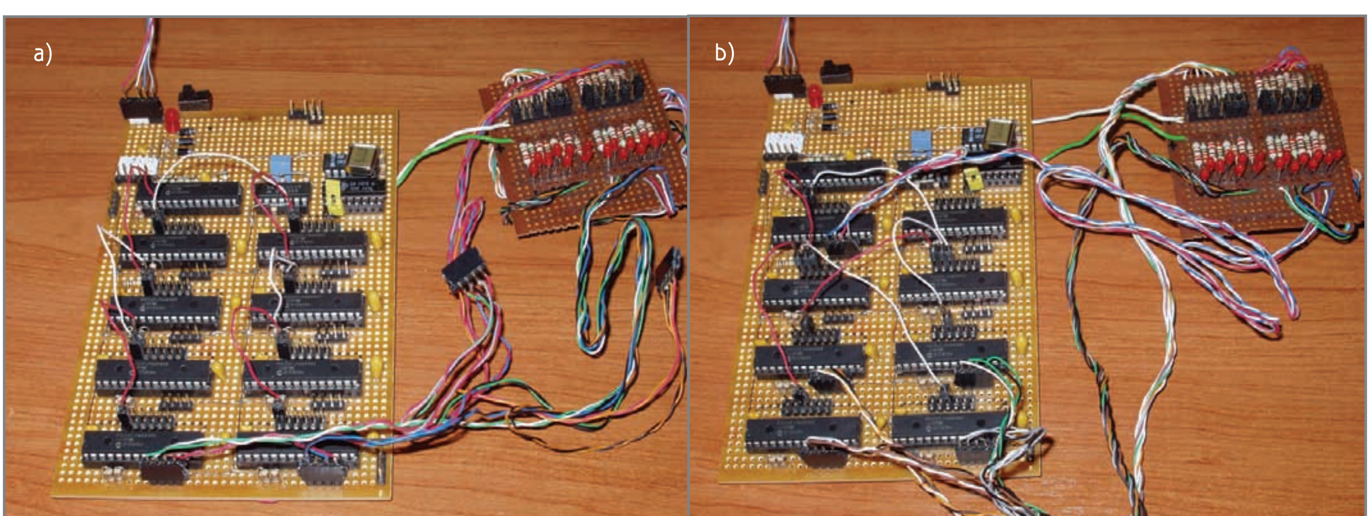


Photo 3a—This is the parallel computer configured as a linear array. b—Here you see the parallel computer configured as a tree.

the commands are formatted in ASCII, the data passed travels in binary format in some cases. As a result, you cannot always use a simple ASCII terminal program on the PC for debugging.

The commands that are implemented include MCU scanning, selecting one of the available MCUs, reading and writing the flash memory of the selected MCU, reading and writing the configuration words of the selected MCU, and erasing the flash memory of the selected MCU.

A small problem is encountered when using this software. The flash memory is completely erased in each program cycle. As a result, the configuration words have to be reprogrammed in each program cycle. So the configuration words are set to their initial state and then reprogrammed to use the external clock. This procedure can slightly confuse the MCUs. They have to run for a while with the external clock and then they operate as expected (which means the clock can freeze).

In the following sections, I'll present some parallel algorithms. The code that implements most of these algorithms is not optimized for speed as it has educational purpose. The last example illustrates a speed-optimized algorithm. All algorithms use a common initialization of the UART. The UARTs are configured to operate at their maximum baud rate ($F_{OSC}/8$). No interrupts are used. The input and output of the UART are mapped to the RP*x* pins depending on the specific algorithm and topology. This mapping can change during the algorithm execution.

TRANSFER TEST

Let's start with the linear array (a simple transfer test). The interconnection network for this algorithm is a linear array of one dimension. This means that all MCUs are considered to form a line. Each MCU communicates with the previous and the next MCU, except for the first MCU, which has no previous, and the last, which has no next. Figure 4 shows this topology. The arcs indicate that the MCUs send data to the right and receive data from the left. The problem is to transfer (or route) data from the leftmost MCU (1) to the rightmost MCU (8). The solution is as follows: Each MCU has a variable that holds the data to be propagated. Each one of the inner MCUs (2 to 7) transmits the value of its variable to its right MCU and receives a new value for the variable from its left MCU. The leftmost MCU reads a new value from PortA instead of reading it from another MCU. For practical reasons, this is a 2-bit value from PortA(0, 1). The rightmost MCU writes its variable's value to PortA(3, 4) instead of sending it to its right.

The code for the three aforementioned cases is in Listing 1. (Refer to the "Data_Transfer_Code.doc" file on the *Circuit Cellar* FTP site to see Listing 1 arranged in three columns. This arrangement will enable you to see the similarities and differences in the code that's executed in each MCU. You can also check the timing that's critical for correct communication, and you'll better understand the need for `NOP` commands.) The delay between transmission and reception operations is inserted to ensure the correct communication between the MCUs. The delay does not correspond to the exact number of cycles required, but it gives enough time for the UARTs to transmit and receive an 8-bit value. All MCUs have exactly the same initialization code.

In this algorithm, each MCU uses one UART. The receiver of the UART is mapped to pin RP14, while the transmitter is mapped to RP15. The interconnection network is formed by connecting the RP15 of MCU k to the RP14 of MCU k + 1. The NOP lines are inserted to keep synchronization of all MCUs. The first few pieces of data output at PortA of MCU 8 have no meaning, as they reflect the values of the variables inside the MCUs. The loop has to be executed seven times to output the correct input. Any change in the input data will be transferred to the output in seven loop executions.

A simple test to ensure the correct

operation of the clock "freeze" function is to stop the clock, change the input, and see that the output does not change. When the clock is reenabled, the output changes again.

Photo 3a shows the parallel computer (version 1) with this topology and a simple test board with jumpers for input and LEDs for output. The project is stored in the folder LinearArray_route. The source file LinearArray_route.s holds the assembly code for the left MCU. When the project is built, the hex file produced must be manually renamed. (This is already done and the resulting file is LinearArray_route_L.hex.) Then you must replace the code that is specific to the left MCU with the code that is specific for the middle MCUs (a simple copy-paste operation) and rebuild the project. The new hex file is renamed LineArray_route_M.hex. The procedure is repeated once more for the right MCU (file LinearArray_route_R.hex). Now the user can use the programmer to store the code to the MCUs. Note that the use of conditional assembling improves this procedure.

SUM OF SMALL INTEGERS

Now let's consider a tree algorithm (the sum of small integers). The interconnection network for this algorithm is a binary tree with four leaves. Figure 5 shows this topology. The arcs indicate the direction of the data transfers. MCUs 1, 2, 3, and 4 are called leaves. MCU 5 is the parent of MCUs 1 and 2, so MCUs 1 and 2 are the children of MCU 5. The same naming holds for the other MCUs. MCU 7 is also called the root of the tree.

The problem is to sum the data that are presented at the inputs of the leaves. The result is output from the root. The solution is as follows. Each leaf reads the data from PortA. For practical reasons, this is a 2-bit value from PortA(0, 1). It then transmits this value to its parent. Each of the middle MCUs reads the values from its left and right children. It then sums these two values and transmits the result to its parent. (This can be performed for many levels of the tree if the tree has more MCUs.) The root MCU performs all the operations of a middle MCU (except for the transmission) and also outputs the result to PortA(0, 1, 3, 4). Note that there are four numbers summed, each 2 bits, so the result requires 4 bits for correct representation. The code for these three cases (leaf, middle, and root) is in the Tree_Sum_Algorithm.doc file on the *Circuit Cellar* FTP site. The delay between transmission and reception operations was discussed for the previous algorithm. All MCUs have exactly the same initialization code except for the PortA initialization (the leaves require 2-bit input, while the root requires 4-bit output).

In this algorithm, each MCU uses one UART. The transmitter is mapped to RP11. The receiver is connected first to RP15 (in order to read the value from the left child) and then to RP13 (in order to read the value from the right child). This means the MCU at the left must transmit its value in the first step, and the MCU at the right must transmit its value in a second step. This is reflected in the L and R columns for the leaves and the middle MCUs. The difference in the L and R code is in the instance that the data are transmitted. The interconnection network is formed by connecting the RP11 of the L MCUs to the RP15 of its respective parent, and the RP11 of the R MCUs to the RP13 of its respective parent. The NOP lines are inserted to sustain the synchronization of all the MCUs. The first few pieces of data output at PortA of MCU 7 have no meaning, as they reflect the values of the variables inside the MCUs. The loop has to be executed three times to output the correct input. Any change in the input data will be transferred to the output in three loop executions.

The code can be written in a different way, where two UARTs are used for the reception of values. This way, the entire process is accelerated as the values can be received from both children at the same time and no pin-mapping reconfiguration is required. However, the dynamic pin remapping will be necessary for networks in which some MCUs have to transmit or receive data from three or more MCUs. So this is a good example of how this task can be accomplished. Photo 3b shows the parallel computer (the first version, V1.0) with this topology and a simple test board with jumpers for input and LEDs for output.

The project is stored in the Treesum folder. The Treesum.s source file holds the assembly code for MCU 1. When the project is built, the hex file produced must be manually renamed (as in the previous example). The resulting hex files are: Treesum_leaf_L.hex, Treesum_leaf_R.hex, Treesum_middle_L.hex, Treesum_middle_R.hex, and Treesum_root.hex.

FIR FILTER

The aforementioned algorithms are purely educational in nature. No one uses eight MCUs to perform sorting or summing! These "educational" algorithms show that the typical solution involves some steps of communication that alternate with steps of computation. In the previous examples, the communication steps were proven to be slow. (There is a "wait" function between each transmission and its respective reception.) This "idle" time can be used for computations.

The present example shows the design of an algorithm that can be efficient and practical. The code is neither complete nor tested yet. The "problem" is to perform the computations of a finite impulse response (FIR) filter. This is expressed with the following equation:

$$y = \sum_{i=0}^{N-1} x_{N-i} \times b_i$$

y is the output value. x_i are the input values. b_i are the filter's coefficients (or taps).

The interconnection network for this algorithm is a linear array of one dimension, as in the first example. The solution is as follows. Each MCU holds a set of coefficients and the respective input values. For example, each MCU can hold 64 coefficients, leading to a FIR filter with 512 taps (i.e., 8 × 64). Each MCU calculates the sum of products for the values it holds. Each MCU communicates with the MCU on its right to send the partial result and one of the input values. This way the input values "travel" through the MCUs. Each MCU adds the partial result received from its left MCU to its internal partial result before it transmits it to the MCU at its right. (The leftmost MCU assumes zero for input. The rightmost MCU does not have to transmit the input data value, but only the result that is now the final result.) The data values are 16 bits wide and the sums are 32 bits wide. The external clock is a multiple of the data input/output rate (an external PLL can be used).

The time between the transmission and the reception can be used for the calculations that do not depend on the received values. This way the communication overhead is minimized as it is restricted to the commands that read/write the UART registers. Of course, having 6 bytes for communication in each loop, there must be enough work for the MCUs to keep them busy during the communication phase. This means the parallel filter will be efficient only when the filter's length (i.e., the number of taps) is large. But this is the case that requires the use of a parallel machine!

PROBLEMS SOLVED

In this article I described the implementation of a parallel computer that uses a Microchip PIC24FJ16GA002-I/SP as the basic processing element. The microcontrollers use their UARTs to communicate. You must build an interconnection network between the microcontrollers. The peripheral pin select feature is used to dynamically map the UARTs to desired microcontroller pins. All the microcontrollers execute their code in complete synchronization.

As I explained, you can use the parallel machine for educational purposes, as well as for solving computation-intensive problems that can't be solved with a simple microcontroller. Developing the code is not a simple task because the parallel algorithms are more complicated than the usual algorithms. Some examples were presented to demonstrate the use of this parallel computer.

As you can see in Photo 1, the prototype consists of eight microcontrollers. However, the communication capabilities will enable you to construct much more complex machines. For example, you could build a hypercube of 4,096 (i.e., 2^{12}) elements. Downloading the code to the microcontrollers is performed with a PC that runs a special program that provides the GUI.

Author's note: Additional information about parallel processing is on my website http://users.sch.gr/achorevas/Parallel_theoryev.htm.

Antonios Chorevas (achorevas@sch.gr) holds an MSc in Electronic Automation and a PhD in Parallel Computers and Signal Processing from the University of Athens in Greece. He now teaches Informatics at the high school level. Antonios's interests include microcontroller-based designs, DSP, robotics, assembly programming, and FPGAs.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/247.

RESOURCES

P. Chaudhuri, "Parallel Algorithms: Design and Analysis," Prentice Hall, Englewood Cliffs, NJ, 1992.

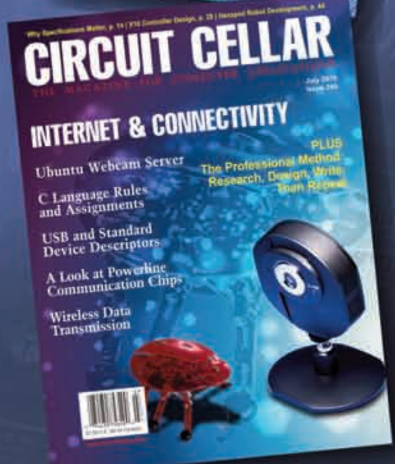
F. Thomson Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, San Francisco, CA, 1991.

SOURCE

PIC24FJ16GA002 Microcontroller
Microchip Technology, Inc. | www.microchip.com


The 4-to-20-mA Current Loop, p. 28 | Inside a Pro's Electronics Lab, p. 30 | MIPS Chips, p. 66
CIRCUIT CELLAR
THE MAGAZINE FOR COMPUTER APPLICATIONS
EMBEDDED DEVELOPMENT
August 2010
Issue 241
Embedded Safety and Failure Mode Analysis
How to Measure a Crystal's Properties
Advanced Features of USB Explained
An X10 Controller Program and Utility
Control How Apps Exchange Data
PLUS
How One Designer Uses Electronics & Embedded Technologies to Solve Real-World Problems ... on a Daily Basis


CIRCUIT CELLAR
THE MAGAZINE FOR COMPUTER APPLICATIONS
COMMUNICATIONS
WWVB Receiver DSP
OAE Probe Amp & Otoacoustic Experiments
The USB Enumeration Process Explained
Understanding DC/DC Converters
Custom Interpreter Development


CIRCUIT CELLAR
THE MAGAZINE FOR COMPUTER APPLICATIONS
INTERNET & CONNECTIVITY
Ubuntu Webcam Server
C Language Rules and Assignments
USB and Standard Device Descriptors
A Look at Powerline Communication Chips
Wireless Data Transmission
PLUS
The Professional Method: Research, Design, Write, Then Repeat

by Carlo Tauraso (Italy)

Wireless Access and Control

A "MiWi" Network Card Reader

Need an inexpensive wireless access control system? This article presents the MiWi protocol and details how to design a network card reader.

The production of wireless devices has grown steadily in recent years. In particular, so-called wireless personal area networks (WPANs) are quickly gaining relevance. These are networks with a range of a few meters used for different devices placed near a user. The most widely used technologies for this purpose are Bluetooth, HomeRF, and ZigBee. A few years ago, Microchip Technology released the "MiWi" protocol, which is based on the IEEE 802.15.4 standard and enables you to realize networks of devices that can communicate within the 2.4-GHz frequency band. Soon after I learned about the protocol and started considering the design possibilities, I built a MiWi network card reader (see Photo 1). In this article, I'll describe my design and software.

My MiWi network card reader can communicate with the data card independently via a node (coordinator), which will pass data directly to a PC via USB. I developed software that records the data received, notes the date and time, and decodes bitstreams using two common standards: ANSI 5 bit and ANSI 7 bit. This makes it possible to understand the different features of a WPAN network as well design something useful like a custom wireless access control system.

Using the same circuit diagram as the coordinator, I developed a second set of firmware that turns it into a 2.4-GHz spectrum analyzer. The device scans the entire ISM band and sends the level of RF spectral energy on each channel to a PC via a USB port. I also developed a simple software monitor to graphically display the band and its variations. Thus, the same circuit can be transformed into a useful tool to identify possible sources of interference within the 2.4-GHz frequency band.

MiWi NETWORK

There are basically two types of devices in MiWi networks: a PAN coordinator and an end device. In this project, I used a star network configuration consisting of one PAN coordinator node (connected to a PC via USB) and one or more end devices (with a card reader onboard). All end devices communicate only with the PAN coordinator that starts the network, selects a PAN ID (16-bit address) that uniquely identifies the network, and selects one of the 11 ISM 2.4-GHz communication channels. The end device searches the ISM band for a specific PAN ID, and then it joins the network and starts to transmit data packets. Every node (coordinator and end device) has a unique 8-byte address called an extended organizationally unique identifier.

In this design, the coordinator has the following EUI: 00-15-19-01-19-21-01-20. The first end device has the following EUI: 11-15-00-15-12-18-01-03. If you want to add other end devices, you can simply modify the EUI variable in the firmware source code (MIWIDefs.h file). The MiWi protocol transports data packets by using reports. There are different report types, and there are different report IDs for each report type. Report type 00h is reserved for the MiWi protocol stack packets; all other report types are available for the user. For this design, I used a user report type 12h, with report ID 34h and a payload of 64 bytes (which is sufficient to transmit all 400 bits recorded into the magnetic stripe card).

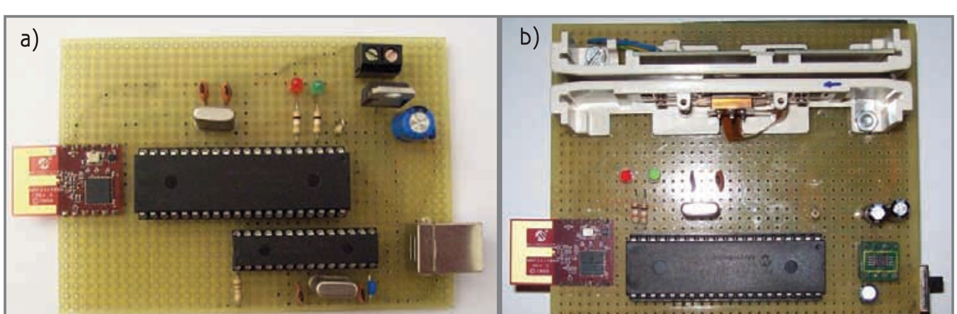


Photo 1—The final wireless card reader prototype. a—The RF module with the PCB antenna is on the left. b—The magnetic swipe card reader is at the top.

A MiWi network is a multi-access network, so all nodes share the same medium of communication. In this design, all nodes are allowed to transmit at any time as long as the channel is idle. This is the so-called non-beacon-enabled network. The IEEE 802.15.4 defines three frequency bands with a fixed number of channels. I chose the 2.4-GHz frequency band with 11 channels available and a maximum data throughput of 250 kbps. It is oversized compared to what is necessary for my application, but I can use it as the basis for future development. Figure 1 depicts a real application based on my design.

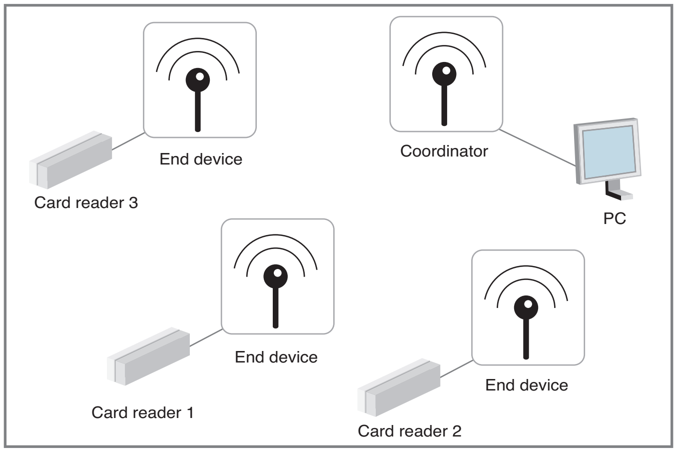


Figure 1—This is the layout of a real application built around my design. Three card readers are implemented.

THE COORDINATOR

The design is based on a Microchip Technology MRF24J40MA RF transceiver module. It works within the 2.405-to-2.480-GHz ISM band and communicates with a simple four-wire SPI (see Figure 2). It has a PCB antenna fabricated on the copper trace with an optimal radiation pattern and an operating range of about 400′. The coordinator design has two sections: an RF section and a USB section. The former features an MRF24J40MA with a Microchip Technology PIC18LF4620 (see Figure 3). The USB section features a PIC18F2550. I used the MiWi Wireless Networking Protocol Stack to manage all the MRF24J40MA's functionality. It's available for free on Microchip's website.

To transfer the data received from the RF module to the PC's USB port, I used an internal parallel bus that connects the PIC18F4620's PORTD with the PIC18F2550's PORTB, respectively. In the PIC18F2550, TIMER0 is configured as an 8-bit counter with external clock source. Its value increments on low-to-high transitions on the clock line (RA4). When a data packet comes from the MiWi network, the PIC18LF4620 saves all 64 bytes into a buffer. Then it reads and sends them sequentially to PORTD. For each byte, the PIC18LF4630 makes a low-to-high transition on clock line RA0 connected to the PIC18F2550 RA4 pin. TIMER0 increments and the PIC18F2550 reads PORTD and inserts the byte value in a buffer. After the sixty-fourth transition, the buffer is full, so the PIC18F2550 sends a 64-byte packet to the PC via USB. Here, the software records and decodes the bitstream from the swiped card. Figure 4 details the communication interface between the PIC18LF4620 and the PIC18F2550.

Many USB applications will likely have several different sets of power configurations. In this project, I use the "Bus Power Only Mode" so all the power (5 V) is drawn via USB. I used two small capacitors (C1 and C2) for filtering. The PIC18LF4620 and the RF module need a 3.3-V power source, so I inserted an LM1086-3.3v regulator from STMicroelectronics (www.st.com). A 20-MHz crystal is used as a reference to generate the system clock. For full speed operation, the clock source must be 48 MHz. The PIC18F2550 device includes a phase-locked loop (PLL) multiplier circuit designed to produce a fixed 96-MHz reference clock from a fixed 4-MHz input. I use the prescaler (PLLDIV2..PLLDIV0) to divide the 20-MHz primary frequency by five. The resulting signal (4 MHz) goes to the PLL and, finally, the 96-MHz output is divided by two (CPUDIV1..CPUDIV0), generating the required clock for MCU core and USB module. The C3 and C4 capacitors are chosen according to the Microchip datasheet.

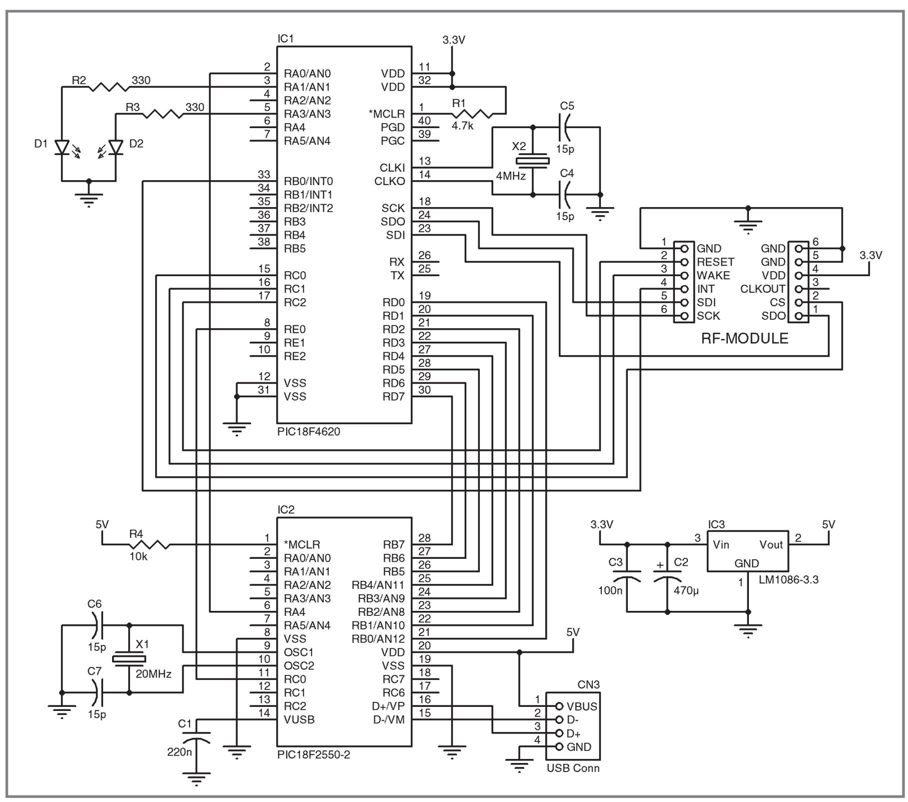


Figure 2—The coordinator has an RF section and a USB section. The design features a PIC18LF4620 and a PIC18F2550.

