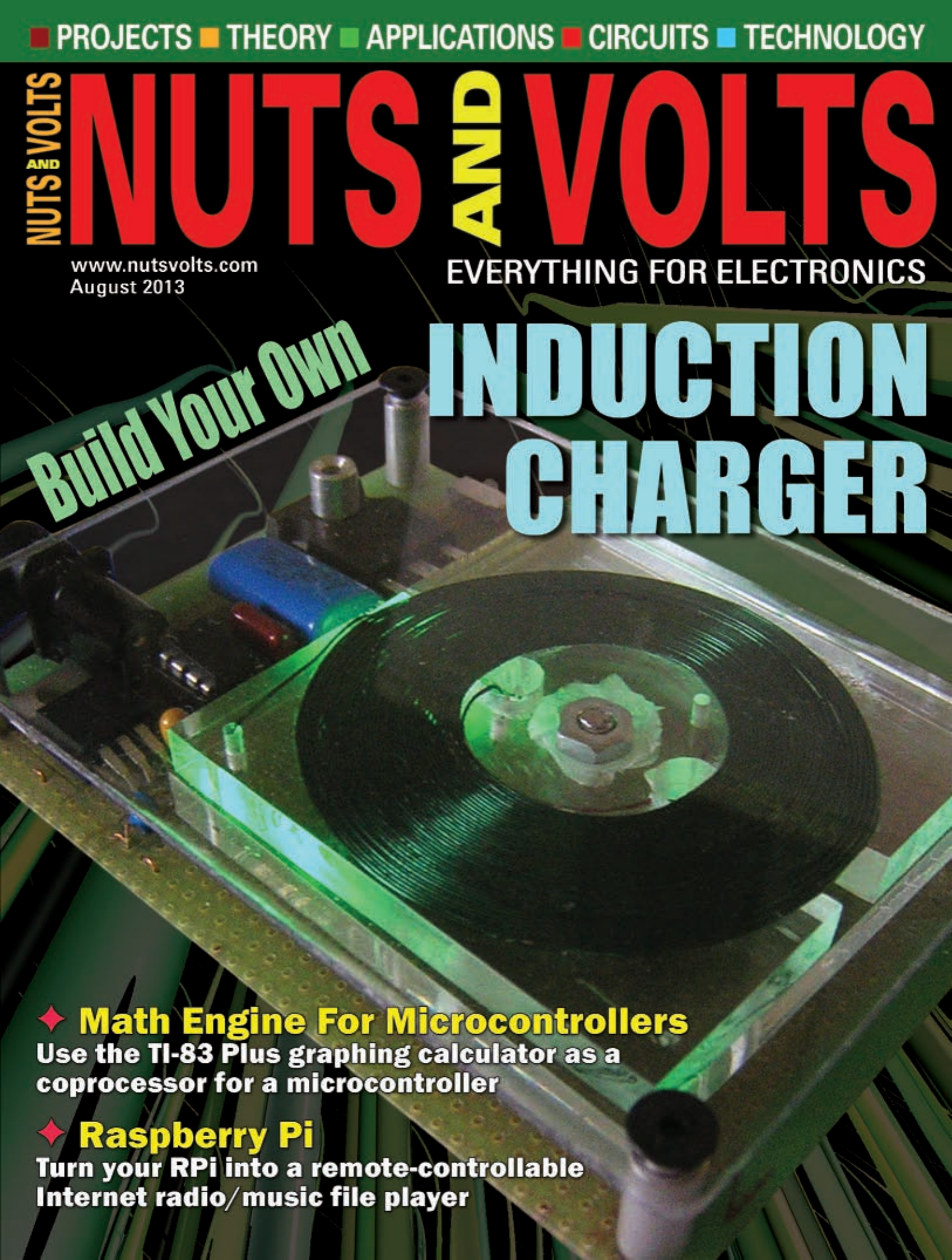
PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