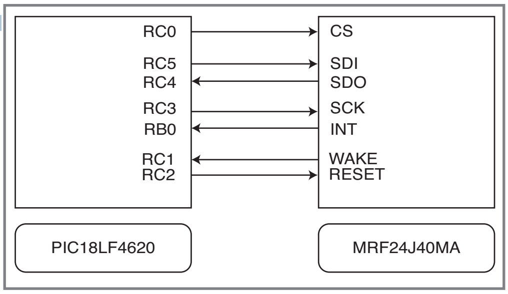


Figure 3—This is the communication interface between the PIC18LF4620 and the MRF24J40MA.

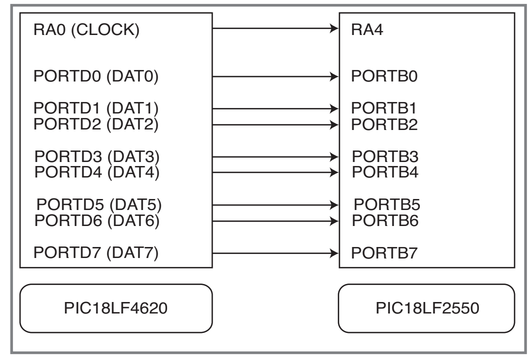


Figure 4—Here is communication interface between the PIC18LF4620 and the PIC18F2550.

D1 and D2 are two colored LEDs used to communicate the circuit state to the user. When D2 (green) is on, the device has successfully formed a MiWi network with PANID 0A0Bh. D1 (red) indicates a packet received from the network. R2 and R3 limit the current flowing through D1 and D2. RC5 and RC4 are directly connected to USB data lines. There are two fundamental bus speeds: full (pull-up on D+) and low (pull-up on D–). I use an internal pull-up on D+ to specify full-speed mode so it is not necessary to insert another discrete component.

The construction is simple. All components fit in a 100 mm × 70 mm prototype board. There are no critical parts. You can find them in any electronic shop. Use 3-mm LEDs, ceramic capacitors, and 0.25-W resistors to minimize the dimensions. You have to be careful with the USB connector. An error can damage your PC's USB port. I used a type-B USB connector so you can connect it with a canonical A-B USB cable.

You have to be careful with the RF module, too. Use a low-power soldering iron and mount it on the edge of the host PCB. It is recommended that an area of 3 cm around the antenna be kept clear of metal objects. Photo 1a shows the final prototype. On the left side is the RF module with the PCB antenna.

The heart of the circuit is a PIC18LF4620 with a 3.3-V power supply. The voltage choice prevents the need for signal level conversion between the microcontroller and RF module, and allows it to be powered by a battery (see Figure 5). Unlike my previous project, this time I used an OMRON V3B-4K magnetic card reader. It's an affordable (less than $30) reader with a TTL-compatible interface and a maximum current use of 5 mA.

My reader has an interesting feature that differentiates it from other designs: it allows for different configurations of the read head. You can move it to read all three tracks of the card, not just the usual second track. If you look at the read head side, you'll see that there are already three pairs of holes on the plastic support. If you want to read Track 1, for example, you simply move the read head in position number 5, as shown in Photo 2. Likewise, if you want to read Track 3, you move the head in position number 3.

The reader uses the same TTL interface as many others. There are three communication lines: card loading signal (CLS), read clock signal (RCP), and read data signal (RDP). CLS, RCP, and RDP are normally high. You swipe the card and the CLS goes low. While CLS and RCP are low, the PIC18LF4620 inserts a bit value set to 1 when RDT is equal to 0. On the other side, it inserts a bit value set to 0 when RDT is equal to 1.

The V3B-4K has a nine-pin connector. (You can review the detailed pinout in the OMRON's datasheet.) The PIC18LF4620 has 3.3-V logic, but the input pins are compatible with TTL signals. I used three Schottky diodes as a precaution, and I enabled the internal pull-ups on PORTB. Therefore, when the card reader sends a high logic level (5 V), the diode does not conduct and the pull-up supplies the voltage (3.3 V) input to the PIC. When the card reader sends a logic level low, the diode conducts, pulling the PIC input line to ground. The end device power section consists of four NiMH batteries with a total voltage of 4.8 V and a capacity of 2,000 mAh. The batteries are connected

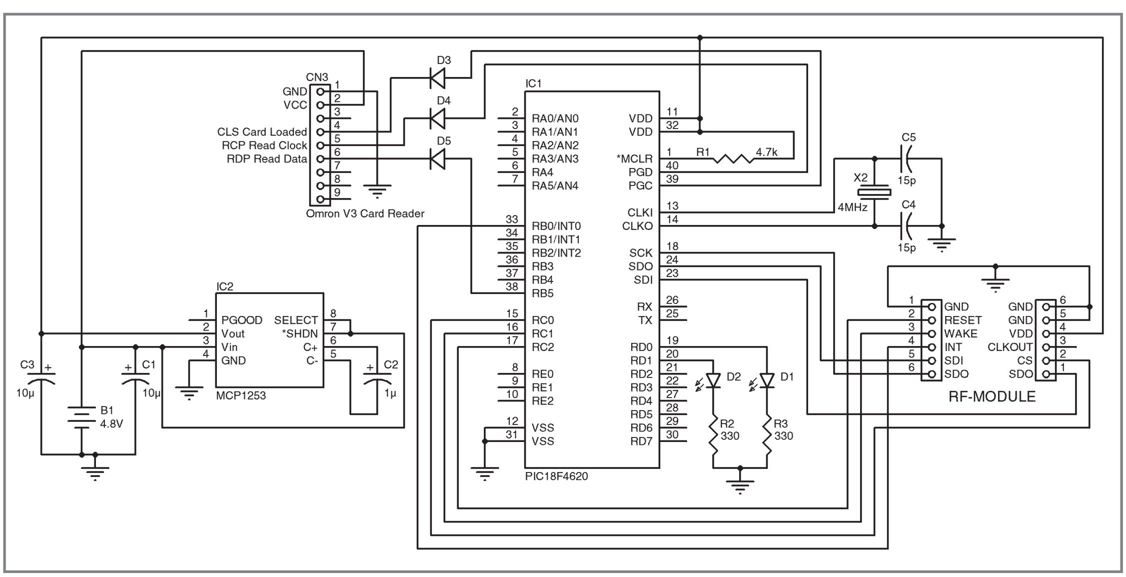


Figure 5—The end product includes a PIC18LF4620 (IC1), an MCP1253 (IC2), and an MRF24J40MA (IC3). I also included an OMRON card reader.

to a Microchip Technology MCP1253 inductorless, positive-regulated, charge pump DC/DC converter. The device allows the input voltage to be lower or higher than the output voltage by automatically switching between buck and boost operation. In my design, it generates a regulated 3.3-V fixed voltage. For the charge pump, I used a 1-µF electrolytic capacitor with optimal results. To filter the input and output, I added two 10-µF capacitors. I disabled the shutdown function connecting the SHDN pin to VIN. The chip package is an MSOP-8, so I had to use a converter for the assembly. It is very easy to find on the market a specific MSOP-8-to-8-pin DIP adapter (www.cimarrontechnology.com). The circuit maximum current use is about 50 mA; therefore, using four NiMH batteries (2,000 mAh), the unit can handle 36 hours of continuous transmission at room temperature.

THE FIRMWARE

I developed three different forms of firmware for this project. Firmware A,

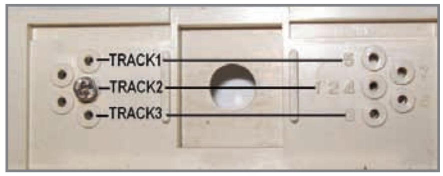


Photo 2—Black lines indicate the different read head positions. Simply move the head to read the other tracks.

for the PIC18LF4620, manages the coordinator and the communication between PIC and the RF module. Firmware B, for the PIC18F2550, manages the communication between the coordinator and the PC's USB port. Firmware C is used to manage the card reader and the communication between the end device and the coordinator. For development, I used the C18 language and the MiWi wireless networking protocol stack. In the 2010 Microchip Technology application note titled "MiWi Wireless Networking Protocol Stack," you'll find a lot of information about the protocol and a description of the main functions for managing a MiWi network.

Firmware A is loaded into the PIC18LF4620 on the coordinator board. Its processing flow is divided into three main phases. The first is a general board initialization. It is important to activate the SPI module inside the PIC necessary to communicate with the MRF24J40MA. In the second stage, the PIC forms the MiWi network with PAN-ID value equal to 0A0Bh and waits for packets from different end devices. In the third stage, when the PIC receives a packet containing a complete user report, it transfers the 64 bytes to PORTD. From here they then move to the second microcontroller PIC18F2550. The transfer is achieved by sending a rectangular pulse on pin RA0 for every byte sent to PORTD. If the reception is partial or the user report does not contain a known identifier, it is simply ignored by deleting the input buffer. In Listing 1, you can see the `switch` statement that performs the data extraction from an input buffer called `pRxData`, including the received packet type identification

Listing 1—This code performs data extraction and type identification.

```
switch(*pRxData++)                //report type
    {
    case USER_REPORT_TYPE:
      switch(*pRxData++)        //report id
      {
        case CARD_REPORT:
           {
           if (RxSize >= 64) //Have you received the whole packet or not?
              {
                 for (i=0; i<64; i++) //while (RxSize--)
                 {
                    PORTD = *pRxData; //send value to USB section
                       PUL = 1; //Pulse to USB
                       for (x=0;x<=0x1FF;x++); //WAIT
                       PUL = 0; //Pulse to USB
                    *pRxData++;
                 }
                    }
           LED_R = 0;
           DiscardPacket();
           }
           break;
        }
```

and the 64-byte sequence transmission. After that, the PIC calls the `DiscardPacket()` function, which frees the input buffer so it can receive the next packet, avoiding a buffer overflow. Stack MiWi allocates two buffers of only 128 bytes each: one for input packets and one for output packets (see Listing 1). The MIWIdefs.h file is available on the *Circuit Cellar* FTP site.

Firmware B is loaded into the PIC18F2550 chip on the coordinator board. It monitors an internal counter (Timer0), which is incremented each time a pulse appears on pin RA4. For each pulse, the micro saves the byte on PORTB in a buffer called `xbuffer`. Every 64 bytes, it transmits the complete packet to the PC via USB bus.

Firmware C is loaded into the PIC18LF4620 chip on the end-device board. Even in this case, the processing flow consists of three phases. The first one initializes the board and the internal SPI module needed to communicate with RF module. During the second phase, the micro performs a band scan recording the MiWi network PAN-IDs within the end-device operating range. Then it reads the table sequentially and, when it finds a PAN-ID equal to 0A0Bh, it joins that network. If it doesn't find the right network, it rescans the band until the search is successful. The third phase will start only if the network with a PAN-ID equal to 0A0Bh was found and the joining operation is going to succeed.

During the third stage, the firmware waits for the CLS line to go low. This happens every time a user swipes a card. Then, the microcontroller records the bitstream, encapsulates it into an appropriate user report, and transmits it via RF.

SOFTWARE

I developed the "MiWi Card Reader Datalogger" program in C# for the .NET Framework 2.0 (see Photo 3). It enables you to record the card data received from the coordinator. The user interface is intuitive. First, connect the coordinator board to a free USB port. The operating system will recognize it as a human interface device (HID). You don't need to install any drivers because all Windows operating systems (from Win98 SE on) have an HID driver inside. The green LED will light up indicating that the MiWi network is active. Double-click the icon and run the program. If everything is well, you'll see the message "Mi-Wi Coordinator connected, swipe the card." Using the combo box `Code Table`, you can select the encoding standard used to record card data, choosing between two possible items: ANSI 5 bit and ANSI 7 bit. Then you should initialize

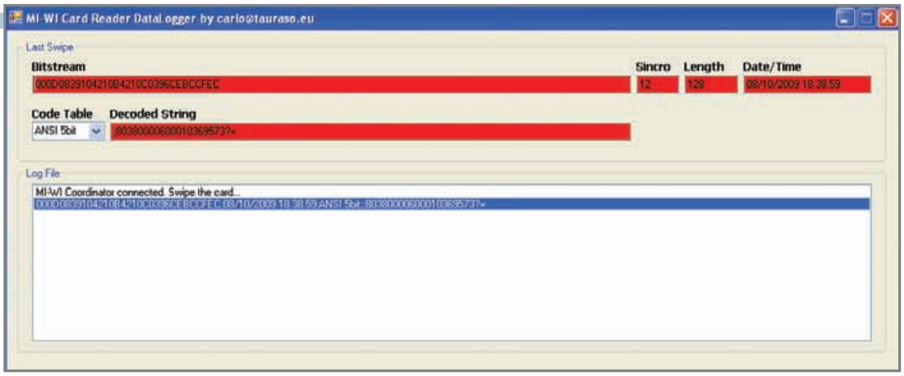


Photo 3—Here you see the sequence number read from my European sanitary card.

the end device, so put switch 1 in the On position. The lighting of the red LED will indicate that the node is searching the network with a PAN-ID equal to 0A0Bh. As soon as it finds the network, the red LED turns off and the green LED turns on. Now you can swipe the card. During this operation, the red LED will illuminate to indicate that the CLS is low. Receipt of the data packet is indicated by a brief flash of the red LED on the coordinator board. At this point, you will see in the form fields all the information recorded on the card's magnetic stripe. In the `decoded string` field, you can see the bitstream decoded according to the standard chosen.

The most important fields, separated by a semicolon, will appear in the list box named `Log File`. At any time, you can save the sequence of records from the list box into a text file. Just right-click and you will get a pop-up menu with the Save Log item from which you can start saving.

SPECTRUM ANALYZER

By replacing firmware A with firmware A+ and firmware B with firmware B+, you can transform the coordinator node into a very useful scanner for monitoring the presence of possible interference sources into the ISM band. After replacing the firmware, reconnect the coordinator node to a USB port. Even in this case, the device is recognized as a HID. Double-click the icon and run the ISM Spectrum Analyzer (see Photo 4). It is a .NET 2.0 project, so it is ready for Windows Vista or Windows 7. With this software, you can monitor the ISM band usage in real time. Click the icon. When you are ready to start monitoring, click the Start button (the button label changes to Stop). During monitoring, all controls are disabled; you can only stop the recording process by clicking the Stop button again. The RSSI values of each channel will be updated in real time.

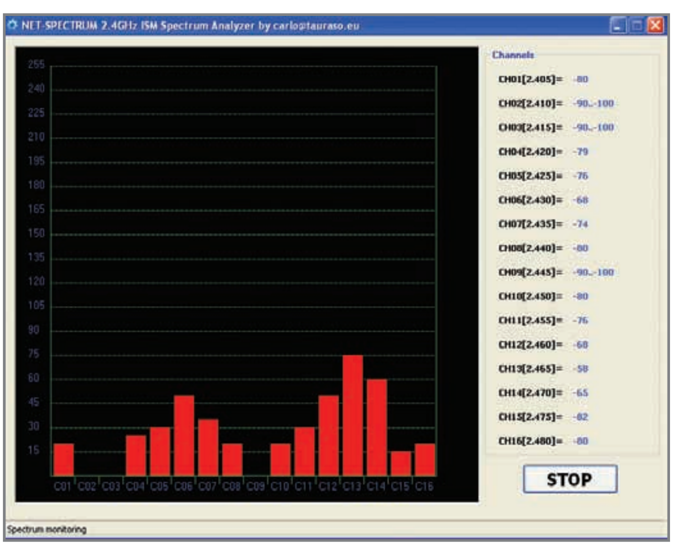


Photo 4—A spectrum-monitoring example. On the right side is the received power (dBm).

GO WIRELESS

My design is more than an inexpensive wireless access control system. You can use it as a basic design to understand how MiWi networks work. Just unplug the card reader from the end device board to achieve something different, such as a network for temperature monitoring. The RF operating range is sufficient to cover your home, and the transfer rate is optimal for most uses, especially for monitoring signals with slow variation and a low transfer rate. I hope you'll find this circuit useful for your next wireless communication project.

Carlo Tauraso (carlotauraso@gmail.com) wrote his first assembly code in the 1980s for the Sinclair Research ZX Spectrum. Today he's a senior software engineer who performs firmware development work on network devices and various kinds of microinterfaces for a range of European companies. Several of Carlo's articles and programming courses have been published in Italy, France, Spain, and the United States.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/247.

RESOURCES

D. Flowers and Y. Yang, "Microchip MiWi Wireless Networking Protocol Stack," AN1066, Microchip Technology, 2010.

OMRON, "Manual Swipe Magnetic Card Reader," V3A, 2002, www.coronaprint.com/datasheet/V3A_e.pdf.

SOURCES

MCP1253 Converter, MiWi wireless networking protocol stack, MRF24J40MA transceiver module, PIC18F2550, and PIC18LF4620
Microchip Technology, Inc. | www.microchip.com

V3B-4K Magnetic card reader
OMRON Industrial Automation | www.omron247.com

by David Penrose (USA)

Remote Humidity Control System

This innovative humidity control system calculates water vapor pressure from temperature and humidity readings. When it detects that the outside air is drier than the air indoors, it triggers a ventilation system. You can monitor and control the system via an Internet-connected PC.

My humidity control system switches between a fresh air vent and a dehumidifier to maintain low humidity in my basement with minimal use of electricity (see Photo 1). When the air is drier outside than in the basement, the system pulls in outdoor air. When outdoor moisture is greater than it is in the basement, the system shuts down the ventilation and starts a dehumidifier. While performing this balancing act, the design records the data to a USB drive and enables remote observation and control via Ethernet. All this is accomplished with a WIZnet W7100 microprocessor.

CONTROL CONCEPT

My unfinished basement is the hub of my design activities and contains a lot of equipment and documents. Unfortunately, it is below ground level and quite a bit of moisture seeps through the concrete walls and floor. This moisture, if left unchecked, can rust metal parts and turn my documents moldy. To control this moisture, I run a dehumidifier. When the outside air is drier than the basement air, I shut off the dehumidifier and open up the windows. This saves a bit on electricity since the dehumidifier consumes about 600 W and runs about half the time to maintain 60% humidity. Although the process works, a computer can easily make the decision on when to ventilate and when to use the dehumidifier. Plus, computer control can save me the work and also provide 24-hour monitoring. This was the basic idea for my design, a "green solution to basement humidity control."

Relative humidity cannot be used directly to compare the amount of moisture in two areas. The temperature of the two areas also must be considered. Since my basement is generally cooler than the outdoors, outdoor air brought in will be chilled by the basement and its relative humidity will increase (cooler air cannot hold as much moisture as warm air). To decide when to ventilate, the system needs to calculate the water vapor pressure (WVP) of both areas. WVP can be derived from temperature and humidity with a simple table and some math. These two measurements of WVP then can be used to decide when to ventilate and when to use the dehumidifier.

I wanted to be able to remotely control and monitor the device without having to go into the basement. The W7100, with its embedded Ethernet capability, provided the basics for a server system that can be accessed by a client browser. Since the system collects temperature and humidity data, it seemed a waste not to store it for later analysis. I decided to include a USB flash drive to record this information together with the time and date. The final implementation also included two additional temperature/humidity sensors to monitor the inlet and outlet of my HVAC system. This data was useful for monitoring the performance of my new geothermal system. Having decided on these basic requirements, I arrived at the system illustrated in Figure 1. It was time to begin implementing it.

HARDWARE

The WIZnet evaluation board was a perfect development and target platform for my basement system since it integrated the processor, display interface, Ethernet connectivity, serial port, and status LEDs on a PC board with space for my additions. I had done quite a bit of 8051 programming in the past, but I'd never attempted any TCP/IP communications, so this also became a great learning exercise.

I decided to use a Future Technology Devices International VDrive2 for the interface to a USB flash disk. This module provides a very simple interface for

Photo 1—The complete basement humidity control system is assembled on a table. The heart of the system is the small wooden case in the center, which contains the iMCU evaluation board and associated electronics. The humidity/temperature sensors are on the left. The control box and vent control devices are on the right.

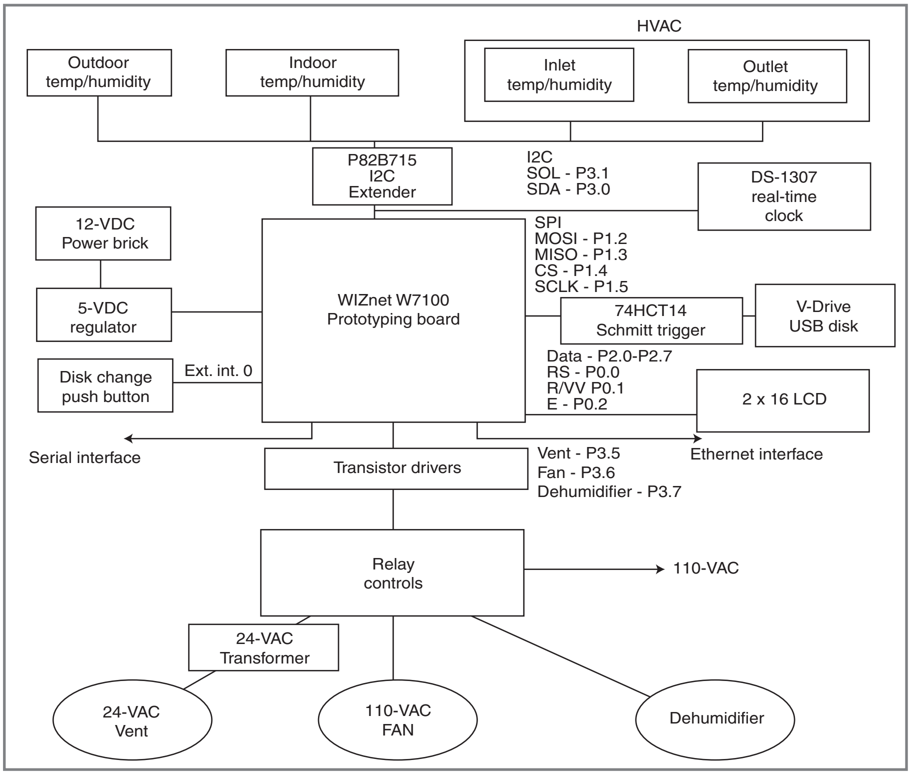


Figure 1—The iMCU7100 evaluation board provided most of the key components for the project integrated on one circuit board. The board also allowed room for the additional electronic components illustrated in this block diagram. The regulator, relays, and vent components are external to the board.

reading and writing a flash drive without having to keep track of low-level FAT details. It also allows for two different logical interfaces to a microprocessor based on a jumper pin setting. The first interface is a standard serial communication channel, while the second is a SPI. Since I did not want to tie up the W7100 serial interface (I use it for debugging), I elected to use the SPI option. I'm not sure if the serial interface would have caused me similar problems, but getting the VDrive2 to work with the W7100 was one of the most challenging problems I faced.

I had used the VDrive2 in other projects and had a library of code that worked quite well. The two areas I knew would take some work were making sure I had the timing right and verifying the W7100 would drive the VDrive2 pins correctly. The first challenge was solved with a little work using test code and an oscilloscope. The second challenge proved to be one of those swamps that took considerable time to drain. Even with correct timing and proven code, I couldn't get the VDrive2 to talk correctly. It would respond to initialization strings, but its responses were nonsense.

The W7100 uses internal pull-up resistors to generate high levels on its port pin, so my first attempt to get the drive to work was to add parallel resistors to speed up the low-to-high transitions. This did not help. I next added series resistors to try to eliminate what I believed to be noise on the data lines. This also did not help. Finally, I put together some diagnostic software and, together with the oscilloscope, trapped what appeared to be noise on the data line from the VDrive2 when the SCLK line was changing from low to high.

At that point, I decided to swallow my pride and add an external chip to the prototyping board to clean up the clock edges with a Schmitt trigger (74HCT14). The interface sprung to life. I'm not sure if it was just my module that was sensitive to the clock edges, but this is something to keep in mind as you work with interfaces from the W7100 to other devices.

To complete the hardware for the VDrive2 interface, I added a push button switch to signal the processor that the flash drive will be removed. The software checks this switch, and when it's pushed, the software will suspend operations and close any open files. When the switch is pressed again or when the software senses a flash drive being inserted, it will begin processing again.

The next addition to the prototype board was a real-time clock. I chose the DS1307 with a small back-up battery. The interface to the DS1307 is I^{2}C. Normally, these signal lines can be generated by a microprocessor by setting the port pins to an open-collector configuration and then adding some pull-up resistors. The W7100 does not have configurable port pins, but it implements all pins as open-collector with an effective 75-kΩ pull-up. I added some 22-kΩ pull-ups and the interface worked fine.

My temperature and humidity sensors

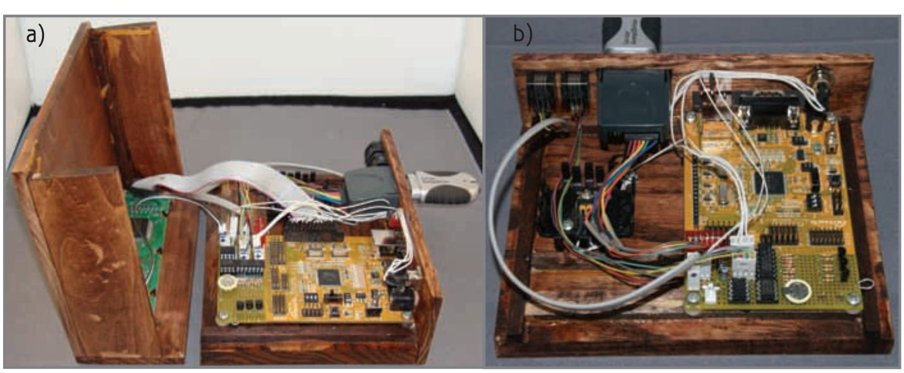


Photo 2a—This view of the open project case shows the components mounted to the evaluation board, the regulator, and the VDrive2, which is mounted to the back of the case. The "disk change" switch is in the center of the case and the power jack is on the right. b—This side view of the open project case shows the LCD connected to the evaluation board and mounted to the front of the case. The case was designed to slip together, which comes in handy during debugging activities.

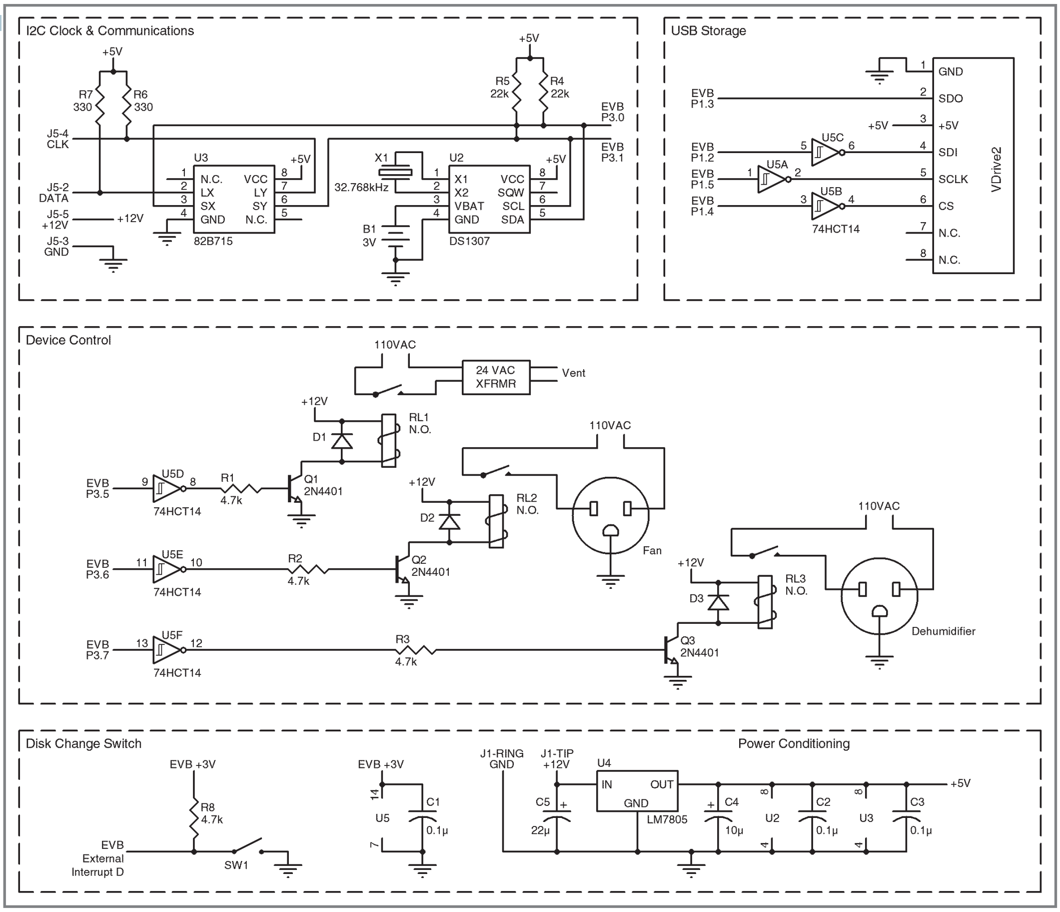


Figure 2—The components added to the evaluation board can be grouped into four sets. The first set is the I²C for the clock and sensor communication. The second is the VDrive2 USB disk interface and the 74HCT14 to clean up the clock lines. The third group consists of the fan, vent, and dehumidifier controller. Finally, the 12-V power brick and regulator complete the system.

are home-built units with an I²C interface, so I added an 82B715 interface chip to drive the long lines to these remote sensors. I added 330-Ω resistors to the data and clock lines on this interface chip to sink enough current for the long distances involved. The W7100 does not have a native I²C logic interface, but this was easily generated using bit-banging in the software.

The remote temperature and humidity sensors use a microprocessor, a DS18B20 1-Wire temperature sensor, and a capacitance-cell humidity sensor. The microprocessor includes the I²C logic and behaves as a slave unit with a unique address. A simpler solution would be to use the Sensiron integrated temperature/humidity sensor (SHT11) sold by Parallax and dedicate individual port pins for each data line. These sensors are quite accurate and easy to use, but I have also managed to destroy a handful of them without knowing how I did it. Use extra caution if you decide to use them. The interface for the temperature and humidity readings in the software is isolated, so it is easy to integrate whatever sensor you select.

The fresh air control for my system consists of a 24-VAC duct damper like the ones used in home air conditioning ducts. I added a 110-VAC duct fan. To control these devices and the dehumidifier, I built a relay board that fit inside a standard junction box, added a 24-VAC transformer to the outside, and equipped it with a dual 110-V socket. Three small neon lamps indicate when the sockets and transformer are active. Three transistors located on the prototyping board control the relays. Since I had extra inverter gates left over, I drove the transistors through them to further isolate any switching noise from the W7100.

The relays I used were rated at 10 A, which was more than enough for the fan and the transformer, but the dehumidifier control was marginal with its 5-A running current and 9-A starting current. My concern was that switching the unit on when the compressor was engaged could exceed the 10-A capability. Fortunately, the dehumidifier is designed to start only the fan when power is supplied and then later engage the compressor, so the relay should not see any significant switching current.

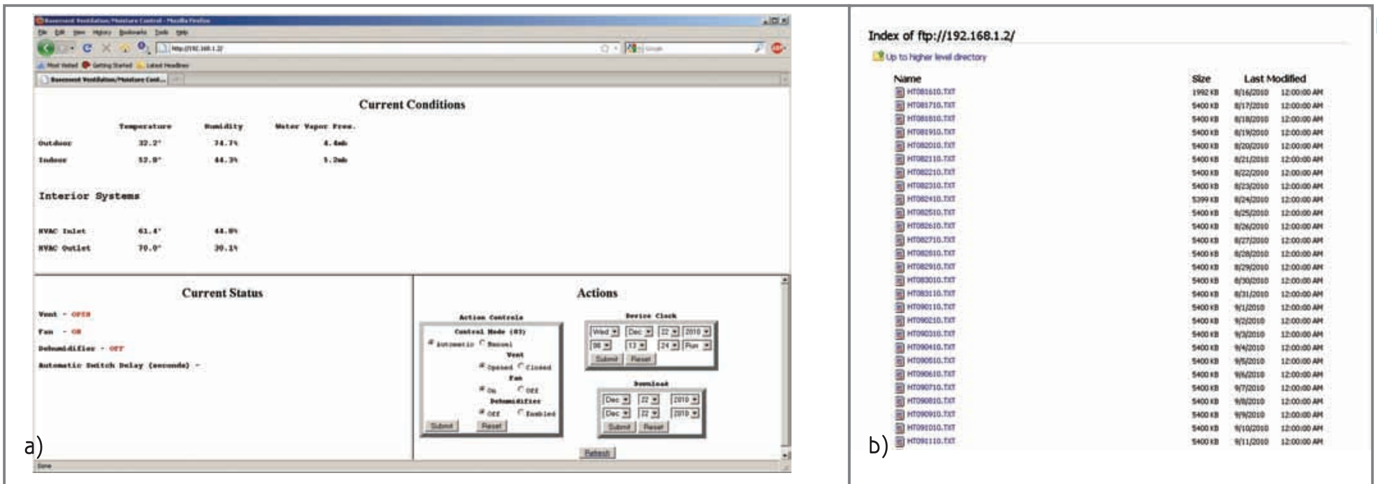
The evaluation board comes with an alphanumeric LCD, but I wanted to use one I had with larger characters. I also wanted to mount this display in my case rather than on the evaluation board, so I built a connector that translated the pin positions on the connector to the LCD cable. The connector and the display are shown in Photo 2a. I added a current-limited 12-V line to the LCD backlight, and it was all I needed for my local display.

A supplied 5-V transformer powers the evaluation board. I needed 12 V for the backlight and the remote relays, so I added a 5-V regulator to my project case for the evaluation board and used a 12-V power brick.

Figure 2 depicts the complete hardware layout. Photo 1 shows the system's elements assembled on a table top. Now for the software and checkout process.

SOFTWARE

When is a hobby not really a hobby? Psychologists theorize that "we are who we were when." This sounds confusing, but it means that we take a "set" just like a nickel-cadmium battery. If for years a person has been programming in assembly language, then he'll probably fall back into this comfort zone when offered a new challenge. That's me. I've been an assembly language programmer for many years, and it just seems natural to resort to these tools for all new



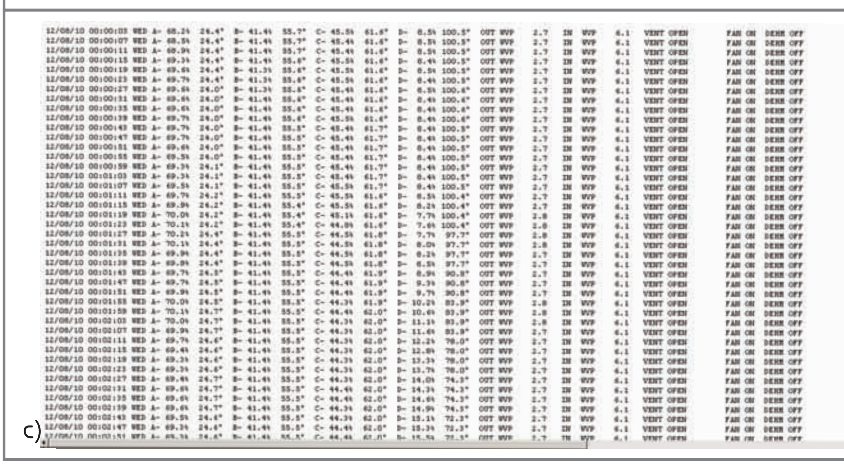


Photo 3a—The three frames presented by the iMUCW7100 server display the current conditions, system status, and control actions. The first two are updated at 10-s intervals, while the third is manually refreshed. All changes to the W7100 and requests for file transfers are accomplished by the third, or action, frame. **b**—The FTP directory is presented when the server is addressed as an FTP server. It shows a list of available data files and their sizes, which should be the same unless there have been power interruptions. **c**—The data stored on the USB drive captures the measured data and system-generated actions. There is considerable space at the end of each of the fixed-length frames to add more data if required in the future.

projects. A while ago, I dug out an old reference text and was carefully paging through it so the time-worn pages would not fall out. My wife was watching this and finally offered the observation that I no longer had "hobbies" but instead had "eccentricities." I thought about this a while and had to agree with her. Nevertheless, I enjoy my eccentricities and this project went forward, implemented in one of my eccentricities.

The first thing I did when I received my W7100 evaluation board was to begin translating the C code for the TCP example to assembly language. I used my own conventions rather than the assembly language generated by the C compiler. This resulted in hitting a number of interesting speed bumps during the process. One of the first involved the use of the WIZCOPY function in the W7100. I had naively assumed that the function was implemented without impacting any of the basic registers of the W7100. I use R7 for most all of my loops and hence was surprised when I finally traced one of my bugs to the WIZCOPY function, also using R7. I guess a compiler is smart enough not to leave anything important in volatile registers when it calls a function, but I rarely use any function that I haven't generated, so this was a valuable lesson learned. The Wireshark network protocol analyzer software is an invaluable tool during this debugging process. This software captures packets based on user filters and then interprets the packets to decode Internet traffic. I could not have completed my project without it.

Once I had the TCP example working, I began adding the HTML to produce the webpages. This is where the W7100's large memory really came in handy. I used data statements to "can" most of the dialog and then used program logic to fill in the dynamic elements, such as the status of the vent, fan, and dehumidifier. I placed the canned data in a library called WIZ_HTML. The basic display the server supplies to the client is divided into three frames as shown in Photo 3a.

The first frame, Conditions, occupies the top half of the screen and supplies information on temperature, humidity, and water vapor pressure. The HTML statements for this display are labeled `HTML_Cx`, where `x` ranges from one to 11 (see Listing 1). In between each of the statements, the program outputs the dynamic information for the display. The `Refresh` command in `HTML_C11` requests the client refresh the page every 10 seconds and hence a new request will be sent by the client to the W7100 server for this frame.

The second frame, Status, is allocated to the left half of the lower half of the screen. It is produced similarly to the Conditions frame.

The interesting frame is the third frame in the lower right of the client display. Called the Action frame, it enables you to interact with the W7100 to control devices, set the clock, and retrieve files. This frame is not automatically refreshed because you might be interacting with it when a refresh occurs. A manual Refresh button is included instead. Most of the display generation for this frame is accomplished in code rather than data statements.

Listing 1—`HTML_Cx` contains all the data statements that generate the Conditions display. Each of the individual data segments is output, followed by dynamic data generated by the code. The last segment specifies the refresh request of 10 seconds.

```
HTML_C1:
    DB       '<HTML><HEAD><TITLE>Current Conditions</TITLE></HEAD>'
    DB       '<BODY><CENTER><H2>Current Conditions</H2></CENTER>'
    DB       '<PRE><TT><H3>  Temperature    Humidity  Water Vapor Pres.</H3>'
    DB       '<H3>Outdoor                                    '
HTML_C1_LEN  EQU  $-HTML_C1
HTML_C2:
         DB       '&deg; '
HTML_C2_LEN  EQU  $-HTML_C2
HTML_C3:
         DB       '%                             '
HTML_C3_LEN  EQU  $-HTML_C3
HTML_C4:
         DB       'mb</H3>'
         DB       '<H3>Indoor                         '
HTML_C4_LEN  EQU  $-HTML_C4
HTML_C5:
         DB       '&deg;                        '
HTML_C5_LEN  EQU  $-HTML_C5
HTML_C6:
         DB       '%                            '
HTML_C6_LEN   EQU  $-HTML_C6
HTML_C7:
         DB       'mb</H3>'
         DB       '<BR><H2>Interior Systems</H2>'
         DB       '<BR><H3>HVAC Inlet                '
HTML_C7_LEN  EQU  $-HTML_C7
HTML_C8:
         DB       '&deg;                  '
HTML_C8_LEN  EQU  $-HTML_C8
HTML_C9:
         DB       '%                             </H3>'
         DB       '<H3>HVAC Outlet        '
HTML_C9_LEN  EQU  $-HTML_C9
HTML_C10:
         DB       '&deg;                        '
HTML_C10_LEN  EQU  $-HTML_C10
HTML_C11:
         DB       '%                            </H3></TT></PRE>'
         DB       '<META HTTP-EQUIV="Refresh" CONTENT="10">'
         DB       '</BODY></HTML>',CR,LF,CR,LF
HTML_C11_LEN  EQU  $-HTML_C11
```

FTP

The TCP process was easy to implement since WIZnet had supplied prototypes for this logic. Using the file transfer protocol (FTP) was not as easy or straightforward. The FTP process involves two asynchronous pipelines between the client and the server. The first is the control pipeline, which is similar to what I already described. The twist in the logic is the data pipeline that FTP uses. The control pipeline is used to define which type of transfer will occur and what data will be involved. The data pipeline actually transfers the data. This transfer can take place with ports defined by the client or the server. If the server is allowed to define the port numbers, this is called a passive transfer and uses the PASV verb in the dialog. If the client defines the port number, then the PORT verb is used and the process becomes more difficult.

Most FTP transfers use the passive approach, which avoids problems with firewalls. I elected to implement this approach. Some browsers, such as Internet Explorer, may default to the PORT protocol and will need to be modified to use the passive approach. This is easily accomplished through an advanced option in the Tools menu. Other browsers, such as Firefox and Opera, default to the passive mode and seem to work very predictably. Photo 3b shows the FTP directory returned to the client by the iMCU server, while Photo 3c shows the format of the data files delivered to the client and stored on the USB drive.

LOCAL DISPLAYS

The 2 × 16 display is used to show the same information that is available over the Ethernet link, only in a much abbreviated and time-sequenced fashion. If you look closely at Photo 1, you can see one of the displays, which shows the status of the controlled equipment. Photo 4 shows another of the four different displays presented by the unit, which are changed at 5-s intervals. This display indicates outdoor ("A") and indoor ("B") humidity and temperature. I use letters to indicate outdoor and indoor readings since 16 characters is not enough to include any more descriptive information.

A third display presents humidity and temperature of the inlet and exhaust of the heater/air conditioning unit, while the final display shows water vapor pressure for the outdoors and basement. Although not shown, each time the system makes a change to the equipment, it starts a 15-minute countdown that prevents further changes until the timer expires. This prevents unnecessary changes as conditions are stabilizing. The local display shown in Photo 1 will present this timer on line one.

During power-up, the display will indicate the state of the various initialization routines, but this information is quickly replaced with the time-sequenced displays. If the system hangs during initialization, the display will be left indicating the process that is having difficulties. If you press the "disk change" button on the back of the unit, the display will change to indicate it is waiting for the disk to be changed and processing will be suspended.

PACKAGING

I built a project case for the humidity control system that comfortably handles all of the devices. Photo 2b shows all the connectors I added to the evaluation board. During the debugging process, the evaluation board made many trips in and out of the case. Using connectors allows it to be easily removed and replaced. I tried to keep the connectors' pinouts fairly similar, since it is guaranteed they will be swapped accidentally at some point. Keeping power and ground consistent prevents extreme disasters, although not all problems will be averted. Caution is called for when this many similar connectors are involved. Also, be sure to make reliable connectors or you will spend considerable time chasing broken connector wires rather than solving real problems.

I use RJ-11 connectors for both the remote relay box and the remote humidity/temperature sensors. Again, the potential exists for getting the wrong connector in the wrong jack, but the pinouts should prevent major disasters.

The remote relay box is switching 110 V and hence has the potential for being dangerous if not properly wired (see Photo 1). The box is metal. The green wire from the 110-V plug is grounded to this case. The ground from the plug is also attached to this grounding point, as is the ground line from the 24-V transformer. Use caution when wiring the hot and neutral wires. Keep in mind that the black wires are hot and the white wires are neutral. Also, maintain a good separation from the 12-V relay control lines and these switched 110-V lines.

I used a small circuit board to fix everything in place. If everything is wired correctly, the hot lines will attach to the narrow slot on the plugs and the neutral lines will attach to the wider slots. You can check this with a plug-in circuit checker that you can buy at a hardware store. I highly recommend you do so. Don't risk placing 110 V where it could come into contact with a user.

INTEGRATION

If you examine the project code on the *Circuit Cellar* FTP site, you'll notice a few `IF` statements that enable or disable various sections of the code. The `IF` statements that use the condition `TEST_CODE` enable logic that eliminates the temperature and humidity sensors and instead steps through all valid ranges for temperature and humidity. This enables the calculation and action routines to be verified. It takes a while to loop through all combinations, but the results are recorded on the VDrive2 and can be examined offline for correctness. While the loop is running, you can observe the actions on the fan, vent, and dehumidifier.

The second useful debug tool is the set of `V_DRIVE` statements. These disable the VDrive2 interface so all other logic can be checked without the USB drive. I found this necessary as I struggled with the slow clock line to this interface.

NEXT PHASES

The next step for the project is to replace the hard-coded address of 192.168.1.2 with a DHCP option and allow the unit to get its address automatically. I also plan to look at making the device accessible from outside my house so I can check on the unit remotely when I'm away. There is plenty of processing power left in the W7100, so I expect this list of expansions will grow over time.

Photo 4—This view of the local display shows the abbreviations that were used to fit information to the 16-character lines of the display. In this case, "A" refers to outdoor measurements and "B" to indoor measurements.

In March and September 2010, the *New York Times* ran articles about the "Internet of Things." Authors such as Ashlee Vance and Nick Bolton described how the Internet is expanding by connections to smart phones, TVs, and other consumer devices. This new environment is the "Internet of Things." My green solution to basement humidity control is now one small piece of this growing universe of Things. It has proven to be a very useful Thing. ■

Author's note: I credit my wife, Melinda, for putting up with strange devices appearing throughout the house and also editing my rough thoughts into finished articles.

Dave Penrose (david.penrose@comcast.net) is retired from the aerospace industry. He began his career programming test software for space vehicles and has maintained a lifelong passion for developing microprocessor projects. He holds an Expert Class Amateur Radio License (K1DHP) and has an MS in Computer Science from the University of Santa Clara and a BA in Mathematics from the University of California at Berkeley.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/247.

RESOURCE

D. Tweed, "iMCU W7100 Embedded Networking Made Simple," *Circuit Cellar* 233, 2009.

SOURCES

VDrive2 Interface module
Future Technology Devices International Ltd. | www.ftdichip.com

DS1307 RTC Chip and DS18B20 sensor
Maxim Integrated Products | www.maxim-ic.com

HS1101 Humidity sensor
Measurement Specialties, Inc. | www.meas-spec.com

89LPC925 Processor in temperature/humidity sensor
NXP Semiconductors | www.nxp.com

SHT11 Humidity/temperature sensor
Sensiron | www.sensiron.com

24-VAC Air duct damper (#307107C) and in-line booster fan (#3011)
Smarthome | www.smarthome.com

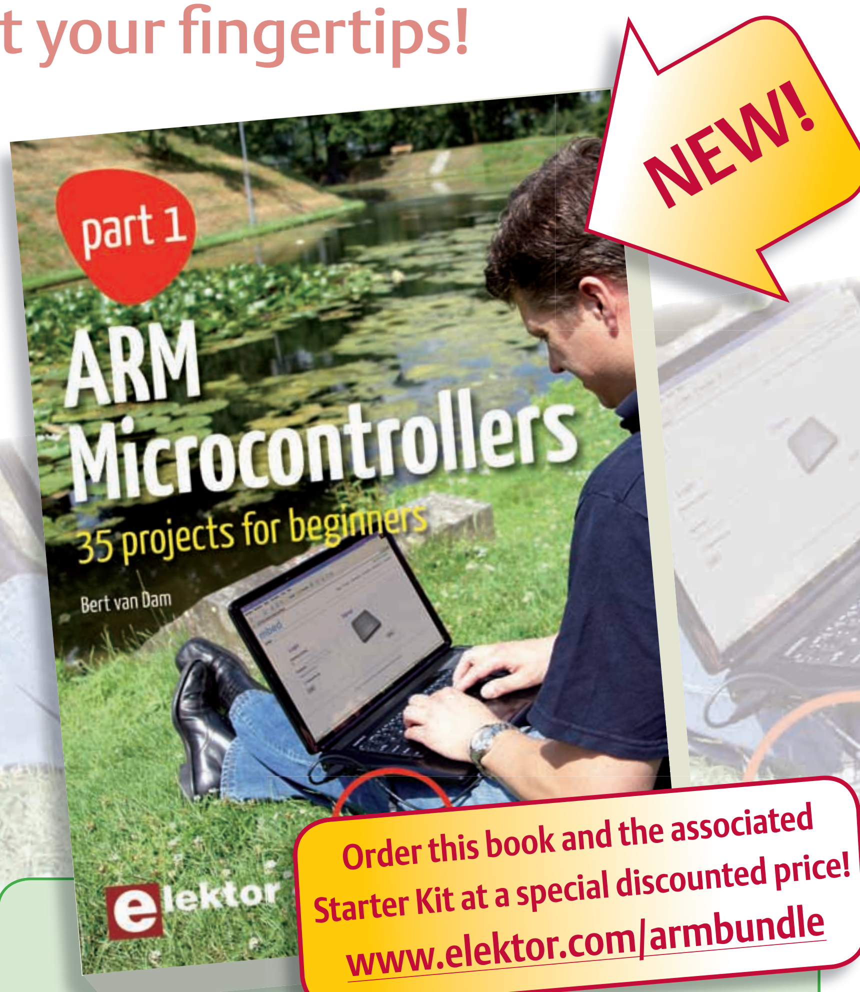
82B715 I²C Bus extender
Texas Instruments | www.ti.com

Network Protocol Analyzer Software
Wireshark Foundation | www.wireshark.org

iMCU7100 Evaluation board
WIZnet | www.wiznet.co.kr

Elektor Shop

The world of electronics at your fingertips!