NUTS AND VOLTS

www.nutsvolts.com
August 2013

EVERYTHING FOR ELECTRONICS

Build Your Own INDUCTION CHARGER



✦ Math Engine For Microcontrollers

Use the TI-83 Plus graphing calculator as a coprocessor for a microcontroller

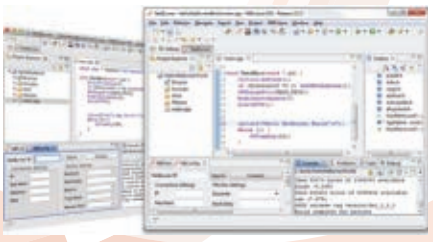
✦ Raspberry Pi

Turn your RPi into a remote-controllable Internet radio/music file player

Ethernet Core Modules with
High-Performance Connectivity Options

- **MOD5270**
  147.5 MHz processor with 512KB Flash & 8MB RAM · 47 GPIO
  3 UARTs · I²C · SPI
- **MOD5234**
  147.5 MHz processor with 2MB flash & 8MB RAM · 49 GPIO · 3 UARTs
  I²C · SPI · CAN · eTPU (for I/O handling, serial communications, motor/timing/engine control applications)
- **MOD54415**
  250 MHz processor with 32MB flash & 64MB RAM · 44 GPIO · 8 UARTs
  5 I²C · 3 SPI · 2 CAN · SSI · 8 ADC · 2 DAC · 8 PWM · 1-Wire® interface
- **NANO54415**
  250 MHz processor with 8MB flash & 64MB RAM · 30 GPIO · 8 UARTs
  4 I²C · 3 SPI · 2 CAN · SSI · 6 ADC · 2 DAC · 8 PWM · 1-Wire® interface

Add Ethernet connectivity to an existing product, or use it as your product's core processor



The goal: Control, configure, or monitor a device using Ethernet

The method: Create and deploy applications from your Mac or Windows PC. Get hands-on familiarity with the NetBurner platform by studying, building, and modifying source code examples.

The result: Access device from the Internet or a local area network (LAN)

The NetBurner Ethernet Core Module is a device containing everything needed for design engineers to add network control and to monitor a company's communications assets. For a very low price point, this module solves the problem of network-enabling devices with 10/100 Ethernet, including those requiring digital, analog and serial control.

MOD5270-100IR.....$69 (qty. 100)
MOD5234-100IR.....$99 (qty. 100)
MOD54415-100IR.....$89 (qty. 100)
NANO54415-200IR...$69 (qty. 100)

NNDK-MOD5270LC-KIT.....$99
NNDK-MOD5234LC-KIT.....$249
NNDK-MOD54415LC-KIT.....$129
NNDK-NANO54415-KIT.....$99

NetBurner Development Kits are available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kits include all the hardware and software you need to build your embedded application.

- **For additional information please visit**
  http://www.netburner.com/kits

NetBurner
Networking in One Day!

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

freescale™
Alliance Member

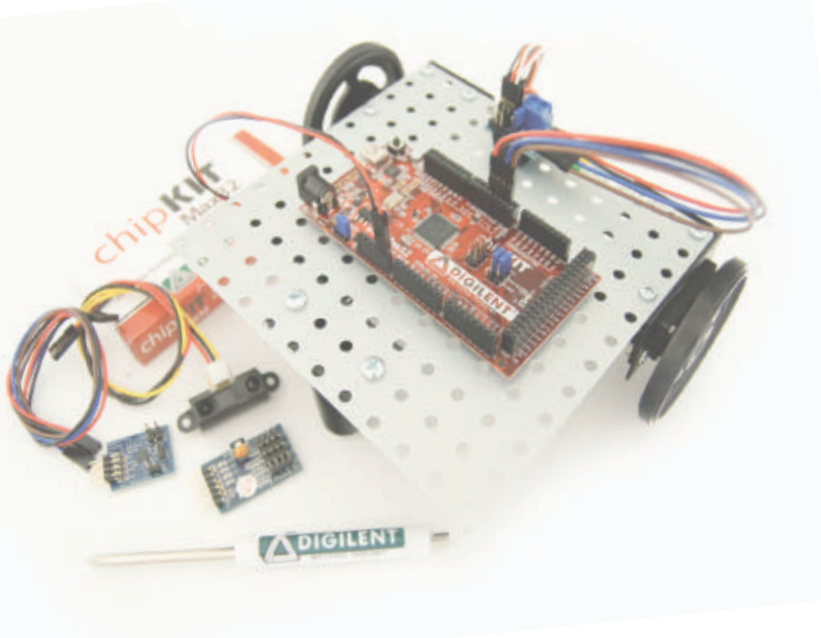
EasyPIC v7 with **click™ boards** makes perfect match for any project you are working on. Just place your click board into the mikroBUS™ host socket and it's ready to work straight away. Adding **new functionality** to your development board was never so easy!