Order this book and the associated Starter Kit at a special discounted price!
www.elektor.com/armbundle

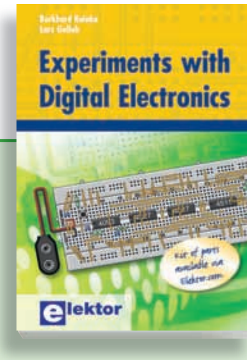
Books

Associated starter kit available

ARM Microcontrollers

This is the perfect book for people who want to learn C and who want to use an mbed ARM microcontroller in an easy and fun way. The mbed NXP LPC1768 uses cloud technology, a revolutionary concept in software development. This means you do not need to install software on your PC in order to program the mbed! The only thing you need is a browser such as Microsoft Internet Explorer, and a USB port on your PC. No previous experience or knowledge required. You can get access to your project from any PC anywhere in the world and continue working on it. When you are done a few simple mouse clicks transfer the program to your mbed hardware. Examples of the projects in this book: flashing light, timer, light activated switch, digital thermometer, people detector, USB communication, talking microcontroller, debugging, sound switch, and much more. Also order the associated starter kit (Art.# 109022-71) and get started straight away with the projects described in the book!

256 pages • ISBN 978-0-905705-94-1 • $47.60

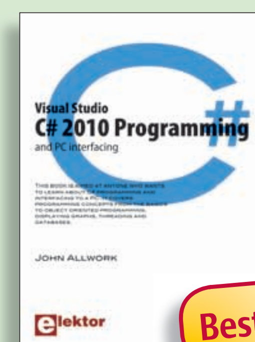


An introduction to digital control electronics

Experiments with Digital Electronics

This book presents fundamental circuits using gates, flip-flops and counters from the CMOS 4000 Series. Learning these fundamentals is best done using practical experiments. Each of the 50 experiments presented in this book has a circuit diagram as well as a detailed illustration of the circuit's construction on solderless breadboard. Building these digital circuits will improve your knowledge and will be fun to boot. Many of the circuits have practical real-life applications.

176 pages • ISBN 978-0-905705-97-2 • $42.80



Bestseller!

Visual Studio

C# 2010 Programming and PC interfacing

This book is aimed at anyone who wants to learn about C# programming and interfacing to a PC. It covers programming concepts from the basics to object oriented programming, displaying graphs, threading and databases. The book is complete with many full program examples, self assessment exercises and links to supporting videos. All code examples used are available – free of charge – from a special support website. Professional quality software tools are downloadable – also free of charge – from Microsoft.

305 pages • ISBN 978-0-905705-95-8 • $47.60

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

CD/DVD-ROMs

More than 75 power supply designs

CD The Power Supply Collection 1

This CD-ROM contains more than 75 different power supply circuits from the volumes 2001-2005 of Elektor. Highlights include the Cuk Converter, Automatic Battery Switchover, Battery Voltage LED, Digital Benchtop Power Supply, Lithium-Ion Charger and much more. Using the included Adobe Reader you are able to browse the articles on your computer, as well as print texts, circuit diagrams and PCB layouts.

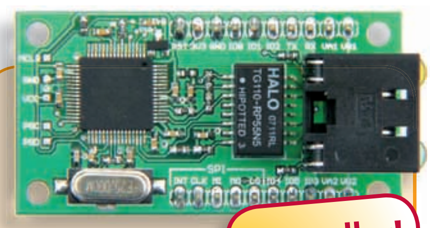
ISBN 978-90-5381-265-5 • $28.90

See the light on Solid State Lighting

DVD LED Toolbox

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

ISBN 978-90-5381-245-7 • $46.00



Bestseller!

NetWorker

An Internet connection would be a valuable addition to many projects, but often designers are put off by the complexities involved. The 'NetWorker', which consists of a small printed circuit board, a free software library and a ready-to-use microcontroller-based web server, solves these problems and allows beginners to add Internet connectivity to their projects. More experienced users will benefit from features such as SPI communications, power over Ethernet (PoE) and more.

Module, ready assembled and tested

Art.# 100552-91 • $85.50

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

www.elektor.com/shop

elektor

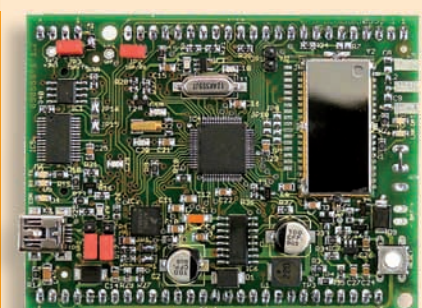
Elektor US
PO Box 180
Vernon, CT 06066
USA
Phone: 860-875-2199
Fax: 860-871-0411
E-mail: sales@elektor.com

Digital Multi-Effects Unit

It's a simple fact: every recording sounds better with the right sound effects. Here we prove that it's possible to generate a variety of effects digitally, including hall, chorus and flanger effects, without having to work yourself to the bone with DSP programming. The circuit is built around a highly integrated effects chip and features an intelligent user interface with an LCD. The result is a treat for the eye and the ear.

Kit of parts including PCBs, programmed controllers and EEPROM

Art.# 090835-71 • $266.20



Reign with the Sceptre

This open-source & open-hardware project aims to be more than just a little board with a big microcontroller and a few useful peripherals — it seeks to be a 32-bit ARM7 fast prototyping system. To justify this title, in addition to a very useful little board, we also need user-friendly development tools and libraries that allow fast implementation of the board's peripherals. Ambitious? Maybe, but nothing should deter you from becoming Master of Embedded Systems Universe with the help of the Elektor Sceptre.

PCB, populated and tested, test software loaded

Art.# 090559-91 • $143.60

Kits & Modules

by Robert Lacoste (France)

Don't Fade Away !

A Multipath Fading Experiment

Multipath fading is a nasty phenomenon that can ruin a short-range radio application. This article explains the topic and details some techniques you can use to protect your RF systems.

It's 6:59 AM. You finish your coffee, jump into your car, switch on the radio to get the morning news, and leave. As you start driving on a straight urban road, the radio is working perfectly. But soon thereafter, you hear the following from the radio host: "And yesterday evening, very surprisingly, the winner was..." Nothing. Your radio no longer transmits anything except noise! But then, a few moments later, after traveling a couple of meters, the radio starts working again. And now it's too late.

Sound familiar? If so, you've likely experienced what RF engineers call multipath fading.

The nasty phenomenon of multipath fading jeopardizes a lot of short-range radio projects. As you will see, a system advertised with a 200-m range may not work in some circumstances even at a 5 m. Worse, it may happen without any visible obstacles in the radio's path.

In this article, I'll explain the phenomenon of multipath fading so you'll be able to describe it to an angry customer or boss. Because an explanation may not be enough, I'll also describe actual tests I completed in a "typical" indoor environment, as well as some design alternatives that could make your RF system far more reliable.

RADIO THEORY

Let's start with the basics. In open field—that is, with two antennas facing each other without any obstacles or walls around—the loss of a radio-frequency link is well approximated by the Friis formula:

$$\text{Free space loss (dB)} = 20\log_{10}(d) + 20\log_{10}(f) - 147.55$$

In this formula, d is the distance between the transmitter and the receiver, expressed in meters. f is the working frequency in hertz. For example, if you make the calculation for a 200-m link with a carrier frequency of 2.4 GHz, you'll find a loss of about 86 dB. That means that if you transmit 100 mW, you will receive 100 mW × $10^{-86/10}$, which makes 244 pW, assuming that the antennas have no gain. This formula also implies that the signal attenuation should increase by $20\log_{10}(2) = 6$ dB each time the distance is doubled.

Unfortunately, a lot of radio links are now used in indoor or dense urban environments; and in these situations, the relationship between distance and signal attenuation is much more complex due to two main problems. The first appears when there is an obstacle on the transmission path—that is, on the straight line between the transmitter and receiver (or, more exactly, too close to this line). This causes the so-called slow-fading phenomenon: the signal level goes up and down whether or not obstacles are present. So, if you are walking, the typical distance between each minimum is linked to the size of the obstacles (e.g., buildings, trees, and bodies), so more or less independent of the frequency. The fading appears and disappears quite slowly, which is why we use the term "slow fading."

The second problem, fast fading, is a little more difficult to understand. It is due to the multiple copies of the same signal that are received on the antenna due to reflections on walls, ceilings, floors, furniture, and so on (see Figure 1). The receiver antenna has no way to "separate" these different signals and in fact it adds them all and sends their sum to the electronics. All these signals will have the same carrier frequency but different phases. In some situations, a reflected signal will be out of phase with the direct path

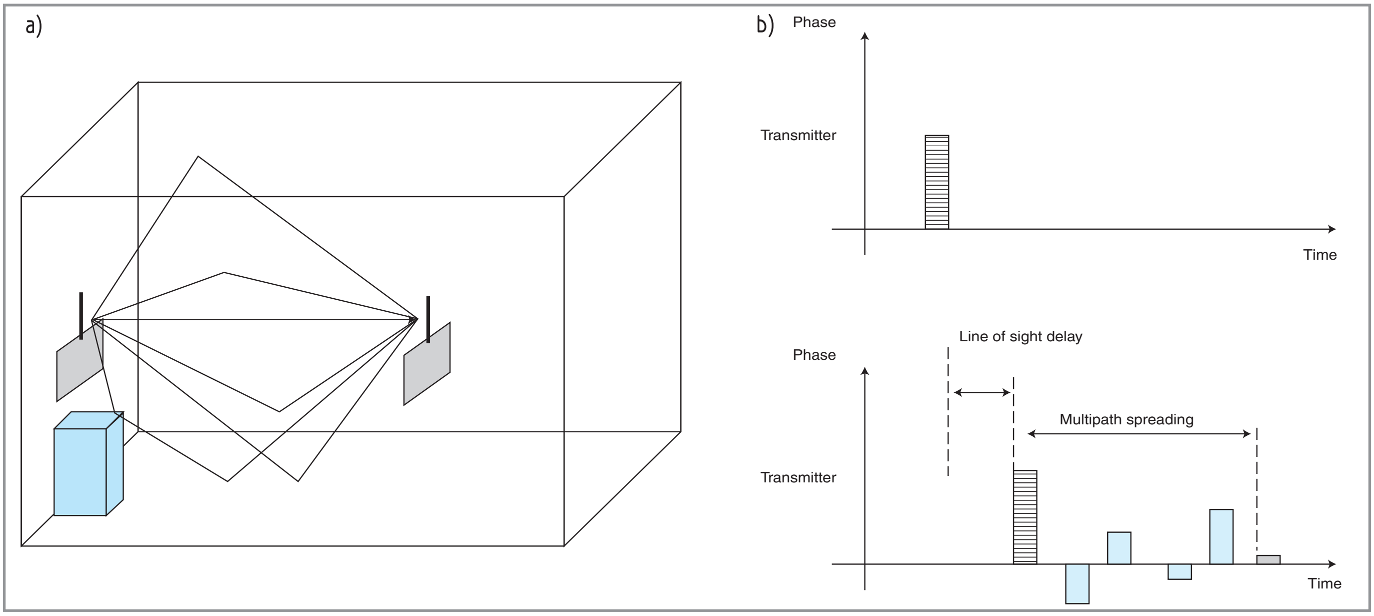


Figure 1a—Multipath fading occurs when a radio signal is transmitted between two nodes through several spatial channels due to reflections and the diffraction of radio waves on walls, floors, and other objects. b—In the time domain, when a short RF burst is transmitted by one node, the burst is received several times by the destination node, one time through each spatial path. Each reception is associated with a given attenuation and delay. If the different reflections overlap, there will be intersymbol interference.

signal, and the summation gives a heavy reduction of the signal strength.

Reciprocally, a reflected signal can be more or less in phase with the main signal, giving an artificially increased signal strength. The variations in the reception level due to the destructive interference is called multipath fading, or fast fading. Why is it fast? Simply because the typical displacement distance between peaks and valleys in the signal strength is roughly half a wavelength, which gives 6.25 cm at 2.4 GHz. Why? To move from a signal in phase to a signal out of phase is equivalent to increasing the path length by half a wavelength.

In summary, multipath fading is a fast varying reception level of a radio signal in the presence of multiple reflections. Note that the received signal level changes when one of the nodes is moved (transmitter or receiver), but also when the environmental conditions are modified (e.g., when moving an object that changes a reflection path). It also changes when the carrier frequency is modified as the phases are frequency-dependent.

The multipath effect is also one of the key contributors to another problem with high-speed RF links: intersymbol interference. When the duration of a binary bit is shorter than the time between the first and the last echoes, there will be a mix between the successive bits, making reception a challenge.

Multipath fading is a complex phenomenon. It's nearly impossible to simulate the reflections on every object. So it is more easily modeled with statistics. There are two basic models. One is the well-known Rayleigh fading model, which gives a good statistical estimation of multipath fading when there is no dominant line-of-sight propagation (i.e., nearly all power comes from reflected paths). It is typically used for the simulation of urban cellular networks as well as for tropospheric propagation. The other model, the Rician fading model, accounts for the presence of a line-of-sight propagation plus some reflections. Once again, both are statistical, giving

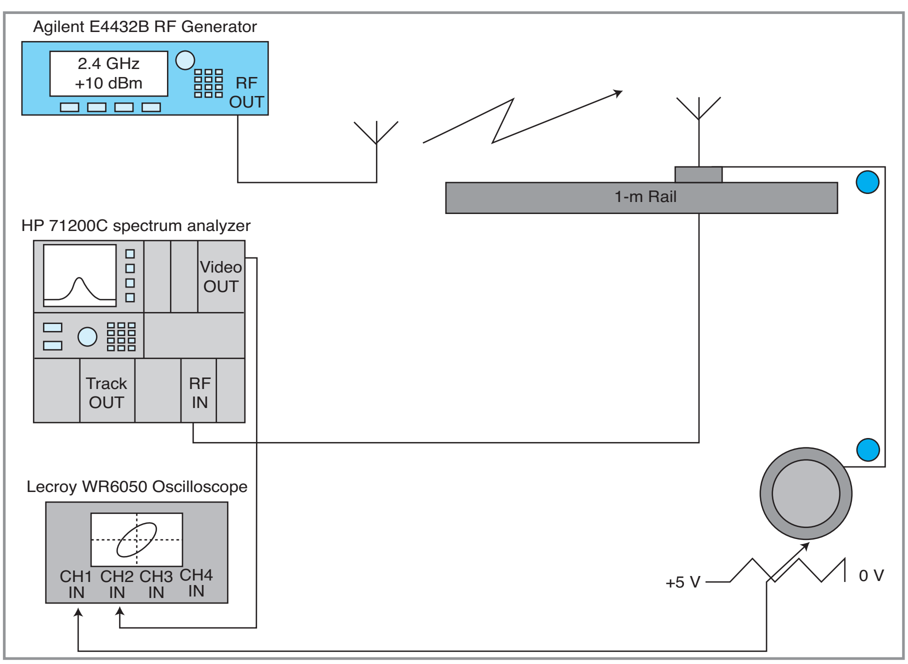


Figure 2—My test setup includes an RF generator, a pair of antennas, a sliding rail to move one of the antennas, a spectrum analyzer as a receiver, and, last but not least, a potentiometer to measure the position of the slider and to drive the x-input of the oscilloscope.

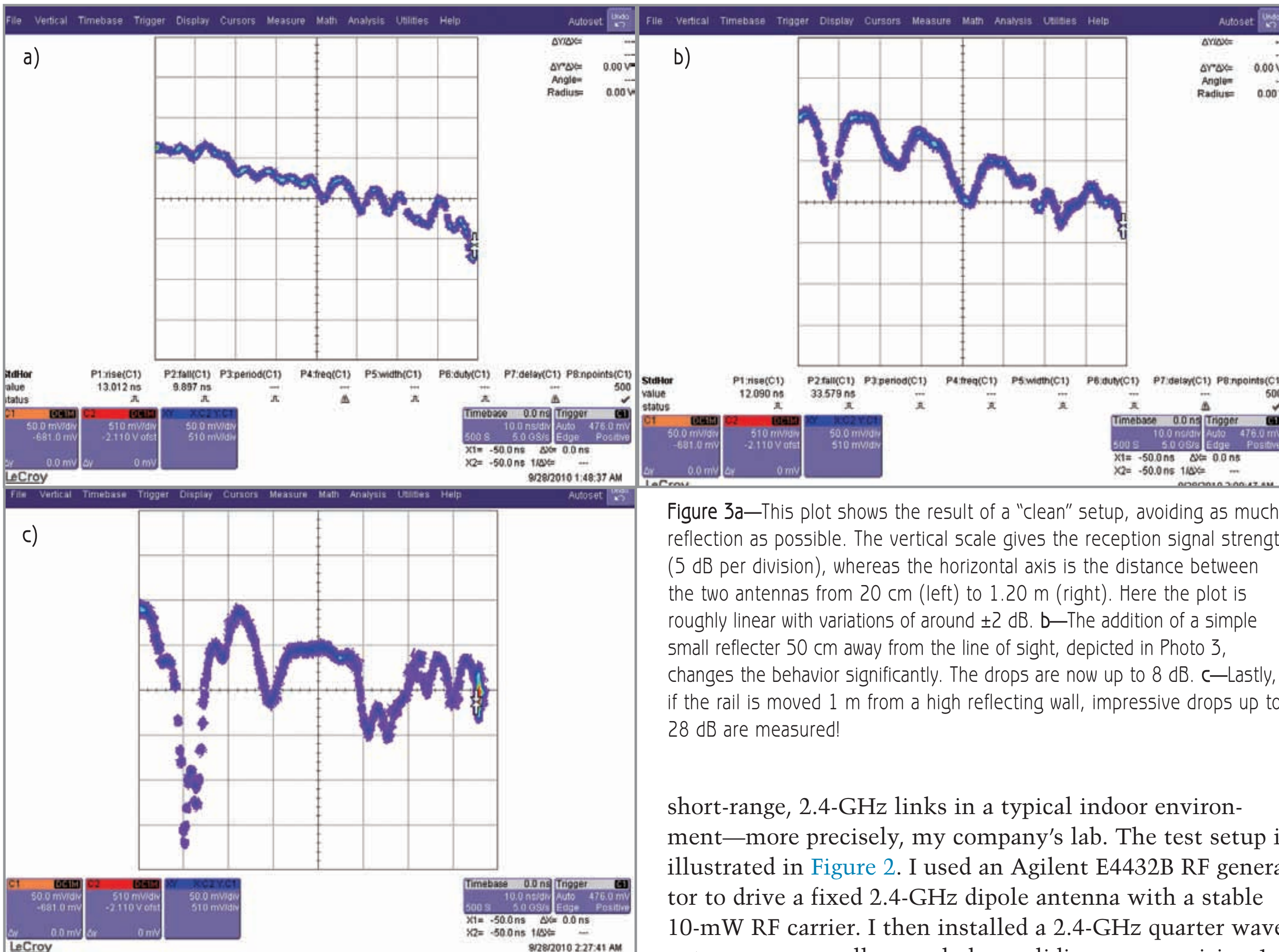


Figure 3a—This plot shows the result of a "clean" setup, avoiding as much reflection as possible. The vertical scale gives the reception signal strength (5 dB per division), whereas the horizontal axis is the distance between the two antennas from 20 cm (left) to 1.20 m (right). Here the plot is roughly linear with variations of around ±2 dB. b—The addition of a simple small reflecter 50 cm away from the line of sight, depicted in Photo 3, changes the behavior significantly. The drops are now up to 8 dB. c—Lastly, if the rail is moved 1 m from a high reflecting wall, impressive drops up to 28 dB are measured!

probabilities that the signal will be reduced by a given amount.

If you look at these models, you'll see that variations in the 10-dB range are very common, with occasional attenuation levels of 30 dB or more. Even 10 dB is not minor. It's no less than losing 90% of the signal power on some spots, and with 30 dB you lose 99.9% of the power. Therefore, when devices move or the environment changes, the multipath fading can give unexpected drops in signal quality in indoor or urban environments, or even loss of communication.

Let's consider a numerical example. A not-so-unusual 25-dB drop is equivalent to a coverage distance reduced by a factor of nearly 20. A system usually working with a 200-m range, like a classic Wi-Fi system, may not work in some locations 10 m from the transmitter! Multipath fading is also a nightmare when you're trying to use the signal strength, or RSSI, to estimate the distance between a transmitter and a receiver.

A final note on the theory: In the frequency domain, the same situation exists. For fixed antenna locations, the multipath fading effect gives "good" frequencies and other frequencies where high attenuation occurs.

TO THE BENCH

Theories are useful, but I always like to test them with experiments. I wanted to measure the fading effect on short-range, 2.4-GHz links in a typical indoor environment—more precisely, my company's lab. The test setup is illustrated in Figure 2. I used an Agilent E4432B RF generator to drive a fixed 2.4-GHz dipole antenna with a stable 10-mW RF carrier. I then installed a 2.4-GHz quarter wave antenna on a small ground plane, sliding on a precision 1-m aluminum rail. The line-of-sight distance between the two antennas was then adjustable from 20 cm to 1.20 m (six times farther), giving a theoretical attenuation of 15 dB from one end to the other.

The key to such an experiment is to make sure that the different measurements are made in exactly the same positions. I could have used a motorized slider, but I took a

Photo 1—The rail is 1 m long, which allows for a transmission distance from 20 cm to 1.20 m. You can see that the receiving antenna is fixed on a small ground plane to avoid ground effects. It is gently pulled with a fishing line.

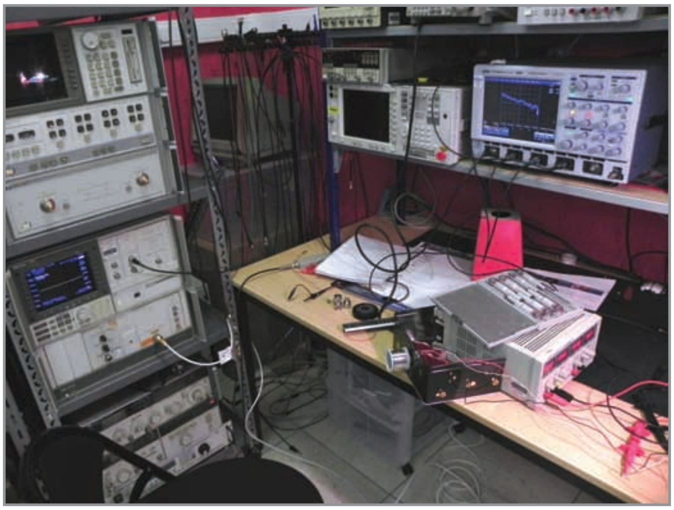


Photo 2—The fishing line is wound around a drum fixed on the shaft of the 10-turn potentiometer (bottom right). On the left, you can see the spectrum analyzer driving the oscilloscope on the top right.

simpler approach. The slider is simply pulled by a fishing line, which is wound around a small drum fixed on the shaft of a 10-turn potentiometer. Thus, when I manually turned the drum, I moved the slider without having to come too close to the antenna, and the potentiometer gave a voltage which is proportional to the position of the slider. I wanted to have attenuation versus distance plots, so I simply connected the potentiometer cursor to the x input of my oscilloscope, a Lecroy WaveRunner 6050 switched to x/y mode with a long screen persistence. To measure the reception signal strength, I connected the sliding antenna to my HP71200C spectrum analyzer configured as a tuned receiver, meaning in zero-span mode. (Refer to my *Circuit Cellar* 245 column on frequency domain equipment.) The test platform was ready after I used one last BNC cable to connect the video output of the analyzer to the y-axis input of the oscilloscope. Photo 1 and Photo 2 give you an idea of the actual test configuration.

I started with a clean environment without any reflective objects in the area. My office was a great location because the walls are made of a light material so reflection is limited. Figure 3a shows the result. As expected, the signal strength slowly decreases when the distance increases, even if there are some small variations. Overall attenuation from 20 cm to 1.20 m is 13 dB, which is in line with the 15-dB theoretical prediction.

I then simply added a small reflector 50 cm away from the line of sight. I placed the 19″ piece vertically on a table (see Photo 3). Figure 3b depicts a small modification with impressive results. The signal strength variations are now up to 8 dB! Moreover, when the antennas are just 30 cm apart, there is a strong minimum, giving a signal strength as low as with a 1 m distance without the reflector.

Lastly, I pulled the experiment close to my equipment rack, which is more or less full of metallic equipment. With the rail parallel to the rack and 1 m away from it, the result is shown in Figure 3c. This time the drops are up to 28 dB—meaning, in some positions, the antenna's signal power is reduced by a factor of 630 (i.e., $10^{28/10}$)! Impressive, isn't it? I must emphasize that these results are associated with direct line-of-sight transmission. If all signals come from reflections, the relative effect of multipath fading will be even larger.

COUNTERACTIONS

What if you can't accept such large variations? For example, what if you need a reliable link or you want to use the RSSI level to estimate a distance? The first solution is frequency hopping. Remember that fadings are related to the relative phases and amplitude of the received signals. Therefore, changing the frequency of the carrier RF frequency will change the position of the peaks and valleys. A frequency-hopping system will be more robust to multipath fading than a fixed-frequency system, at least as long as the protocol allows lost frames. At a given position, some frequencies will be less attenuated than others. Mathematicians have proved that, on average, the distance (in hertz) between two consecutive valleys is roughly inversely proportional to the time delay between the different signal paths. So, in engineering terms, the farther the frequency separation the better. Unfortunately, regulations say that the 2.4-GHz ISM band can be used only from 2.4 to 2.48 GHz in most countries, giving a maximum 80-MHz frequency-hopping distance. This translates to a 12-ns spreading time, or 2 × 1.5 m reflection. Therefore, frequency-hopping will be more efficient with long range links, but it will provide limited improvements when the distances are shorter than a couple of meters. This is exactly what the experiment showed, as you can see in Figure 4a. There is an improvement, but some significantly deep fadings still exists in my test configuration.

As an alternative to frequency hopping, you could use a spread spectrum modulation. It has roughly the same effects on multipath fading as frequency hopping, but only if the frequency spread is very large. Thus, a system using UWB technology, with 500 MHz or more bandwidth, probably

Photo 3—You can see the reflector I added to get the plot shown in Figure 3b. This is simply a 19″ instrument put vertically 50 cm from the antennas. It's a small environmental change, but it has a huge impact.

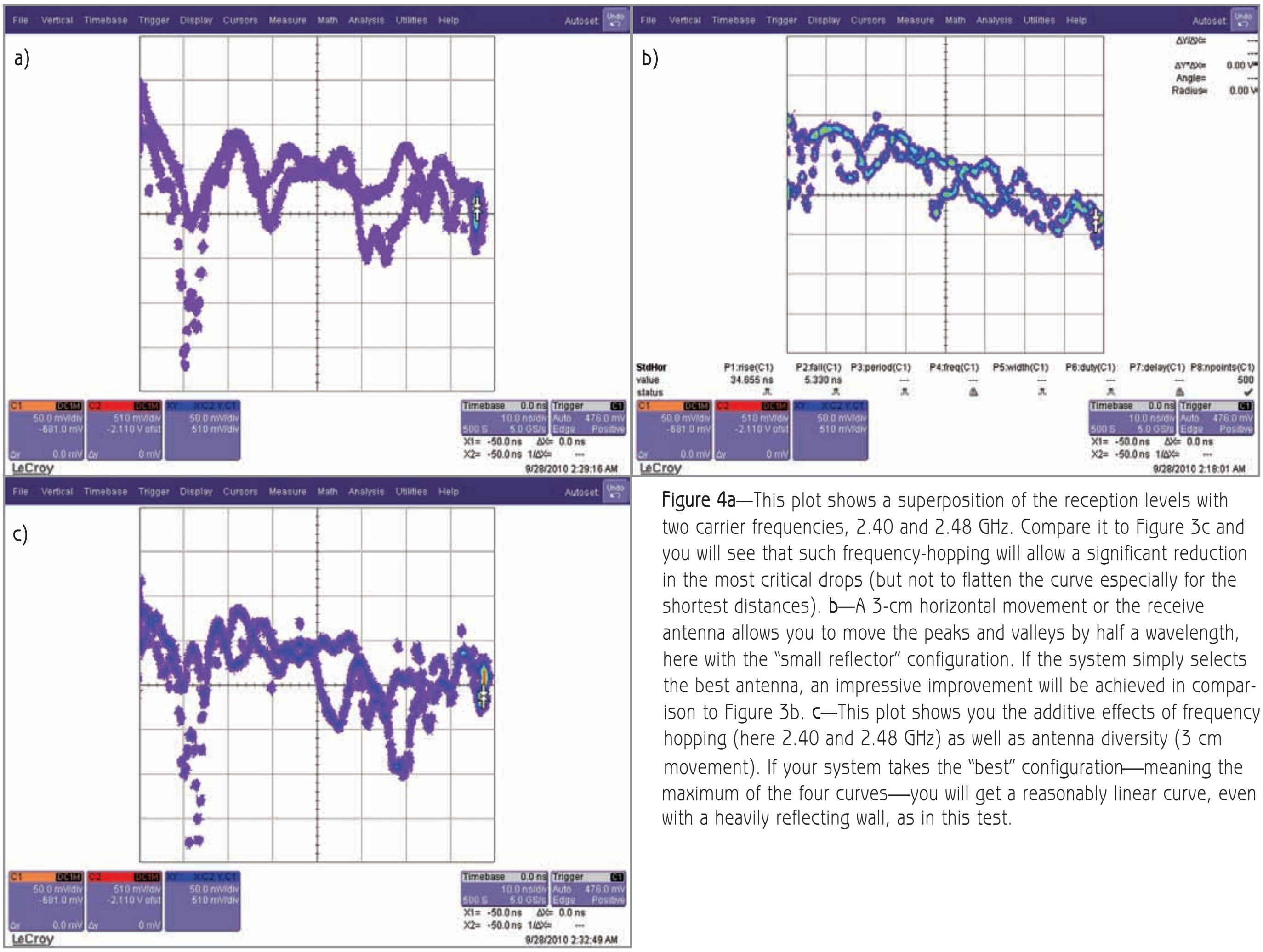


Figure 4a—This plot shows a superposition of the reception levels with two carrier frequencies, 2.40 and 2.48 GHz. Compare it to Figure 3c and you will see that such frequency-hopping will allow a significant reduction in the most critical drops (but not to flatten the curve especially for the shortest distances). b—A 3-cm horizontal movement or the receive antenna allows you to move the peaks and valleys by half a wavelength, here with the "small reflector" configuration. If the system simply selects the best antenna, an impressive improvement will be achieved in comparison to Figure 3b. c—This plot shows you the additive effects of frequency hopping (here 2.40 and 2.48 GHz) as well as antenna diversity (3 cm movement). If your system takes the "best" configuration—meaning the maximum of the four curves—you will get a reasonably linear curve, even with a heavily reflecting wall, as in this test.

won't suffer significantly from multipath fading. But this is less the case with typical 2.4-GHz systems like Wi-Fi or ZigBee, which use quite small modulation bandwidths (from 2 MHz for IEEE802.15.4 systems to 20 MHz for Wi-Fi). To convince myself, I switched the generator to a 2-MHz AWGN signal and I didn't get any visible improvement compared to a CW signal.

Any better solution? Of course! As the valleys in the signal strength are by nature limited to a small physical area, the best solution is to use antenna diversity. This means using two or more antennas on the receiver, or on the transmitter, separated by an adequate distance to be sure to avoid "bad spots" with at least one of the antennas. Do you remember that the average distance between valleys is roughly half the wavelength? So, ideally, the antennas should be separated by a quarter of the wavelength, which gives a little more than 3 cm at 2.4 GHz. Of course, your system will need to manage the two antennas, so you will need either an RF switch or two transceivers plus some firmware to select the "best" antenna at any given time. That will enable you to drastically improve the system's performance. I simulated it simply by taking two measurements with a 3 cm lateral displacement of the rail, and the result is provided in Figure 4b. If you always select the best signal, you need to look at the maximum of both curves, which no longer show any deep fading. Of course, nothing forbids you from using two antennas simultaneously and two or more frequencies (see Figure 4c), but it comes with a cost in terms of system and protocol complexity.

There are also some far more clever ways to exploit antenna diversity than just selecting the best one, but these are usually too complex for embedded designs, except when using ready-made modules, such as MIMO RAKE receivers. These systems implement several virtual subreceivers ("fingers"), each assigned to a different multipath component of the signal. Through a learning sequence they actually identify and extract each multipath, providing an improvement in signal quality, a reduction in intersymbol interference, and the possibility to increase the communication bandwidth. Amazing.

Returning again to simpler systems, note that "time diversity" (which could be as simple as sending the same message several times) is equivalent to spatial diversity when one of the nodes is moving. Some messages may not be received, but the next one will be. Unfortunately, this doesn't apply to fixed objects, but the solution is efficient for things like cellular systems, especially when associated with time-interleaving codes that "spread" the critical information over time.

Another, often complementary, solution is to use polarization diversity. This means using different polarizations for the two receiving or transmitting antennas. This helps for multipath fading, but also for "polarization fading"—when polarizations of both antennas are unfortunately 90° from each other. That's probably why the antennas on your Wi-Fi router are adjustable.

RF REVEALED

Let's summarize. Multipath fading can reduce the RF coverage of a transmission link by a factor of 20 or more if the system uses a fixed frequency and a single antenna. However, as predicted by the theory, and as confirmed by my experiments, you can use either frequency hopping and multiple antennas or polarization diversity to reduce its impacts typically down to a reasonable ±5 dB. That's still high, and you will need a more complex design to support it, but your system will be far more reliable.

I hope I've shed some light on the topic of RF technology. I strongly encourage you to reproduce these experiments yourself, even with simpler setups (e.g., with a pair of radio modules, using the RSSI level as a field strength indicator). Experimenting is the best way to learn more about the subject! ■

Robert Lacoste lives near Paris, France. He has 20 years of experience working on embedded systems, analog designs, and wireless telecommunications. He has won prizes in more than 15 international design contests. In 2003, Robert started a consulting company, ALCIOM, to share his passion for innovative mixed-signal designs. You can reach him at rlacoste@alciom.com. Don't forget to write "Darker Side" in the subject line to bypass his spam filters.

RESOURCES

A. Bicchi, K. Henrik, and M. Lindh, "An Experimental Study of Exploiting Multipath Fading for Robot Communications," Proceedings of Robotics: Science and Systems, 2007, www.roboticsproceedings.org/rss03/p37.pdf.

M. Burns and T. Starr, "Implementing Diversity Using Low Power Radios," AN085, Texas Instruments, http://focus.ti.com/lit/an/swra317/swra317.pdf.

R. Lacoste, "Multipath Fading Effects on Short Range Indoor RF Links," RL1039-004, Alciom, 2010.

SOURCES

E4432B Digital RF generator and HP71200C spectrum analyzer
Agilent | www.agilent.com

Waverunner 6050 oscilloscope
Lecroy | www.lecroy.com

ICbank
Gateway to Electronics

3 PORT INTERFACE
RS-485 to Ethernet Converter

RS-485 to Ethernet Converter

Powerful feature

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

Specification

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

MP3P DIY KIT, Do it yourself

(Include Firmware Full source Code, Schematic)

· myPIC

Only
$160 qty 100 **$220** qty 1

Powerful feature

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

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

· myWave (MP3 DIY KIT SD card Interface)

Only
$150 qty 100
$200 qty 1

Powerful feature

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

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

· myAudio (MP3 DIY KIT IDE)

Only
$180 qty 100
$220 qty 1

Powerful feature

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

ICbank Gateway to Electronics

Youngsan Cho / Senior Account Manager / Sales & Marketing ICbank Inc.

Tel : 82-707-019-3924. Fax : 82-2-6340-1888. Cell : 82-17-268-2078. Email : youngsan@icbank.com
18F Ace High -End 2, 222-14 Guro Digital Valley, Guro-dong, Seoul, 152-050, Korea

Website : http://www.icbank.com/

by Ed Nisley (USA)

Parallel NiMH Cell Measurement

Got batteries? You need to test them. Here you learn how to build an eight-channel parallel cell tester, with an Arduino Mega microcontroller board providing PWM DAC outputs to control the discharge currents, ADC inputs to measure the cell voltages under load, and serial output over a USB connection for data logging.

Had portable electronic devices existed around 1500, Desiderius Erasmus might have said: "Batteries: can't live with them, can't live without them." Two centuries after Alessandro Volta invented the Voltaic Pile, electrochemical cells remain the weak link in our gadgets: they're always dead or dying when you need them.

I've been using a West Mountain Radio CBA II battery analyzer to measure battery and cell performance. However, it measures one cell at a time, which means testing all eight cells from a 9.6-V NiMH pack at real-world discharge currents takes a week: eight 2500-mAh cells at 500 mA might require 40 hours!

So I'm building an eight-channel parallel cell tester, with an Arduino Mega microcontroller board providing PWM DAC outputs to control the discharge currents, ADC inputs to measure the cell voltages under load, and serial output over a USB connection for data logging. Photo 1 shows the four-channel breadboard that produced the data for this column; a higher-priority task got in the way of finishing the second circuit board by the deadline.

Sometimes life is like that.

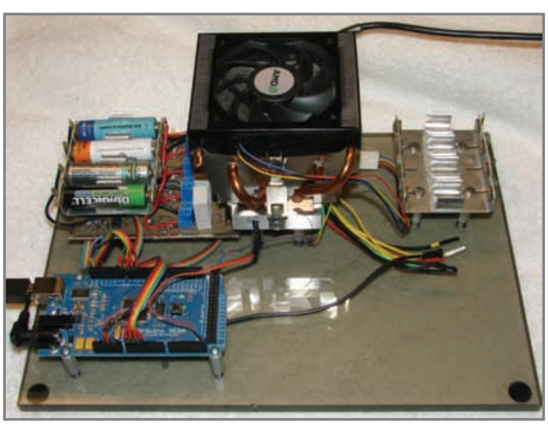


Photo 1—This breadboard setup allows convenient access to all the parts, but isn't suitable for a finished unit. The mismatched quartet of AA NiMH cells provided the data for the graphs in this column. The empty cell holder on the right shows where the second circuit board will fit; until then, the heatsink fan is completely unnecessary.

DISCHARGE CHANNELS

The four-cell holder on the left side of the heatsink in Photo 1 sits atop standoffs on the circuit board shown in Photo 2. The four test channels line up across the board, with DAC inputs from the Arduino Mega arriving on the ribbon cable in the foreground and power MOSFETS attached to the heat spreader plate under the heatsink in the background, behind the current sensing resistors.

The schematic in Figure 1 shows the circuitry corresponding to a single cell. Two circuit boards, each with four channels, make up an eight-cell tester, with all eight MOSFET current sink transistors sharing the common heatsink appearing in the middle of Photo 1.

R1 and C1 convert the Arduino's PWM analog output into a DC voltage corresponding to the desired discharge current. Because contemporary NiMH AA cells have capacities around 2 Ah, I decided on a 2.5 A maximum discharge current. As a result, the 0-to-5-V DAC output produces a 0-to-2.5-A load current and the circuit has an overall transconductance of 500 mA/V.

The default Arduino PWM frequency is 488 Hz, which is ordinarily too low for simple low-pass filtering. However, I recently bought some 1206-size 10-µF surface-mount ceramic capacitors that, in combination with a 10-kΩ SMD resistor, produce a time constant of 100 ms and a rolloff around 1.6 Hz, while using minimal board space. The tradeoff for a very low rolloff is that the DAC setting cannot change more than a few times each second; that's acceptable in this application.

Figure 2 shows the DAC output for a 250-mA load current: the PWM signal has an 11% duty cycle that produces about 53 mV DC, with no detectable 488-Hz ripple. Under ideal circumstances the duty cycle would be 10% for a 50-mV output, but, as we'll see later, the hardware may require a slight offset.

Op-amp IC1B converts the 100-mV/A output from R8, the current-sense resistor, into 1 V/A. IC1A drives Q1's gate voltage so that the voltage from IC1B exactly matches the value produced from the voltage divider formed by R2 and R5, thus compensating for the MOSFET's strongly nonlinear voltage-to-current characteristic.

The negative feedback loop around IC1A closes through Q1 and IC1B, a long and slow path that's a recipe for oscillation. C6 and C7 reduce the bandwidth of the op-amps, taking advantage of the fact that this circuit operates at almost DC "frequencies."

The MOSFETs act as both constant-current loads and power dissipation elements: a 2.5-A load on a 1.2-V NiMH cell produces 3 W. Because operating all eight channels at that level will dump 24 W, I mounted the transistors on a thick aluminum plate capped with a surplus CPU heatsink. Thermally conductive pads insulate the MOSFET body tabs from the heat spreader plate.

Because less aggressive discharge currents produce proportionately less power (500 mA dumps less than 5 W), I inserted an LM335 temperature sensor into a hole drilled in the heat spreader and controlled the fan through a digital output using a small MOSFET. Those parts don't appear on the schematic, but the circuitry should be obvious after you consult the datasheets and source code.

CIRCUIT SUBTLETIES

The physical requirement of mounting the MOSFETs to the heat spreader dictated the board layout: each channel occupies a narrow slice of the board. Most of the components are on the top surface shown in Photo 2, with the bypass caps and gain-setting resistors on the bottom surface.

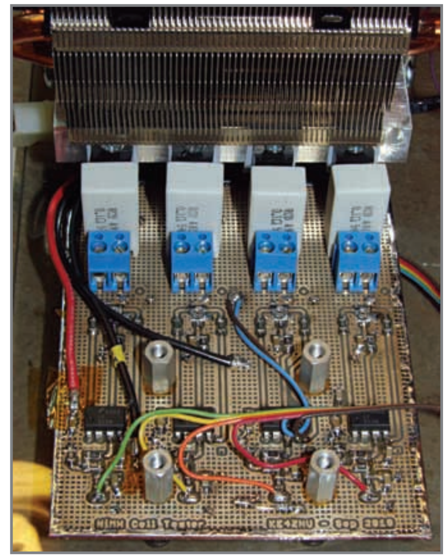


Photo 2—The top side of the board holds most of the components, with a few SMD parts on the underside. Thermally conductive pads separate the MOSFET drain tabs from the aluminum heat spreader plate below the heatsink.

The layout separates the high-current cell discharge paths around the current-sensing resistors from the low-voltage PWM filters at the other end of the slices. Because the power supply doesn't source or sink high currents, connecting its common ("ground") lead near the center of the board minimizes the effect of any IR drops from the MOSFET currents.

I used a power supply brick that produces regulated +5-, –5-, and +12-V outputs, terminated in a modified four-pin Molex-style connector of the type used for PC IDE hard drives; a yellow band on one of the black leads identifies it as –5 V, rather than a second common lead. The +12-V supply powers the heatsink's cooling fan and the +5-V regulator on the Arduino Mega microcontroller board.

Although the Arduino can derive power from its USB host connection, that's not suitable for this application due to its effect on the PWM outputs. The DC value of a low-pass filtered PWM output is directly proportional to the digital signal's maximum voltage, which, in turn, depends on the logic supply voltage. The USB supply voltage differs on each computer in my collection and, worse, the current passes through a fuse and a MOSFET switch on the Arduino board that add an unpredictable series resistance. Therefore, I

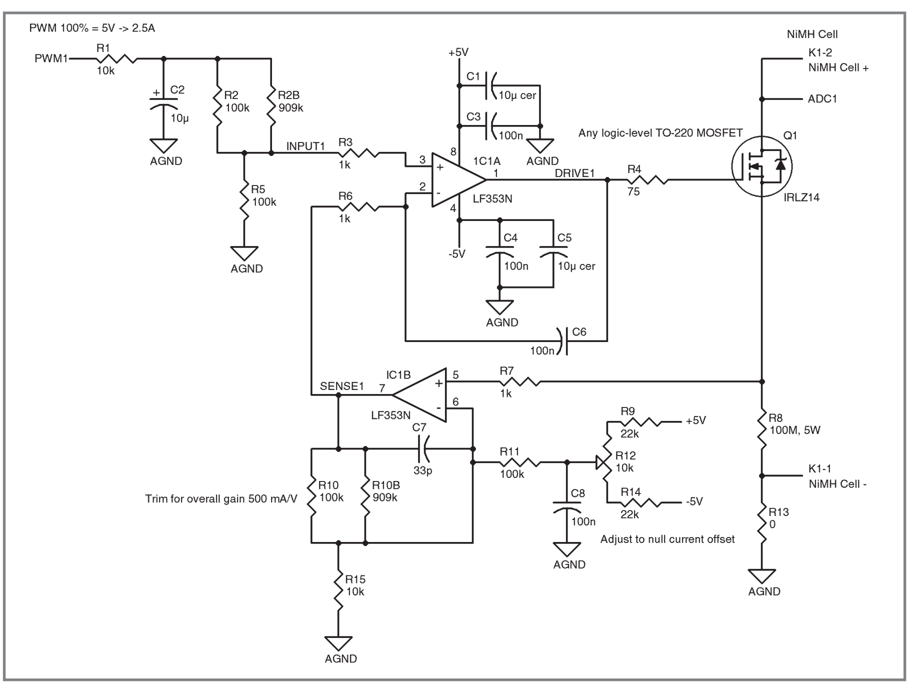


Figure 1—Each circuit board has four identical channels that convert a PWM signal into a DC voltage that sets the load current for the NiMH cell. The Arduino microcontroller board must have an external power supply to ensure a constant VCC voltage.

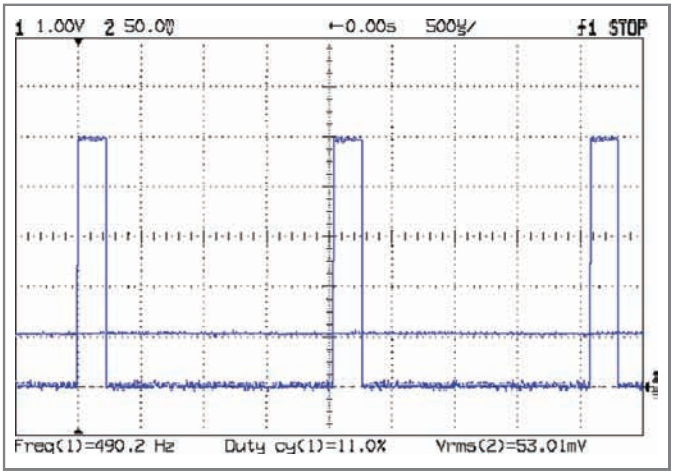


Figure 2—An RC filter converts the PWM output to a DC voltage. Note the vast difference in vertical scales.

used an external +12-V supply to get a constant +5-V logic level for the PWM outputs.

With that in place, I found that the Arduino Mega board draws about 75 mA with no additional I/O devices: the on-board 5-V SMD regulator dissipates half a watt from a 12-V supply. I epoxied a small DIP-IC-style heatsink atop the regulator to reduce its case temperature. A much larger heatsink would be in order for applications drawing significant output current from the digital pins.

I chose IRLZ14 MOSFETs for their logic-level gate input voltages. While the industry still hasn't standardized a definition of what "logic-level MOSFET" means, which is understandable in an age when logic power supplies are falling from 5 V through 3.3 V on their way below 1 V, these MOSFETs are characterized to pass 6 A at V_{GS}= 4 V.

The LM353 dual-supply op-amps are relatively old devices with a narrow input common-mode range and a restricted output swing. However, you'll face a similar problem with contemporary single-supply op-amps: this circuit requires good linearity for inputs near 0 V, coupled with about 4 V of MOSFET drive.

The split +5/–5 V supply moves the negative common-mode limit to about –2 V, well below the minimum current-sensing resistor output. The MOSFET gate requires no DC drive current and the very low operating frequency eliminates any problem with settling time, so an LM353 can provide just enough voltage to properly drive the gate.

If you're using a single-supply op-amp, however, pay attention to its specs to avoid being tripped by tighter common-mode limits and lower output drive than your circuit requires. Many analog applications require higher supply voltages than you may expect, simply because not all analog properties scale neatly to digital supply voltages.

CALIBRATION & FIRMWARE

I wrote a calibration routine to step the load currents from 0 to 2000 mA, pausing 5 seconds at each 100-mA step, while I measured the actual current on an ammeter. Graphing that current against the setpoint, then doing a linear regression, revealed the gain and offset errors for each channel.

My stock of 100-mΩ current-sensing resistors has 5% accuracy, but the four I picked lie within about 2% of each other and that spread accounted for all of the observed gain errors. I adjusted the trimpot on each channel to remove its offset error.

While it would be feasible to correct the gain errors in software, normal Arduino PWM outputs have 0.4% resolution: 1 count in 256. You can improve the resolution with a dab of

```
# NiMH Cell Tester
# Ed Nisley - KE4ZNU - 6 Oct 2010
# Circuit Cellar - February 2011

# Load current: 500 mA
# Dropout voltage: 500 mV
# Minimum delta-V: 10 mV
# Temperature ADC offset & scale: 0 (1000/1000)
# Voltage ADC offset & scale: 0 (1000/1000)
# Current DAC offset & scale: 3 (1000/1000)

# Temperature: 20C = 68F
# Fan is off

# Initial cell voltage and state
# 0 1348 testable
# 1 1348 testable
# 2 1344 testable
# 3 1344 testable

# Starting test
# seconds, (cell V, mA, cumulative mW x sec), instant mW , heatsink temp C, fan state
0,1348,500,0,1348,500,0,1344,500,0,1344,500,0,0,20,0
1,1319,500,666,1329,500,669,1304,500,662,1314,500,664,2661,20,0
12,1309,500,7903,1324,500,7963,1300,500,7823,1309,500,7873,2626,20,0
… omitted …
12319,503,500,7394897,1089,500,7515992,962,500,7329132,1080,0,7254134,1283,27,0
# Cell discharged: 0
12320,493,500,7395146,1089,500,7516536,962,500,7329613,1080,0,7254134,1274,27,0
 … omitted …
12056,1021,500,7279058,1124,500,7370036,1055,500,7195058,508,500,7253883,1857,27,0
# Cell discharged: 3
12057,1016,500,7279567,1124,500,7370598,1055,500,7195585,498,500,7254134,1849,27,0
… omitted …
12530,1060,0,7395146,1021,500,7628139,513,500,7417677,1119,0,7254134,771,26,0
# Cell discharged: 2
12531,1065,0,7395146,1021,500,7628649,498,500,7417929,1119,0,7254134,762,26,0
… omitted …
12789,1119,0,7395146,503,500,7742973,1070,0,7417929,1133,0,7254134,255,25,0
# Cell discharged: 1
12790,1124,0,7395146,493,500,7743222,1070,0,7417929,1133,0,7254134,249,25,0
# All cells discharged: test complete
```

Figure 3—This excerpt from the output produced for a quartet of Sanyo Eneloop cells shows the CSV format. The heatsink barely warms up during this low-power test.