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

www.mikroe.com/click/

GET IT NOW
www.mikroe.com/easypic/

ADVANCED 32-BIT POWER FOR THE ARDUINO™ COMMUNITY



chipKIT™ development boards are the first 32-bit microcontroller-based platforms that are compatible with many existing Arduino™ code examples, reference materials and other resources. They can be programmed using an environment based on the original Arduino™ IDE modified to support PIC32.

- Pin-out compatibility with many existing Arduino™ shields that can operate at 3.3V
- Lower price-point at four times the performance than existing solutions
- Advanced capabilities including integrated USB (Device/Host, OTG) & integrated Ethernet

chipKIT™ Max32

$49.50

- Microchip® PIC32MX795F512 processor
  - 80 Mhz 32-bit MIPS
  - 512K Flash, 128K RAM
  - USB 2.0 OTG controller
  - 10/100 Ethernet MAC
  - Dual CAN controllers
- Can also be programmed in Microchip's MPLAB
- Arduino™ "Mega" form factor
- Compatible with many Arduino™ shields
- 83 available I/O
- User LED

chipKIT™ uC32

$34.99

- Microchip® PIC32MX340F512H microcontroller
  - 80 Mhz 32-bit MIPS
  - 512K Flash, 32K SRAM
- Can also be programmed in Microchip's MPLAB
- Arduino™ "Uno" form factor
- Compatible with many Arduino™ shields
- 42 available I/O
- Two user LEDs

chipKIT™ Uno32

$26.95

- Microchip® PIC32MX320F128 processor
  - 80 Mhz 32-bit MIPS
  - 128K Flash, 16K SRAM
- Can also be programmed in Microchip's MPLAB
- Arduino™ "Uno" form factor
- Compatible with many Arduino™ shields
- 42 available I/O
- User LED

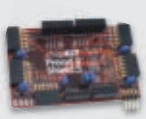
Network Shield™ $54.99

- **Usable with the Max32**
- 10/100 Ethernet
- USB Host, Device, OTG
- Dual CAN transceivers
- Dual I2C™ connectors
- 256kbit I2C™ EEPROM
- 32.768 Khz oscillator for RTCC

Basic I/O Shield™ $37.99

- **Usable with the Max32, Uno32, & uC32**
- 128x32 OLED Graphic Display
- Digital temperature sensor
- 256kbit EEPROM
- 4 switches, 4 push buttons, 8 LEDs
- 4 Open drain transistor outputs
- Analog potentiometer

Pmod Shield Uno™ $21.99



- **Usable with the Uno32**
- Five 2x6-pin Digilent Pmod connectors
- One 6-pin SPI connector
- One I²C daisy chain connector

WiFi Shield™ $49.99

- **Usable with the Max32, Uno32, & uC32**
- IEEE 802.11b-compliant RF transceiver
- 1 and 2Mbps data rates
- IEEE 802.11b/g/n-compatible
- Integrated PCB antenna
- Micro SD card connector
- Four LEDs

chipKIT™ PGM

$26.99

The chipKIT PGM is a simple, low cost, module that supports in-system programming and debugging of applications written for Microchip PIC based microcontroller boards such as the chipKIT™ and Cerebot boards. It is designed to work with the MPLAB® and MPLAB® X development environments available from Microchip.

The chipKIT PGM can also be used to enable in-system debugging of sketches developed using the Arduino™ compatible MPIDE development environment.

chipKIT™

DIGILENT®
BEYOND THEORY

www.digilentinc.com/chipkit

Pololu
Robotics & Electronics

Finding the right parts for your robot can be difficult, but you also don't want to spend all your time reinventing the wheel (or motor controller). That's where we come in: Pololu has the unique products — from actuators to wireless modules — that can help you take your robot from idea to reality.

38 kHz IR Proximity Sensor

ITEM #2460

$5.95

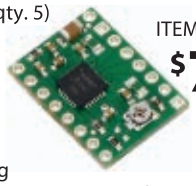
- Typical sensing range up to 24" (60 cm)
- Fixed-gain modulated IR detector
- Small size (0.4" × 0.6")

Stepper Motor Drivers

ITEM #2134

$7.95 (qty. 5)

- DRV8834 driver
- 4-layer PCB
- 1/32 microstepping
- Low voltage (2.5 V - 10.8 V)



ITEM #1182

$7.95 (qty. 5)