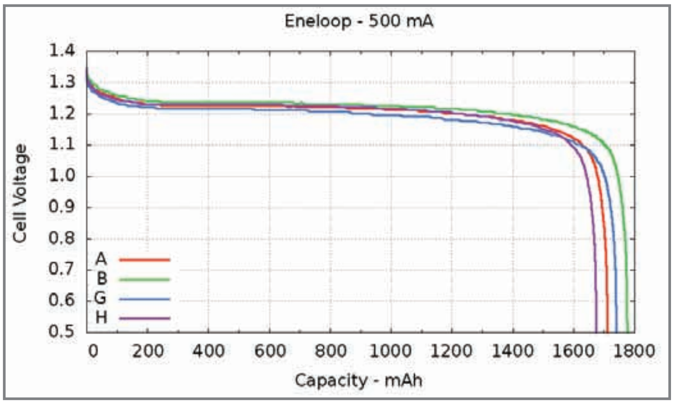


Figure 4—Four new Sanyo Eneloop cells deliver nearly their rated 1,900-mAh capacity at 500 mA after about three weeks of storage. The 5% accuracy of the current-sense resistors accounts for most of the observed ±3% variation, with normal cell differences providing the rest. This plot uses data from the file shown in Figure 3.

custom firmware that uses the hardware's 16-bit PWM modes, the simple fact is that a few percent accuracy is entirely adequate for this application.

Remeasuring the outputs showed a slight offset common to all the channels, so I added a correction to the PWM values. The firmware also applies a gain adjustment, but it's set to 1.0 and has no effect.

I eventually plan to add an LCD to display some status information, an encoder knob to select the test current, and a few buttons. Those are all in the nature of fine tuning and don't affect the results in this column.

DATA FORMATTING

Although the Atmel ATmega1280 microcontroller has 128 kB of flash program memory, the firmware uses barely 7 kB. The setup routine applies a test current and measures the cell voltage; an empty channel returns 0 V and won't be tested. The main loop samples the cell voltages each second, but writes an output record only when any voltage changes by more than 10 mV or 100 seconds elapse since the previous output.

Figure 3 shows the interesting sections of a four-cell discharge test. The firmware produces output formatted as an ordinary comma-separated value (CSV) file that can be captured by a terminal program and imported into a spreadsheet or graphed directly by Gnuplot, as in Figure 4. The header comments summarize the test conditions and display internal firmware values, the heatsink temperature, and the initial cell voltages.

Each data record presents the elapsed time in seconds; the voltage, current, and cumulative energy for each cell; the total power; the heatsink temperature; and the cooling fan state. A comment line identifies when each cell's voltage falls below the dropout level, at which point the firmware sets that cell's load current to zero. The complete file for this test has 331 lines and occupies about 25 kB.

In contrast, the West Mountain Radio CBA II emits a fluffy XML output record *each second* during the entire test for a

Listing 1—This Bash script invokes Gnuplot to convert CSV files into plots similar to Figure 4. The trinary expression compares the load current for each cell with zero to suppress data points after the cell has completely discharged.

```
#!/bin/sh
export GDFONTPATH="/usr/share/fonts/TTF/"
gnuplot << EOF
set term png font "arialbd.ttf" 18 size 950,600
set output "$2.png"
set title "$2"
set key noautotitles left bottom
unset mouse
set bmargin 4
set grid xtics ytics
set xlabel "Capacity - mAh"
set format x "%4.0f"
set mxtics 2
set ylabel "Cell Voltage"
set format y "%3.1f"
set yrange [0.5:1.4]
set datafile separator ","
plot       \
    "$1" \
    using (\$3*\$1/3600):(\$3 > 0 ? \$2/1000 : NaN) title "$3" with lines lt 1 lw 3, \
    "$1" \
    using (\$6*\$1/3600):(\$6 > 0 ? \$5/1000 : NaN) title "$4" with lines lt 2 lw 3, \
    "$1" \
    using (\$9*\$1/3600):(\$9 > 0 ? \$8/1000 : NaN) title "$5" with lines lt 3 lw 3, \
    "$1" \
    using (\$12*\$1/3600):(\$12 > 0 ? \$11/1000 : NaN) title "$6" with lines lt 4 lw 3
EOF
```

single cell: a file containing data for eight cells weighs in at 6.5 MB!

Because the cells discharge at a constant current, the horizontal axis of a test graph can display the elapsed test time in seconds, the cumulative cell capacity in mAh, or the delivered energy in Wh. The Gnuplot commands in Listing 1 produced Figure 4 from the file shown in Figure 3; the mAh values along the x-axis equal the product of the constant load current and the elapsed time converted from seconds to hours.

The snippet of code in Listing 2 shows how the firmware computes the voltage change, instantaneous power, and total energy at each sample. I applied integer arithmetic (to eliminate the need for a floating-point package) with basic values in engineering units of ms, mV, and mA. The scaling values convert milliseconds to seconds and µW (mV × mA) to mW for output.

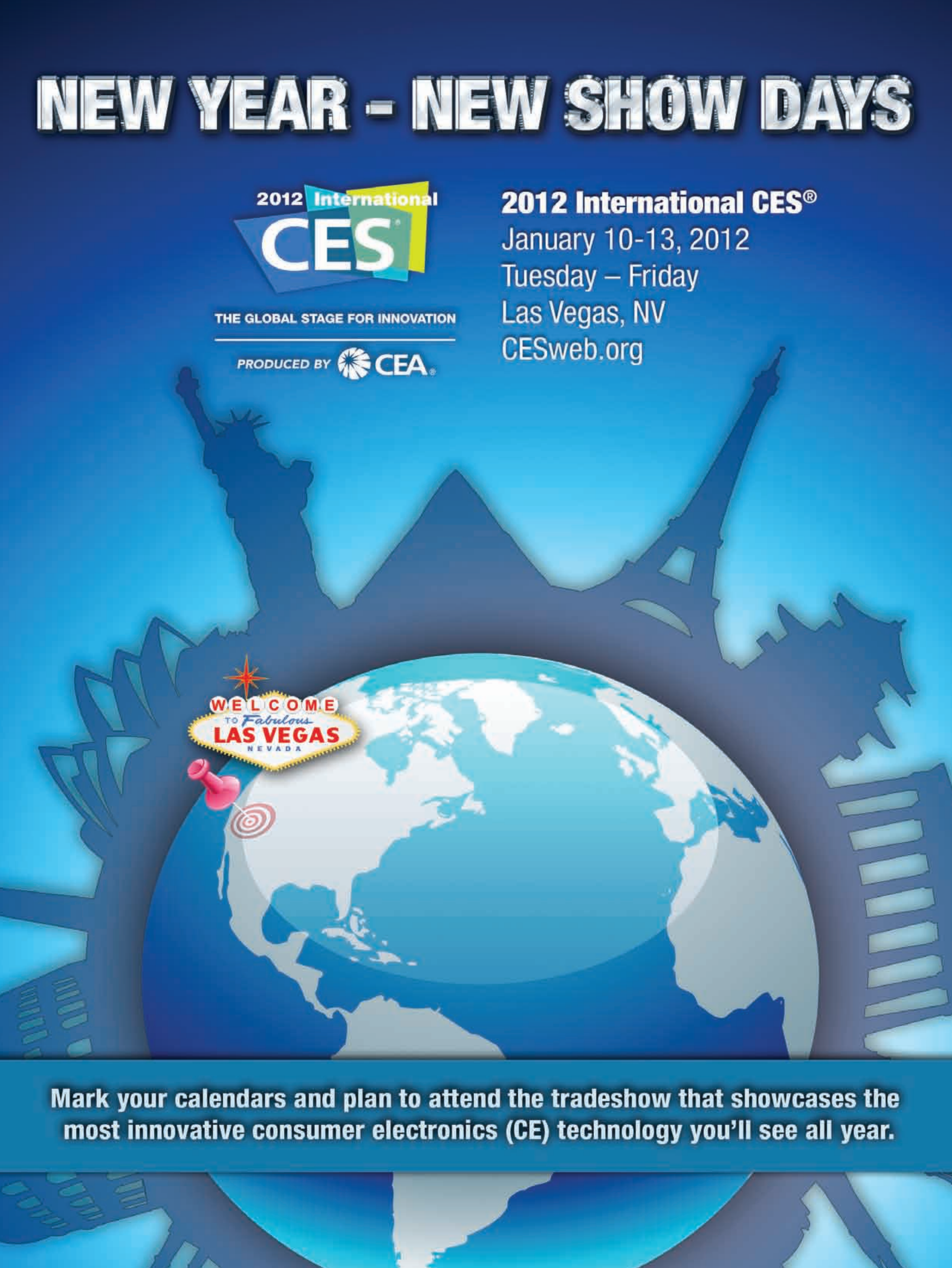
The many typecasts in Listing 2 ensure that the arithmetic operations produce the proper results. For example, I store voltages as `unsigned int` variables because those values will never be less than zero. Their difference can be a negative number, but the unsigned arithmetic result of 2-3 is 65535. The typecasts tell the compiler how to interpret the arithmetic operations in order to get the correct results.

Listing 2—This code snippet calculates the voltage change since the last measurement, the instantaneous power in mW, and the average energy in mW × s. The numerous type casts ensure the proper combinations of variable length and signs.

```
typedef struct {
       byte Status;   // true while testing, false for idle
       unsigned int NewVoltage;      // most recent measurement
       unsigned int LoggedVoltage;   // voltage sent to output
       unsigned int Current; // load current
       unsigned long int Energy;     // accumulated mA x seconds
} CELLDATA;

CELLDATA Cell[NUM_CELLS];     // per-cell information

unsigned long int NowTime;    // this time, right now
unsigned long int LastTime;   // previous time of interest

… much code omitted ...

int DeltaV = (int)Cell[i].NewVoltage - (int)Cell[i].LoggedVoltage;
unsigned int Power = ((unsigned long int)Cell[i].Current *
       ((Cell[i].NewVoltage + Cell[i].LoggedVoltage) / 2)) / 1000ul;
unsigned long int Energy = Power * ((NowTime - LastTime) / 1000ul);

TotalPower += Power;
Cell[i].Energy += Energy;
```

CELL SAMPLES

Figure 5 shows the test results for four cells, each chosen from my motley assortment to illustrate an interesting property. My digital camera reports a pair of 2,650-mAh Duracells as completely discharged after just a few minutes, even though they should have plenty of capacity. The red trace


NEW YEAR - NEW SHOW DAYS
2012 International CES
THE GLOBAL STAGE FOR INNOVATION
PRODUCED BY CEA
2012 International CES®
January 10-13, 2012
Tuesday – Friday
Las Vegas, NV
CESweb.org
WELCOME TO Fabulous LAS VEGAS NEVADA
Mark your calendars and plan to attend the tradeshow that showcases the most innovative consumer electronics (CE) technology you'll see all year.

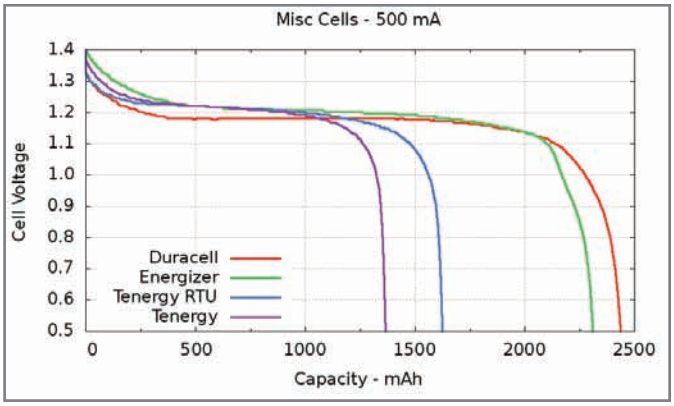


Figure 5—These four mismatched cells demonstrate a range of problems.

corresponding to one Duracell has the highest capacity as measured in mAh, but the lowest voltage during the discharge. The camera uses the battery's voltage as a proxy for its charge state and, as with most devices, rejects batteries still holding plenty of energy.

A pair of 2,200-mAh Energizer cells produce slightly longer life in that same camera. The green trace reveals a higher terminal voltage as it delivers nearly its rated capacity.

The blue Tenergy Ready To Use (RTU) trace shows that 2,300-mAh cell produces much less than its rated capacity, even though it's relatively new. I've been using these in our bike lights because they have a much lower self-discharge than standard NiMH cells, although they drain much faster than the Eneloops in Figure 4.

The purple Tenergy trace has a reasonable terminal voltage, but produces only half of its rated 2,600-mAh capacity. Those cells have been a continuing disappointment; even when new they had a low capacity.

Figure 6 shows an advantage of having the entire test system available for tinkering: you can measure and display any values you like. This graph shows the power produced by each cell and the resulting heatsink temperature due to the total power dissipation. Obviously there's no need for a fan at these low power levels!

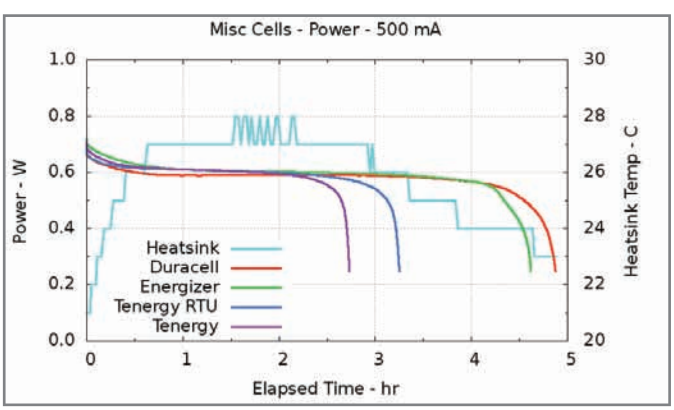


Figure 6—Displaying additional information becomes a simple matter of software when you have the entire measurement and display system under your control. This graph uses the same data file as Figure 5, but plots the delivered power and heatsink temperature against time.

The hardware can apply different load currents to each cell, so you could tweak the firmware to maintain a constant power load rather than a constant current. Based on the shape of the curves in Figure 6, that wouldn't change a cell's actual capacity by very much, but it will simulate the loads found in portable electronic devices.

You could apply a pulsed load current that more closely simulated a digital camera's requirements. I suspect cells would deliver more energy under that regime than under constant load.

Because the ADC inputs can handle up to 5 V, you can also test lithium-ion cells and two-cell NiMH packs. To test larger packs, add a voltage divider to protect the microcontroller's ADC inputs from excessive voltage due to multi-cell packs.

CONTACT RELEASE

My December 2008 column presented capacity tests for several of the packs and cells in my collection, along with a simple slow charger. In April 2007 I took a detailed look at the cell voltages and temperatures produced during fast charging. I started that group of columns in February 2007 by examining the performance of various battery chemistries. Back in 2003, I presented a lead-acid battery float charger.

All this information should help you live happily with your batteries! ■

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

PROJECT FILES

To download the schematic, PCB layout, and CNC files, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/247.

RESOURCES

"Desiderius Erasmus Quotes," Thinkexist.com, http://thinkexist.com/quotes/desiderius_erasmus.

History of the Battery, Wikipedia, http://en.wikipedia.org/wiki/History_of_the_battery.

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

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

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

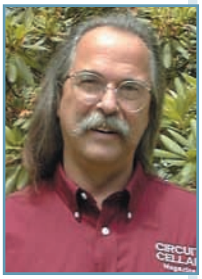
———, "Real-World NiMH Charging," *Circuit Cellar* 221, 2008.

SOURCES

Arduino Mega
http://arduino.cc

CBA Series
West Mountain Radio
www.westmountainradio.com/content.php?page=cba

by Jeff Bachiochi (USA)



Serving Up BASIC

Hardware developers need to understand software, and software developers need to understand hardware. Here you learn about an effective solution for "serving up" BASIC programs.

On an unusually warm day last fall, I decided to go for a run around Crystal Lake, which is located near my home. As I ran, I took in many of the sights and smells of a beautiful autumn day in New England. Someone was burning leaves in their yard, and many of the homes I passed had porches adorned with pumpkins. On the lake, a lone fisherman was trolling for "the catch of the day." I also noticed that the floating boundaries for the town beach had been removed from the lake, as had most of the private rafts and docks that had been stationed along lake's perimeter all summer.

	Files		
Entries	Companies.txt	Company A.txt (one file for each company)	Operations.txt
1	none	none	None
2	Company A	Project 1	Develop Plan
3	Company B	Project 2	Schematic
4	Company C	Project 3	BOM
5	Company D	Project 4	PCB Layout
6	Company E		Coding
7	Company F		Debugging
8	Company G		Fabrication
9	Company H		Documentation

Table 1—These lists hold all the information I need to keep track of multiple projects for multiple companies. The Companies.txt list contains Company Names. Each Company has its own list of projects. The Operations.txt list includes every task that can be performed to complete a project. For instance, I created a password for readers in the users.txt file: "FTB." This password creates a log file FTB.txt and a state file FTBState.txt. These are accessed whenever a reader logs in with the password "FTB."

After completing my normal circuit, I gathered my mail and newspaper and then settled down on my steps to cool down with a tall glass of iced tea. Then it hit me: I hadn't removed my dock from the lake's rapidly cooling water. I realized that if I didn't do it immediately, I'd soon need a wet suit to enter the freezing water. Everyone in the neighborhood knows that once the lake water freezes around a dock's supporting beams, it's over. The freezing and thawing process can rip a dock to shreds, and an ice floe can drag pieces of a dock away from shore and deposit them in their final resting place—Davy Jones's locker.

Every year, in the spring, I say to myself: "Remember what happened last year? You waited too long to remove your dock. Don't let it happen this fall." But guess what happens. Every fall, I end up shivering while pulling in my dock. I suppose it's just my way of refusing to believe the calendar has but a few pages left. If I could just hold onto summer, I'd be able to shorten the fall and limit winter's magnitude. Dream on.

Ask anyone why they live in, or don't live in, New England, and you'll get the same answer: the four seasons. New England is about change. So, this month, I'll do something a little different. I want to bring to your attention software that has actually been around for a few years now, but you probably haven't heard of it. Shoptalk Systems brought BASIC programming to the Windows environment

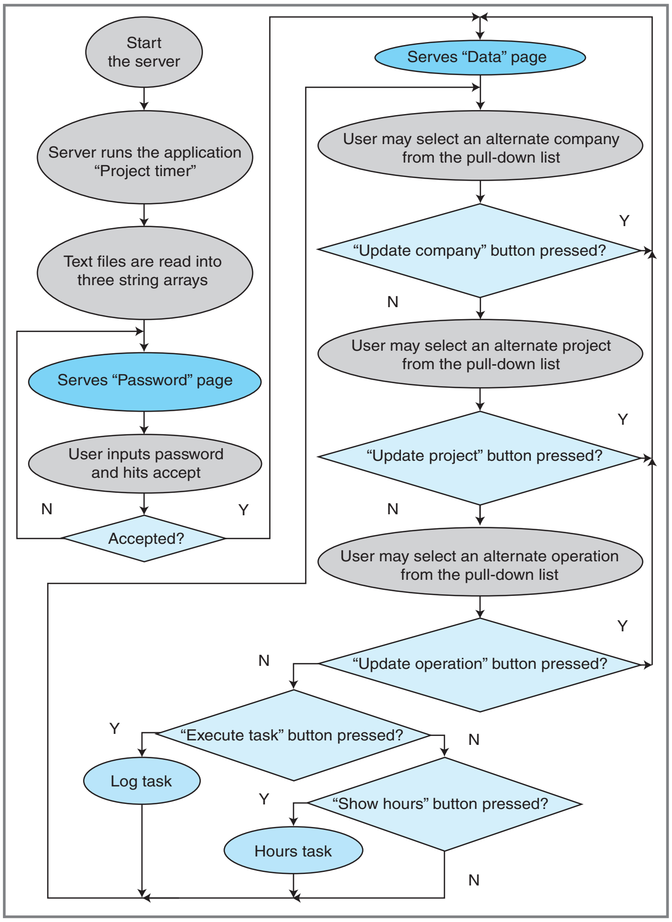


Figure 1—This application consists of two webpages, a password page and a data page. Once entering via a password, you can easily end a previously started task or select a new task to begin. Finished tasks are appended to a log file that keeps track of the time spent for each task.

with Liberty BASIC. I've been using it for things since the early 1990s and have even presented a few projects featuring it along the way. It's latest product is a web programming solution called Run BASIC.

LANGUAGE LEARNING

It is hard to believe that the web has been around a measly 20 years or so. When I had my first experience with the world-wide web, I was intrigued with the possibilities that such a connection exposed to everyone. The only catch was having to learn a new language, HTML.

HTML is a text and image formatting language used to format webpages and was derived from the commands used by typesetters to manually format documents at the time. The page designer uses HTML "tags," or plain text switches, to turn on and off formatting functions, like underlining text on screen. The "begin underline" tag <u> instructs the browser to begin underlining the following text until it finds an "end underline" tag </u>. Pretty simple.

The basic webpage might contain a `doctype` declaration that helps the browser determine how to interpret the HTML tags, along with page definition, enclosed in <html> begin and </html> end tags. This basic page is divided into two sections with the <head> </head> and the <body> </body> tags. The "head" contains nonprinted information that could prove useful in determining the kind of information on the webpage, and it might contain "title" and "meta" tags. Meta tags provide descriptions and keywords often used by search engines. The "body" contains those items that will be displayed in the browser, usually text and images. Here, "text" tags (e.g., <u>) manipulate how text is displayed. You have complete control over text font, color, and size, as well as paragraph and positioning through tags. One of the most used tags is <a>, the anchor tag, which generates a clickable link. This link might enable a jump to a particular location on the same page, a transfer to a new webpage, or even a connection to send an e-mail message.

As you likely know, HTML's simplicity is somewhat overshadowed by its potential. Naturally, tools for making page design quick and easy are always in vogue. While the basic tool for HTML is the text editor, advanced tools enable you to create without knowing much HTML. Free and moderately priced programs are available on the web.

Do you want to do more than just display information? Get ready to dive into the deep end. If your HTML page collects data or displays dynamic data, where will the data go to or come from? Basically, there must be other applications running that handle the data. The common gateway interface (CGI) uses an application that runs on the server and passes data back and forth to the browser, while scripting, like Java, can run directly in the browser. In many cases, you'll want your data to be collected and stored in some nonvolatile fashion, so you'll need to learn more than just HTML. Enter Run BASIC.

Why another web design tool? Imagine writing a BASIC program that interacts with any user in the world through a web browser. Your program serves its content, gathers user input, and manages the data without ever having to rely on any external application. All that's required is Run BASIC and an Internet connection. To demonstrate

	Files		
Entries	**users.txt**	**<Password>.txt (one file for each user)**	**<Password>State.txt (one file with a single entry for each user)**
1	Password1	Task 1	Present State
2	Password2	Task 2	
3	Password3	Task 3	
4	Password4	Task 4	
5		Task 5	
6		Task 6	
7		Task 7	
8		Task 8	
9		Task 9	

Table 2—The users.txt file is a list of authorized passwords. Each password becomes the base of log and state files for a user. Tasks are appended to the <Password>.txt file. The state for that user is saved in the <Password>State.txt file and updated whenever a link is clicked (data is updated).

Listing 1—Any time you click a button, the present state is saved to the <userspassword>State.txt file. This ensures that every new page served uses the latest information.

```
[SaveState]
   open "/public\";PW$;"State.txt" for output as #WriteState
      print #WriteState, PresentCN; ","; CompanyNumber; ","; PresentPN; ",";
      ProjectNumber; ","; PresentON; ","; OperationNumber; ","; StartDate;
      ",";StartTime
   close #WriteState
   return
```

its power, I developed an application that you might find useful.

PROJECT TIME WITH BASIC

Yes, it's project time—well, sort of. Let me explain. Before you can quote a price on a job for a client or employer, you must estimate the number of hours it will take to complete the task. For designers and programmers just starting out, this can be more difficult than the job itself. But those of us who have gone through this exercise numerous times have accumulated data to use as a reference. Accumulated data either confirms proper time estimations or shows where your estimates are off.

I don't necessary like keeping track of every minute spent on a project, but I do know that a careful post-project review can pay dividends. Gathering this information enables me to make estimates for future projects. And the more accurate the estimates, the less danger of blowing the budget or pricing myself out of a job. I maintain three simple lists: clients, projects, and operations. The client list could get quite large if it contained every project and every operation, so I use separate project lists that are linked back to the client list. All operations, on the other hand, are applicable to every project so this list is not linked to either of the first two.

The lists are simple text files that all begin with a "none" entry (see Table 1). You can add entries to any list at any time. Just don't edit the entries because they won't correspond to data saved previously. (This will be clear shortly.)

IT'S TIME

These lists become the seeds for the application I'm presenting. This application—"Project Time"—enables the user to easily start, end, or change any operation. Each log is referenced to a particular company's project. You end up with a text file containing appended log entries. Each entry consists of the following: Company Name, Project Name, Operation, Date, and Number of Seconds. Here's an example:

"Circuit Cellar,FTB241,Documentation, 3/8/2010,21600"

I could simply write this application using Liberty BASIC. It would be a quick task, but only accessible while I was at my PC. Here's where Run BASIC differs. Using Run BASIC, I can access this application from anywhere as long as I have Internet access (e.g., at home, at a client's, at the local library, or at my favorite Wi-Fi hot spot). Further, I can use any available browser, even my Droid cell phone! As for companies with multiple employees, each employee can produce their own log file from any location.

The beauty of Run BASIC is its simplicity. The basic programming is straightforward. Figure 1 shows the program flow. When Run BASIC starts, it checks its preferences to see if an application has been targeted to autorun. If there isn't one, it runs the editor where you can interactively write and run a program. This enables you to review your application, edit any syntax errors, and debug the program's operation. A free demo is available on the Run BASIC website (www.runbasic.com). When an application has been identified as autorun, it's "served" outside of the editor.

The application begins with initialization. The prepared text files, which I mentioned earlier, are read into three string arrays, starting with `Companies$(x)`, which holds all the company names. Since each company has different projects, a double element string array is used: `Projects$(x,y)`. The first element identifies the company. The second element identifies the company's projects. The company names read into the first array are used as project text file names, which make up the second array, `Projects$(x,y)`. Lastly, the Operations.txt file loads the `Operations$(x)` array.

With the arrays loaded with data, the application can serve up a page. Since I don't want everyone to be able to access this stuff, I created a password page to look for proper identification. This page `[password]` consists of a single box and a text link and consists of three commands. The `passwordbox` command produces the box. It is similar to a text box, but it displays dots instead of the entered character within the field. The command `link` creates a connection between an object (the password box in this case) and the action of clicking a link. I labeled the link `Accept`, which enables program flow to branch to `[passwordcheck]` whenever it is clicked. The `wait` command stops further execution. At this point, nothing more can happen, unless the link is clicked. When clicked, execution branches to `[passwordcheck]` where any user-entered password box data is checked. While I could have hard-coded the correct password into the application, I used a password text file. Using a text file has two advantages: it can be easily changed without affecting the application and it can contain a number of acceptable passwords so multiple users can be logged in. If the

Listing 2—Whenever Run BASIC serves up a webpage to a user, it uses data previously stored in the <userspassword>State.txt file.

```
open "/public\";PW$;"State.txt" for input as #ReadState
   line input #ReadState, item$
close #ReadState
```

entered data matches any password in the text file, I branch to [update]; otherwise, I branch back to [password].

Once cleared through security, the second page [update] is served (see Photo 1). A greeting appears at the top of the page followed by three column labels: Present Task, New Task, and Choose a New Task. Below these labels are three rows of two text boxes and a list box. Each row has its own label. The first row is for Company, the second row is for Project, and the third row is for Operation.

The column under New Task lists any task in process (being timed). Presently, with "none" in the Company text box, there is nothing in process, and so the text below indicates "No task to end." The second column under the heading New Task also has "none" listed in the Company text box. Nothing has been chosen, and so the text below indicates "No Task to Begin." The third column heading, "Choose New Task," has list boxes instead of text boxes. This is where you make your selections. The execution has again halted on the server waiting for you to do something.

Clicking on the Company list box shows the list of Companies in string array `Companies$(x)` to choose from. You choose by clicking on the desired entry and then clicking the Update Company link. Like the `link` command in the password screen, links are placed in this second page to trigger certain actions based on your selections in associated objects. The Update Company link sets a variable to the index number of the selected company. This value becomes the first element "x" of the `Projects$(x,y)` array allowing the Project list box to be filled with the appropriate projects for the chosen company.

Clicking on the Project list box generates a list of projects associated with the selected company in the string array `Projects$(x,y)`. You make a choice by clicking on the desired entry and then clicking the Update Project link. This link sets a variable to the index number of the chosen project. As choices are updated, the text boxes under New Task are filled. All that's left is to choose an operation from the Operation list box and click the Update Operation link. Each time an update link is clicked, page data is resent with the most recent information. Now the New Task column is complete and the message below indicates "New Task will Begin."

At this point, a new link, "Execute Task," will appear at the bottom, giving you a chance to begin (in this case) a task. When this link is clicked, you save some information:

```
StartDate=val(Date$("days"))
  'days since 1/1/1901
StartTime=time$("Seconds")
  'seconds since midnight
```

Copy the data from the New Task text boxes to the Present Task text boxes. Now you're processing—that is, timing a task. This task ends when you begin a new task by clicking Execute Task again. Should you choose "none" in the Company list box, the present task will end without starting a new task. Note that normal application usage would involve making choices, executing a task, and then leaving. You would then return to end

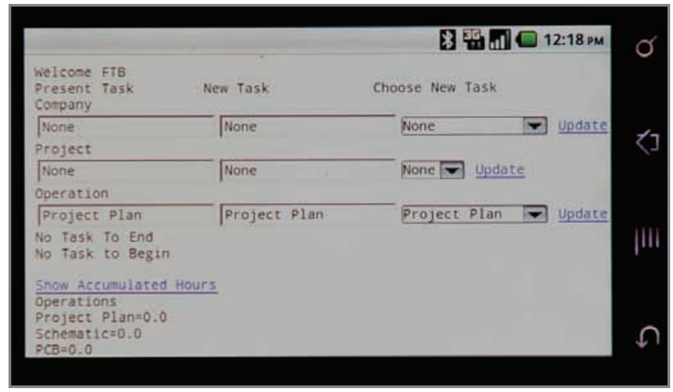


Photo 1—This application is available via any browser connected to the Internet. This application is being served to my Droid, which enables me to log tasks anywhere, anytime (except while driving).

the task or begin a new task, which would also end the present task.

LOGGING

As I previously mentioned, starting a task only sets the start date and start time. It doesn't log anything. I could have chosen to log the start and end times separately, but I thought it would simplify the logging data to make a single entry per task. So, when a task ends, the `StartDate` and `StartTime` are used to calculate the duration of the task in seconds since it was started (even if that should span days.) The log entries are strings of data that get appended to your personal text file (see Table 2). An option here might be to limit a day's time to 8 hours. But as many of you know, a normal work day is rarely limited to only 8 hours, so I chose to let a day's hours continue to accumulate beyond eight.

Any time you sign in to this application, a session begins anew. This means all the variables are initialized and any previous work is forgotten. But that defeats the purpose of this application altogether. You want any prior data remembered. This requires saving data "as you go" so that a new session can continue from where it left off. This is done by saving eight pieces of data any time an update link is clicked to the text file <user>State.txt. In any routine that processes a clicked link, I add a `gosub [SaveState]`, which saves what you see in Listing 1. Likewise, whenever the page is served, the state file is read (see Listing 2). And the string `item$` contains the comma-separated state data that's seeded back into the appropriate variables. Each user's data constantly remains current on each page display. Should a task remain in process, the total time for the task will continue to accumulate since the start time and start date are part of the updated state.

STATISTICS

As if this little program wasn't useful enough, I added a bonus. At the bottom of the page is an additional link: Show Accumulated Hours. With a company, project, and operation selected, the Show Accumulated Hours link will append statistics to the bottom of the screen that show all the operations and the total hours logged for each for the chosen Project.

If you choose "none" for the company, the Show Accumulated Hours link will append statistics to the bottom of the screen showing all operations and the total hours logged for all projects. Photo 1 shows the application on my Motorola Droid.

PERSONAL SERVER REQUIREMENTS

If you don't run your own webserver from your home (or workspace), your webpages are likely hosted by an ISP. If you have a domain name registered, it contains the IP that will get users to your website. While a Run BASIC application can be hosted, it really isn't that difficult to host it right from your home. The problem is that your home ISP probably assigns you a dynamic IP address, which means that it might change from time to time. Thus, the IP that works right now might not be yours tomorrow. So, how can you find your own server when you can't be sure of its IP? You need a dynamic domain name service (DDNS). A DDNS is a domain name service that allows for dynamic IP changes. You pick from a list of domain names (i.e., "myvnc.com") and then add your special prefix. I used my name "jlbachiochi" and so the domain name I end up with is "jlbachiochi.myvnc.com." At home, the PC that acts as the server runs a small application that queries its IP and periodically sends the latest IP to the DDNS. They dynamically update my IP so that anyone going to "jlbachiochi.myvnc.com" gets the right IP. Believe it or not, there are free DDNS hosts out there. An alternative would be to pay your ISP extra for a static IP address.

Run BASIC is configured to serve through the PC's local IP address port 8008. You can change this to port 80 (the normal server port) or forward port 80 on your modem/router to port 8008.

SURFACE SCRATCHES

I didn't spend a lot of time writing this application. I wanted to see what I could accomplish using only the basic commands available. I didn't attempt to add any kind of artistic spin on the look of the pages.

There are still many areas for me to explore using this new tool. It has a built-in commands for handling an SQLite database, table formatting to display arrays, sending e-mail through external SMTP servers, and web requests using `httppost$()` and `httpget$()`. Beyond the basic commands for font, size, color, and so on, you can use Java scripts and cascaded style sheets (CSS) with Run BASIC to add pizzazz to your pages.

If you want to try writing an application in Run BASIC, you can do this using a free copy of the program from www.runbasic.com. If you'd like to sign in to my application, point your browser to jlbachiochi.myvnc.com and use the password "FTB."

I foresee plenty of future applications for Run BASIC. It provides a simple tool to create some useful web applications, even if you aren't a web-authoring expert. ■

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.*

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2011/247.

RESOURCES

Run BASIC Forum, Shoptalk Systems, http://runbasic.proboards.com/index.cgi?board=server

Websitetips.com, "HTML (Section 4): Web Authoring/HTML Tools—Editors, Reference Charts, Validators, and More," 2008, http://websitetips.com/html/tools/#editors/.

SOURCE

Run BASIC
Shoptalk Systems | www.runbasic.com

CIRCUIT CELLAR
EMBEDDED DEVELOPMENT
DATA ACQUISITION

by Tom Cantrell (USA)

Tune In, Turn On

Power Via Radio Waves

The idea of wireless power delivery isn't new. At the dawn of the last century, electronics pioneer Nikola Tesla had *Wardenclyffe*, which was his vision of pervasive power. Today, Powercast is bringing their own IC-age version of Tesla's vision to life, delivering power via radio waves. Tune in and turn on!

A bit more than a year ago, I introduced *Circuit Cellar* readers to Powercast's mini-me version of Nikola Tesla's Wardenclyffe ("Power Pitcher," *Circuit Cellar* 232, 2009). Now Powercast has a bunch of new gear that is beaming away as I write this month's column. Before we take a closer look, let's review some of the basics.

The Powercast scheme is simple enough in principle. Just blast away with a radio transmitter and then harvest power at the receiving end. A low-frequency (i.e., audible) analogy

Photo 1—Tesla would not be surprised to see his Wardenclyffe vision of wireless power delivery finally brought to life, albeit on a smaller silicon scale.

would send amplified sound through the air and then harvest energy with a microphone.[1]

The foundation of the Powercast scheme—namely, the transmitter and harvester—were described in my earlier article. Now Powercast has integrated these with the other pieces of the puzzle to create a full-fledged wireless, and better yet battery-less, sensor network.

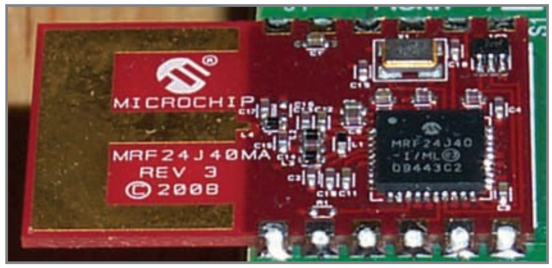


Photo 2—If you want all the benefits of full-featured IEEE 802.15.4 radio communication, but without all the costs and hassles of rolling your own, the Microchip MRF24J40MA module is just the ticket.

KIT(CHEN) SINK

Powercast calls their latest offering the "Lifetime Power Energy Harvesting Development Kit for Wireless Sensors." That's quite a mouthful; but in this case, it's well-deserved considering the full box of goodies I received (see Photo 1).

Big picture: you're looking at a complete system comprising an RF power transmitter, two RF-harvesting wireless sensor nodes, and a USB wireless access point. The gear utilizes Microchip Technology PIC24FK MCUs, so the kit even includes a Microchip PICKit 3 In-Circuit Debugger for embellishing the demo application or developing your own. The $1,250 price tag reinforces the fact this is a real "development" kit, not just "demo" gear. Even at that price, batteries are not included, but that's the point after all.

The action starts with the TX91501 Powercaster Transmitter. Powered from a 5-V at 1-A wall wart, it beams 3 W (a 1 W version also available) worth of 915-MHz ISM-band radio waves to power the sensor modules. The output is spread spectrum modulated to reduce emissions; but as with all things RF, you'll want to confirm there are no interference issues with other radios using the same band.

One notable upgrade to the Powercaster is that now it transmits an 8-bit factory-programmed ID providing the basis for new application features. For example, in a multiple-transmitter application, this could provide a measure of location awareness to the RF-powered devices.

How does the transmitter send both power and data? The answer is simple when you realize that data is, after all, simply modulated power. To send its ID, the Powercaster periodically (e.g., every 10 ms or so) amplitude-modulates the power output with the ID code. At the receiving end, the RF-powered device can take advantage of the fact the 2110 harvester has the ability to measure the received power (i.e., RSSI). It does take a bit of clever hardware (e.g., filter) and software to detect the 8-bit pattern in the shifting power level—all the more so in an environment with multiple Powercasters or dynamic interference. As an aside, I'm told a future upgrade will have the Powercaster transmit timing information as well. That would be a real boon, as it could allow the RF-powered devices to dispense with their own timing logic and, in multiple receiver apps, eliminate the chore of periodically calibrating all the individual clocks to maintain synchronization.

According to the datasheet, the Powercaster beam dispersion is 60 degrees, so a rough estimate of coverage would be a cone with base diameter equal to the distance from the transmitter. Furthermore, the beam is vertically polarized, which can impact the preferred antenna orientation at the receiving end. It's no surprise the transmitter works best with a clear line of sight, so wall or ceiling mounting is a good way to maximize power delivery.

In this era of ambulance-chasing lawyers and web videos of cell phones cooking eggs (right…), it should be noted the Powercaster complies with FCC RF radiation limits as long as a minimum separation (23 cm) from users is maintained and the Powercaster isn't co-located with other transmitters.

MiWi P2P Protocol Features	Program size	Data size
Minimum Stack	Less than 4 KB	100 + Rx Buffer Size + Tx Buffer Size + (9 * P2P Connections)
+Intra-PAN Communications	462 Bytes	---
+Sleep	185 Bytes	---
+Security (No Frame Freshness Check)	500 Bytes	48 Bytes
+Security (With Frame Freshness Check)	1488 Bytes	54 Bytes
+Active Scan	1070 Bytes	69 Bytes
+Energy Scan	752 Bytes	0 Bytes
+Indirect Messaging	950 Bytes	Indirect Message Size * TX Buffer Size
+Indirect Messaging With Broadcast Capability	1228 Bytes	Indirect Message Size * TX Buffer Size

Table 1—By sticking to basics, the MiWi P2P protocol has a tiny footprint that easily fits within the confines of low-cost MCUs.

TO BEE OR NOT TO BEE?

The story starts with the Powercaster. Now let's take a look where it ends—namely, with the appearance of data from the RF-powered wireless sensors on your PC display. The kit outsources that duty to Microchip, using one of their "16-bit nanoWatt XLP" development boards as a USB-connected wireless base station. The base station and sensor boards all utilize Microchip MRF24J40MA 802.15.4 radio modules.

The radio module itself is a cool gadget worthy of a closer look (see Photo 2). Based on Microchip's own 2.4-GHz radio chip, it's a very complete and cost-effective solution. By "complete" I mean the module integrates everything from the ones and zeros (i.e., SPI) to the built-in PCB antenna. By "very complete," I mean it even includes regulation certification, in the United States (FCC),

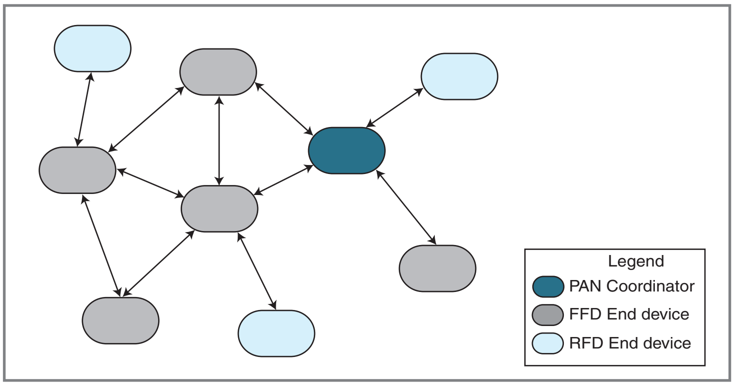


Figure 1—Beyond simple star networks, the "MiWi P2P" protocol supports more complex topologies. There's still a single PAN coordinator, but Full-Function Devices (FFDs) support multiple connections to other FFDs and Reduced Function Devices (RFDs).

Canada (IC), and Europe (ETSI) no less. And the module is less than $10 in singles, quite a reasonable price considering the RF design and regulatory hoops you have to jump through to roll your own.

IEEE 802.15.4 radios are historically associated with the ZigBee consortium. Now, putting it politely, over more than a few years, ZigBee has given new meaning to the term "feature creep." With the consortium now dabbling in everything from TV remotes to the smart grid—not to mention a variety of application-specific "Profiles" in between—the term "feature leap" is more like it.

To be fair, the ever new-and-improved ZigBee does some real heavy lifting with support for advanced features like dynamically adaptive multi-hop mesh networking and industrial-strength encryption. However, these features aren't necessary for simple "wire replacement" apps and come at a significant cost in terms of silicon (a fully loaded ZigBee stack is pushing 128 KB these days) and power consumption, not to mention the licensing and testing fees required for ZigBee certification. Thus, the trend finds 802.15.4 radio suppliers crafting their own mini-me wireless network protocols, and Microchip is no exception. Pop the hood of the Powercast gear and you'll find Microchip's MiWi P2P protocol running the show. The Microchip protocol cuts the fat to achieve two goals. The first is to reduce the size of the network stack itself, which it does admirably. As you can see in Table 1, depending on options, the MiWi P2P stack code size is roughly 4 to 8 KB, while data RAM is maybe 512 bytes or so, which is small enough to fit in a very low-cost MCU. The other goal is to reduce power consumption, even beyond the savings that comes with reduced stack complexity. MiWi P2P does this by assuming a network contains at least one Full-Function Device (FFD) that functions as the single "PAN Coordinator" and is active at all times (e.g., mains powered). In turn, all the Reduced-Function Devices (RFD) in the network can turn their radios off when idle, taking advantage of the "always on" Coordinator to buffer messages until sleeping devices wake up and check their inboxes.

A major reduction in complexity and power consumption is achieved by dispensing with ZigBee's multi-hop capability since devices no longer need to maintain routing tables or pass along data destined for others. In a simple star network, devices have only one connection, to the PAN Coordinator. As shown in Figure 1, the peer-to-peer (P2P) topology is also an option. Once again, there's only one PAN Coordinator, but the difference is that any other FFDs (i.e., besides the PAN Coordinator) in the network can establish single-hop connections to multiple devices (FFDs or RFDs). As in a star network, it's still the case that RFDs only connect to one device. But in a P2P configuration, that can be any FFD, not just the PAN Coordinator.

Although MiWi P2P tosses a lot of ZigBee baggage, since it's running an IEEE 802.15.4 radio, it still has to tote the load imposed by that spec. Most notable is the rather bloaty packet format (see Figure 2), all the more so due to the fact full 8-byte globally unique Extended Unique Identifier (EUI) addresses are used.

HARVEST TIME

So, we've got the Powercaster, a PC plug-in base station, radios and a network protocol. Now let's turn our attention to the wireless, and battery-less, sensors themselves. As you can see in Photo 3, they comprise three components: a baseboard, antenna, and a sensor board. The baseboard contains the 2110 harvester, which captures RF energy from the Powercaster, with two antennae options to choose from (omnidirectional, directional) included in the kit. As you might guess, and will see shortly, the trade-off between the antennae options is orientation flexibility versus gain. Plugging into, and powered by, the baseboard, the sensor board is a downsized version of the basestation (i.e., PIC24FK MCU and MRF24J40MA radio module) supplemented with temperature, humidity, and

Bytes	2	1	2	2/8	0/2	8	Variable	2
	Frame control	Sequence number	Destination PAN ID	Destination address	Source PAN ID	Source address	Pay load	Frame check sequence

Addressing fields

Figure 2—When running on an IEEE 802.15.4 radio, MiWi P2P has to meet demands (e.g., packet format) required by that standard. A shorter and simpler packet format could be an option if the protocol is ported to a simpler radio.

Photo 3—The Powercast wireless (power and data) sensor comprises the '2110 EVB energy-harvesting baseboard, an antenna (omnidirectional shown here) to capture the transmitted power, and the wireless sensor board comprising an MCU, IEEE 802.15.4 radio, and trio of sensors (temperature, humidity, and light).

light sensors plus screw terminals that enable you to connect your own sensor of interest.

The baseboard is available separately (part number P2110-EVB) as a handy tool to prototype your own designs. Let's take a tour using the silkscreen (see Figure 3) as a roadmap. On the upper left is the P2110 harvester (U1) and its antenna connector (J1). The P2110 output voltage comes from the factory set at 3.3 V, but provision is made for adjustment (i.e., P2110 VSET pin) by populating nearby resistors (R2/R4, R5/R6) and switch S1. To the right of S1, S2 selects the destination for the harvested energy as either an LED (D1), current-measuring test points (VOUT-to-LED or VOUT-to-VCC), or your own circuits (VCC). For storage, the harvested energy is directed to capacitor C3 (1,000 µF), C4 (not populated), or the supersized supercap C5 (50,000 µF) by installing the appropriate jumper at JP1.

Switch S3 controls the P2110 DSET pin. When DSET is low (or open since it has an internal pull-down), the harvester output is directed to the VOUT pin as expected. But when DSET is driven high, the harvester output is passed through an internal sense resistor with the resulting voltage indicating the received power level (i.e., RSSI) available at the DOUT pin. So, for normal harvesting operation, S3 is in the OFF position, in which case the P2110 DSET is controlled by the DSET signal in the prototyping area. Setting S3 to EXT connects the DSET pin to the nearby DSET EXT test point. Alternatively, setting S3 to VOUT brings the sense voltage to that same test point for direct measurement.

On the right side of the board, switch S4 lets you choose alternative power options including an external power supply or battery (via test points GND and BATT) or an (optional, not populated) second supercap (C6). All the relevant signals and power supplies are brought to the center prototyping area with a header for connecting your RF-powered application, in this case the wireless sensor module. In addition to the signals mentioned so far, two others worthy of note are INT and RESET which connect to the P2110 pins of the same name. Once it detects sufficient charge has gathered on the storage capacitor, the P2110 turns on the VOUT supply and signals it's ready to go by driving INT high. After the application completes its task, it can turn off VOUT by driving the RESET input high to preserve charge accumulated on the storage capacitor. Otherwise, power will be provided until the storage capacitor runs out of gas at which point the P2110 shuts off VOUT automatically.

TEST DRIVE

Now that you've seen the pieces of the puzzle let's put them together. The transmitter is easy enough, just hang it on the wall—noting orientation (i.e., directionality and polarization)—and plug in its wall wart. After a couple of seconds, a green LED lights up. Do note that if you just set the transmitter on a table or in close proximity to an

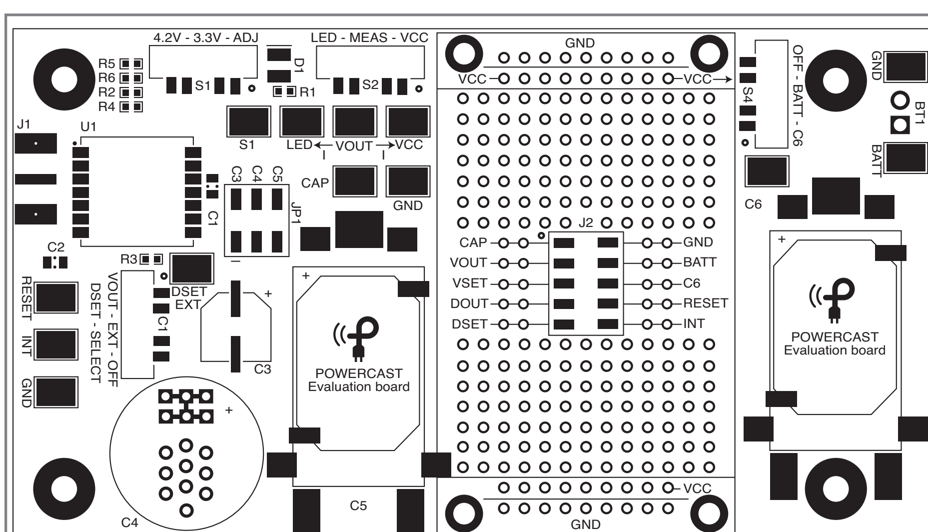


Figure 3—The '2110 EVB is a handy way to prototype an RF energy-harvesting application.

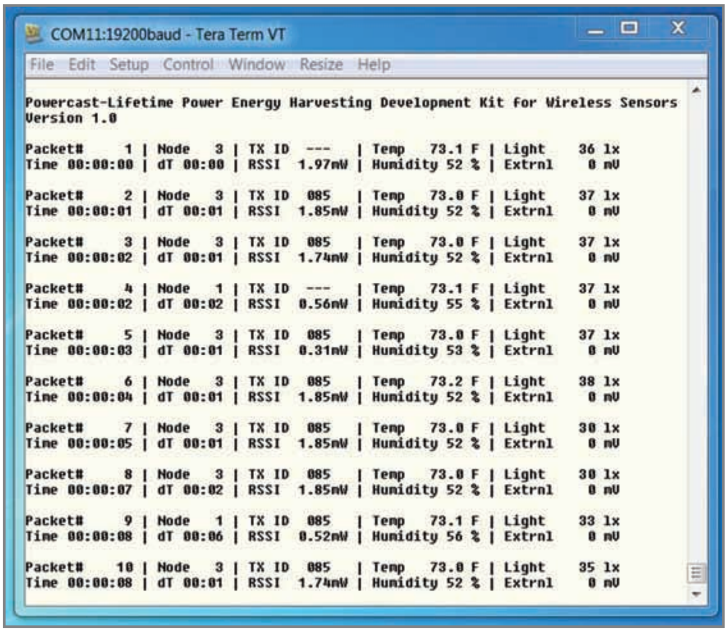


Photo 4—Here's a screenshot of the Lifetime Energy Kit in action. You can see the higher gain of the directional antenna results in more frequent packet delivery.

obstruction, it may automatically shut down (antennae impedance mismatch) in which case the LED will blink red.

Next, get the base station up and running by plugging a radio module into the 'XLP dev board and connecting the lashup to your PC with a USB cable. According to the pre-release documentation, Windows 7 machines are supposed to automatically install the USB driver, but that didn't happen with my Win7 PC. Instead, I used a provided link to download a driver for Windows XP, which worked fine. To confirm all is well, you can look for the virtual COM port in the Device Manager, and then bring up a terminal emulator (19,200 N-8-1) to connect. Just to make sure all is well, hit the MCLR* (i.e., reset) switch on the 'XLP board to confirm you see the demo firmware sign-on banner.

Now connect an antenna to each of the P2110 EVBs. Before you plug in their radios, I suggest you confirm the RF power harvesting is working, which you can do by setting the switch (S2 = "LED") and capacitor jumper (C3) to pulse power to the LED on the EVB. As you change the position and orientation of the EVB and its antennae, the difference in harvested energy is readily apparent in how fast the LED blinks. It's your own modern version of Tesla's original light-bulb-in-hand parlor trick. Once you've confirmed RF harvesting is working, remember to move the switch and jumper back to their normal operation positions (i.e., S2 = "VCC," C5).

Finally, plug a sensor board into each of the P2100 EVBs. Make sure you set the DIP switch on the sensor boards to give each node a unique ID (i.e., 0 to 7) and install the jumpers enabling the temperature, humidity and light level sensors (and optionally a jumper enabling the external voltage input you attach via the screw terminals).

With proper antenna orientation and in reasonably close proximity to the transmitter (e.g., a few feet, but not less than 2' lest you overdrive the harvester), it takes only a minute or so after turning on the Powercaster for the harvester to charge up the supercap, at which point packets from the sensor boards will start streaming into your PC (see Photo 4). As further indication, a red LED on the basestation blinks when packets are received.

I personally would have added a similar radio activity LED to the sensor boards, something I advise for any wireless application. I understand the rationale of not wanting to waste valuable harvested power, but a tiny LED flashed very briefly doesn't really consume that much energy. And if it's still a concern, a "debug LED" jumper could be added or, a more high-tech solution, allow the user to enable/disable the LED in software with a radio command from the PC via the basestation. Anyway, the absence of a visible indicator on the sensor board is all the more reason to perform the pre-op checks (RF harvesting, COM port connection) mentioned earlier.

GAIN MAINLY ON THE PLANE

Taking a closer look at the earlier screenshot yields some basic insights. In this setup, both nodes (#1 and #3) were equally positioned at about 10' from the Powercaster. I held the nodes in the air since setting them on the bench cut the power ouput significantly. Node #1 had the omnidirectional antennae; node #3 had the directional one. As you can see in the screenshot, the latter delivered roughly three-to-four times the gain as measured by the relative frequency of packets and the RSSI readings. You also notice the RSSI jumps around more than a bit, so your application might want to do some averaging rather than rely on an individual reading that's subject to instantaneous interference.

On the one hand, with just a few milliwatts at best on tap, it's clear you'll have to be careful not to bite off more than the harvester can chew. And it's also clear, as generally the case with energy harvesting, that overall efficiency is very low. Consider that it takes a 15-W (5 V at 3 A) wall-wart driving the Powercaster to get 2 mW out of the harvester 10' away.

But it's also true that a few milliwatts goes amazingly far these days. According to the datasheet, the sensor board consumes a mere 50 mW or so (e.g., 15 mA at 3.3 V) for 10 ms to do its thing (i.e., sample sensors and transmit the packet). So, as you can see in Figure 4, it only takes a 2-mW harvest to transmit a packet every second (results consistent with those in the screenshot). Indeed, the figure shows you may be able to do something useful (e.g., packet per minute) with as little as a 0.1-mW harvest.

You can back of the envelope a likely range using the inverse square law. For instance, if the high-gain antenna delivers 2 mW at 10', it should be good for about 0.125 mW at 40'. Just remember these are all favorable "line-of-sight" scenarios. Sealing a node inside a cardboard box (another

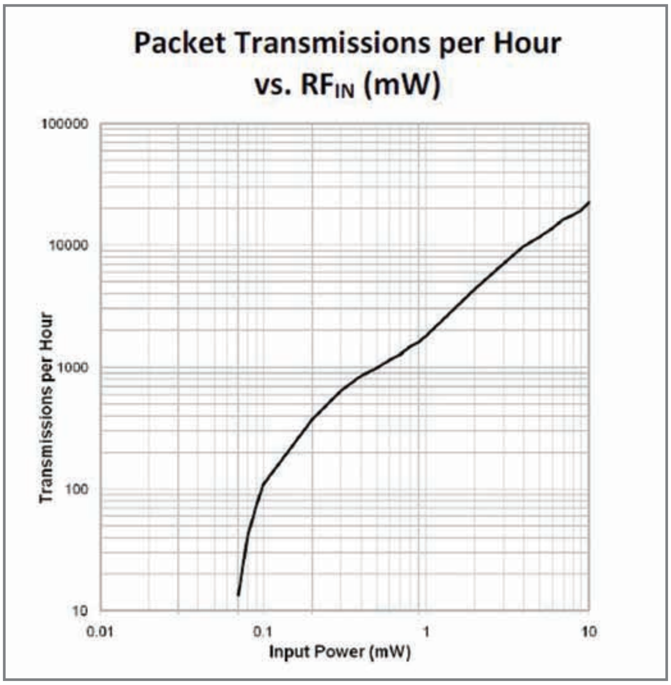


Figure 4—Powercast characterizes the performance of their RF-powered sensor boards in terms of packet frequency versus harvested power. Thanks to low-power silicon, harvesting even a fraction of a milliwatt yields enough power to support real-world applications.

cool parlor trick) cut the received power in half, but that was still enough to be useable. However, blocking the Powercaster with my body (Is it getting warm in here?) seemingly killed the nodes (i.e., I got bored after standing there for a couple of minutes waiting for a packet).

One gotcha I noticed is that every now and then (maybe 10% of the packets) the transmitter ID (TX ID = 085) is missing. It could be a bug, but my guess is it's due to instantaneous interference, the fact the ID transmit time is asynchronous to the node timing, or a combination of both. Here again you should include a bit of glitch filtering in your app software so it doesn't panic over a missing reading. And although sometimes "missing," I never saw an "incorrect" TX ID, which would arguably be a more problematic glitch.

The directionality of the transmitter and different antennae is easily demonstrated by moving the nodes around. As expected, the directional antenna is rather sensitive to aiming, with a ±45° mismatch dropping harvested power by nearly half. By contrast, the omnidirectional antennae lived up to its name with power output relatively unchanged through a complete rotation.

It's no surprise that range and, for the directional antenna, aim have a big effect on the harvest. Perhaps the most interesting result of my experiments was also discovering the importance of the "vertical polarization" spec. Harvested power falls off nearly linearly with the mismatch in vertical alignment between the Powercaster and antenna. In other words, at 45° relative "roll" between the transmitter and receiver the harvest is approximately one-half. In the worst case (i.e., 90° roll), the nodes went off the air. Note this phenomenon was the same for both antenna types. So, keep in mind that for applications in which the vertical alignment may change, two transmitters (e.g., one horizontal, one vertical) may be required to guarantee adequate coverage.

SKY HIGH

Maybe there will come a time when a combination of improving harvester efficiency and ever-lower-power silicon will make "plugging into the sky" a routine proposition. That time isn't yet at hand, but Powercast is demonstrating that the prospect of RF energy harvesting is definitely becoming viable for specialized applications. Sure, it's a baby step, but it's the baby step that every technology takes as it starts the journey from lab to widespread adoption.

That's the lesson of the IC age. When it comes to even the bluest of blue-sky innovations, it's simply a matter of when, not if.

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

REFERENCE

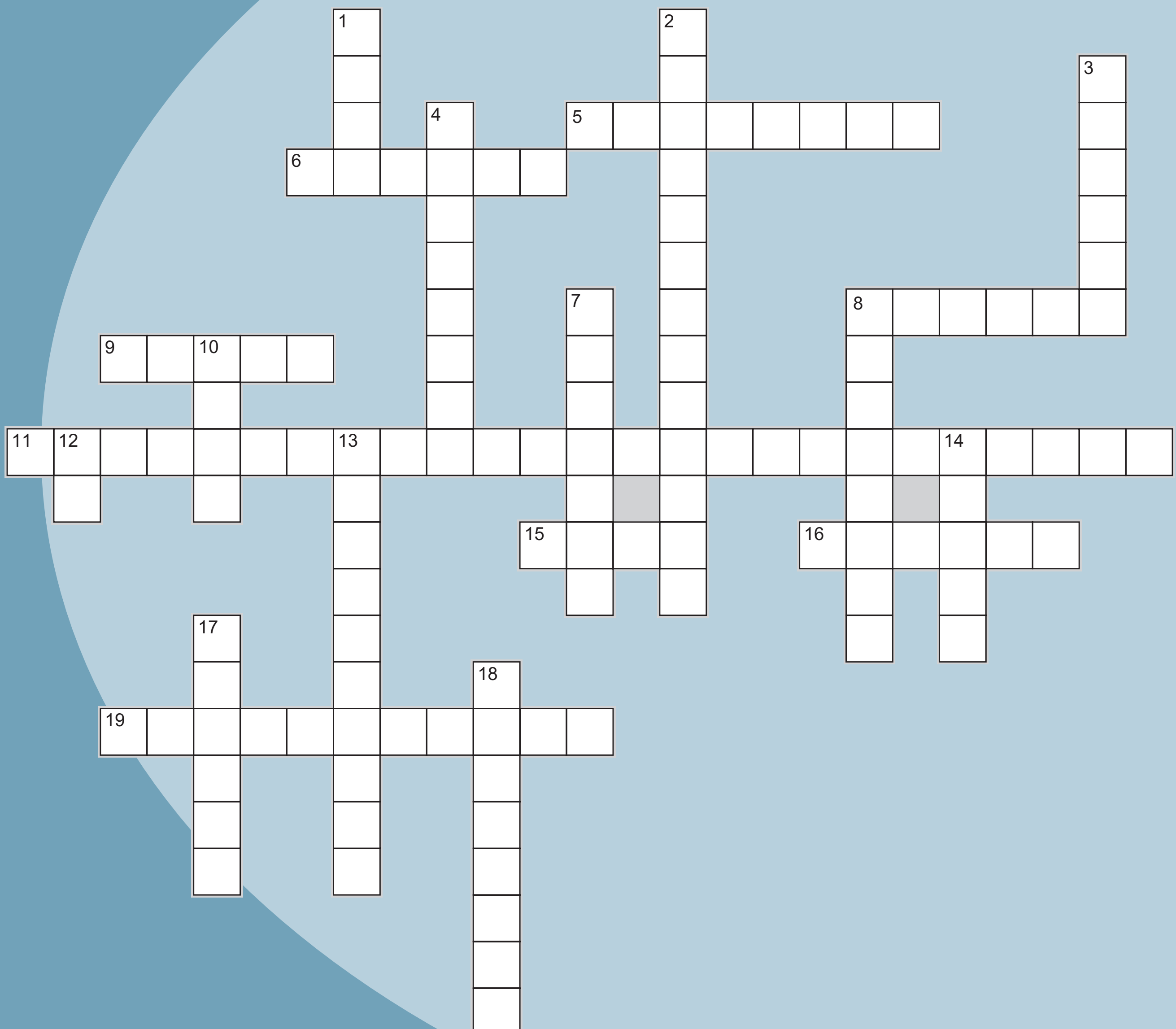
[1] M. Balasubramaniam, "Energy Harvesting System, Apparatus and Method," United States Patent #7116036, October 3, www.freepatentsonline.com/7116036.pdf.

SOURCES

MRF24J40MA IEEE 802.15.4 RF Transceiver module
Microchip Technology | www.microchip.com

Lifetime Power RF Energy Harvesting Development Kit for Wireless Sensors
Powercast Corp. | www.powercastco.com

CROSSWORD



Down

1. Before NASA
2. EVKIT [two words]
3. A cover that protects users from high voltages
4. Memory requiring power
7. Piping for wires
8. Orientation with respect to motion
10. Drop in dB
12. Receive
13. Bus, Star, Ring, P2P
14. 1 square mile = 640 ____
17. Earth wire
18. Tungsten wire through which electricity passes

Across

5. Tx
6. Radar dome; can have shapes such as planar or spherical
8. Add to end
9. What you get when you combine balance and unbalance
11. TX ID [two words]
15. IEEE 802.11
16. Timing XTAL
19. Wave transmission

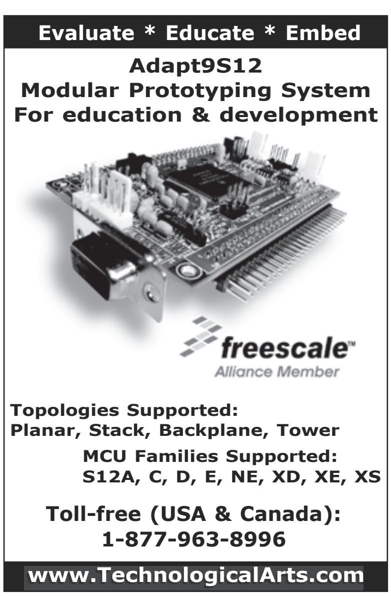
The answers will be available in the next issue and at www.circuitcellar.com/crossword.

IDEA BOX

THE DIRECTORY OF PRODUCTS AND SERVICES

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

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



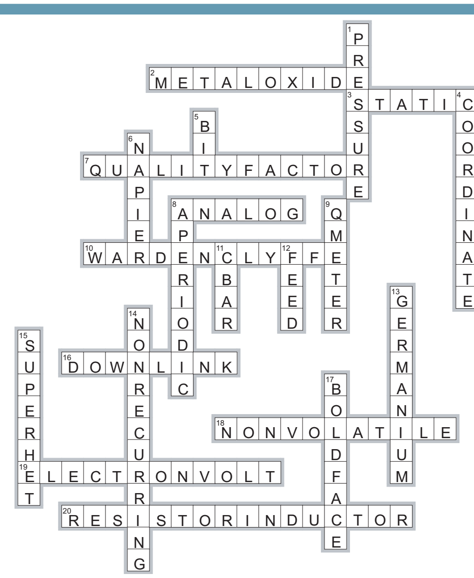
CROSSWORD ANSWERS from Issue 246

Down

1. **PRESSURE**—F/A
4. **COORDINATE**—x, y
5. **BIT**—Binary + digit
6. **NAPIER**—Natural logarithms; "Marvelous Merchiston" (1150-1617)
8. **APERIODIC**—Irregular intervals
9. **QMETER**—Measures a circuit's Q
11. **CBAR**—0.01 bar
12. **FEED**—Line that transmits an RF signal to an antenna
13. **GERMANIUM**—Kilby; Ge
14. **NONRECURRING**—One-time
15. **SUPERHET**—Short for superheterodyne
17. **BOLDFACE**—<B> ... </B>

Across

2. **METALOXIDE**—MO varistor [two words]
3. **STATIC**—Unchanging IP address
7. **QUALITYFACTOR**—A circuit's "Q" [two words]
8. **ANALOG**—Constant signal processing
10. **WARDENCLYFFE**—Tesla Tower
16. **DOWNLINK**—Satellite-to-earth connection
18. **NONVOLATILE**—Memory that doesn't require power
19. **ELECTRONVOLT**—eV [two words]
20. **RESISTORINDUCTOR**—RL [two words]



INDEX OF ADVERTISERS

PREVIEW of March Issue 248

Theme: Robotics

ROV Design: Remotely Operated Vehicle Development

Panning Control: Design a Digital Indexing Panoramic Tripod Head

CNC Router Design: A Look at Computer-Controlled Milling Machines and Routers

Embedded DSP for Lighting

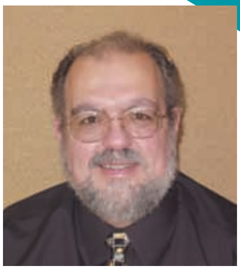
THE CONSUMMATE ENGINEER **The Super-Regenerative Receiver**

LESSONS FROM THE TRENCHES **The Project-Ready Designer:** A Refresher on Project-Essential Concepts

FROM THE BENCH **Direct Line Connection:** When You Don't Require Line Isolation

SILICON UPDATE **Time Traveler:** Embedded Timekeeping and More

PRIORITY INTERRUPT



by Steve Ciarcia, Founder and Editorial Director

Patently Insane

OK, this is a rant. Ever since I wrote a recent editorial called "Winging It" about starting a business, I've received a number of comments from readers. The whole (unspoken) premise of the editorial was to hammer home that there are two basic ways to learn things in life: either by trial and error or someone tells you what you need to know. I was also trying to tell readers what I thought of patents.

One series of e-mails with an engineer in Australia makes me think that I might need a bigger hammer. Apparently, more people like trial and error than I realized. While I didn't ask, I suspect that this engineer has been around doing all this technical stuff as long as I have, but we have completely different opinions when the subject involves inventions. He's a traditionalist and I'm a pessimist. ;-)

Part of our conversation involved him sending me a PowerPoint demo about his latest networked security system product. Besides a top-notch demo, it appeared to be an idea worthy of further investment and production. In fact, after reviewing his website, I can say that most of his other designs are equally innovative. Unfortunately, at this point, he was stuck (as I suspect that many of you might be) on how to capitalize on his inventions. Being a traditionalist and thinking that the patent filing date is the holy grail, he said he had obtained a provisional patent (a filing, that is) so he could better convince potential investors. I just shook my head.

Except for mega-rich companies trying to burn out their competition's resources or the few truly unique ideas left, I think today's patent system is a worthless concept for individual inventors and entrepreneurs like us. Without having the revenue and resources of a Fortune 500 company, we'll always end up on the short end of any litigation. In my opinion, involvement with the patent system is an exercise in frustration, expense, and insanity. Reality is this:

> *A patent is merely a right to sue someone. It does not protect your idea or keep anyone from using it. You have an obligation to fight infringers or you lose your patent. Typical (1st round) litigation averages $500,000. You don't "win" a patent fight. Since the primary consequence of patent litigation is to burn financial resources, you have merely demonstrated a willingness to spend the most.*
>
> *All patents can be invalidated. Considering the millions of people in the world and the millions of places they may have posted or published ideas and designs, it can probably be shown that with the evidence of those other sources, "Your idea could be reasonably obvious to any practitioner in the field," and therefore public domain or not patentable. It just gets down to the expense of finding that info.*
>
> *Thinking that there are large companies out there looking to buy your ideas or market your inventions (and still pay you) is an urban myth (albeit a techie one). "Not invented here" (NIH) is their cardinal rule.*
>
> *If you have an idea worth manufacturing, don't mess around. Either do it or publish it. The market life of technology products today is a year or less. Don't miss the window of opportunity. It takes two to three years to get a patent (and then what?).*
>
> *Nobody voluntarily pays royalties! You will be ripped off (see item 1 again).*

Proprietary ownership of ideas via patents and copyrights probably made complete sense in 1790, but market share and windows of opportunity ultimately rule in the long run today. Consider the early days of Mac OS and Windows. Clearly, Mac was better technically, but Apple used the patent laws to keep it proprietary and refused third-party applications and licensing (still sounds like them today). ;-) Microsoft, on the other hand, promoted third-party applications and built market share. When Apple finally got the message, the OS market belonged to Microsoft.

Patent law is based on the assumption that inventions are unique and distinct and can be individually owned (albeit by a corporation). Except for the few that are truly exceptional, I disagree. Innovation isn't only the result of discovery by individuals, but rather it's the cumulative enhancements, tweaks, and improvements of different people and different evolutions in design on a core idea. It's time to realize that there are also lots of business elements more important than proprietary patent protection. Stop living the patent myth.

steve.ciarcia@circuitcellar.com

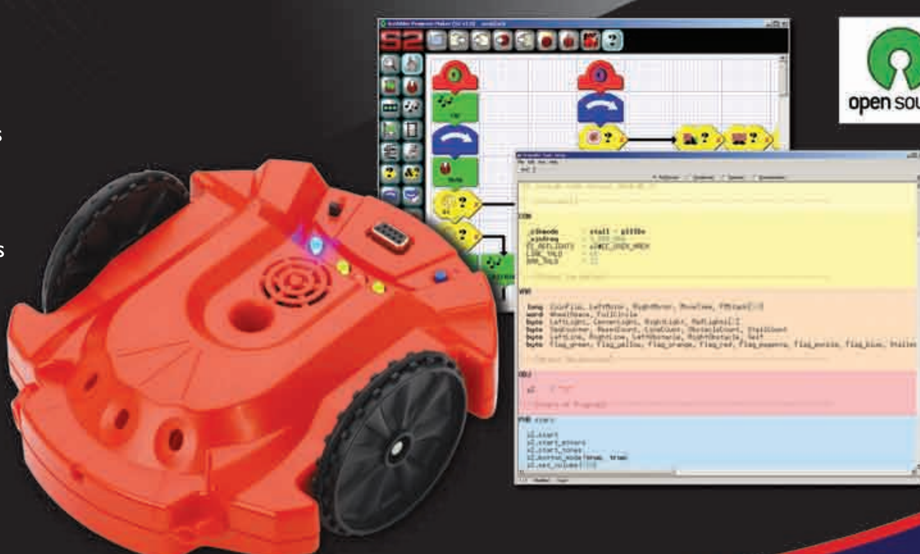
S2 ROBOT

The Scribbler 2 (S2) is a fully programmable, open-source and user hackable robot. S2 features the Propeller multi-core processor for a cutting-edge robotics experience.

The fully-assembled S2 robot (#28136; $129.99) is suitable for a variety of programming skills. The S2 robot arrives pre-programmed with eight demo modes, including light-seeking, object detection, object avoidance, line-following, and art. Use the Graphical User Interface (S2 GUI) tile-based programming tools, or modify the Propeller source code in our easy to use Spin language. Third-party tools allow programming on a Mac and in Linux, using Spin, PropBASIC, and C programming languages. *Six AA batteries not included. S2 is also available with a Serial to USB adapter and cable (#28336; $139.99).*

FEATURES:

- 3 light sensors
- 2 obstacle avoidance sensors
- 2 line following sensors
- 2 independent DC wheel motors
- Wheel encoders
- Stall sensor for wheel motors
- Pen port for drawing
- Speaker makes full range of notes
- Bi-color LEDs
- Programmable indicator lights
- Microphone for detecting tones from other S2 robots
- Hacker port to connect to sensors, RF devices, & servos



open source

Order the **S2 Robot** (#28136; $129.99) at www.parallax.com/go/S2 or call our Sales Department toll-free at 888-512-1024 (Mon-Fri, 7 a.m. - 5 p.m., PST).

Friendly microcontrollers, legendary resources.™

PARALLAX INC

www.parallax.com

"ParallaxInc" on Twitter, Facebook, and YouTube