- A4988 driver
- 1/16 microstepping

Baby Orangutan B-328 Robot Controller

ITEM #1220

$19.95

Complete control solution for small robots

- Integrated dual motor drivers
- User-programmable ATmega328P AVR
- Small 24-pin DIP form factor (1.2" × 0.7")

170-Point Breadboard

ITEM #1488

$2.95

Other colors available

- 17 rows of tie points for up to two 14- or 16-pin DIP ICs
- Tabs allow multiple units to be connected for larger projects

Invenscience i00600 Torxis Servo

ITEM #1390

$289

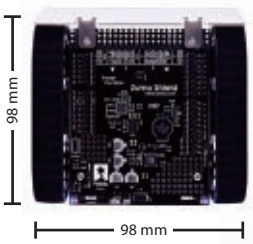
Ultra-high-torque monster servo

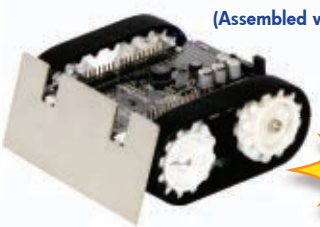
@12V:

- 1 sec/60°
- 1600 oz·in

Zumo Robot for Arduino

(Assembled with 75:1 HP Motors)





ITEM #2506

$99.95

Build and customize your own Zumo!

Arduino-controllable tracked robot small enough for mini-sumo (less than 10 cm × 10 cm) and flexible enough for you to make it your own. If you build it, it will push!

NEMA 23 Hybrid Stepper Motor

ITEM #1477

49.95

Other sizes available

- 1 A/phase @ 8.6 V
- 1.8° step angle
- 190 oz·in
- 56.4 mm × 56.4 mm × 76 mm

Ball Caster with 3/4" Metal Ball

ITEM #955

$2.99

Other sizes available

Our ball casters make a great third contact point for small two-wheeled robots.

Dual MC33926 Motor Driver Shield for Arduino

ITEM #2503

$29.95

Control two brushed DC motors with an Arduino or Arduino-compatible board. Operates between 5 V and 28 V and can deliver a continuous 3 A per motor.

Step-Up/Step-Down Voltage Regulator S10V4F5

ITEM #2121

$4.49

9V and 12V versions available

- 2.5 V to 18 V input
- Fixed 5 V output
- 70% to 80% efficiency

Universal Aluminum Mounting Hub Pair

ITEM #1079

$5.95/pair

Other shaft sizes available

Mount custom wheels and mechanisms to 3 mm motor shafts.

3pi Robot

ITEM #975

$99.95

The 3pi is a great first robot for ambitious beginners and a perfect second robot for those looking to move up from non-programmable or slower beginner robots.

Simple Motor Controller 18v15

ITEM #1377

$39.95

Advanced motor control made simple

Highly configurable DC motor controller that supports four interface modes: USB, TTL serial, analog voltage, and hobby radio control (RC).

30-Tooth Silicone Track Set

ITEM #1416

$14.95

Available with shorter tracks

Track set includes a pair of 30-tooth silicone tracks, two drive sprockets measuring 1.38" (35 mm) in diameter, and two matching idler sprockets along with mounting hardware.

Mini Plastic Gearmotor

ITEM #1125

$5.49

Other shapes and gear ratios available

@4.5V:

- 120 RPM
- 20 oz·in

25D mm Metal Gearmotors with 48 CPR Encoders

$36.95

Other sizes available

Many motor winding and gear ratio combinations stocked.

August 2013

Columns

Departments

Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

Jaycar Electronics

Better, More Technical

www.jaycar.com
1800 784 0263
August 2013

DIY Kits for Electronics Enthusiasts

ARDUINO KITS

Arduino Experimenters Kit

Learn about the exciting new world of Arduino with these easy to build projects. From flashing an LED to moving things with a servo. Complete with instructions and a supporting web page and software examples.

- No soldering required

Cat. XC-4262

$65.00*

Thousands Sold

RGB LED Cube Arduino Kit

Watch video online

This stunning 3D-matrix of 64 RGB LEDs incorporates an onboard Arduino-compatible controller so you can produce mesmerizing light shows controlled by software. Use it as a mood light or create your own "ambient device" that gently notifies you of new email or instant messages. Some assembly required.

- 4 x 4 x 4 matrix of individually addressable 8mm RGB LEDs
- Size: 106(W) x 130(H) x 106(D)mm (assembled)

Cat. XC-4274

$65.00*

Automotive Kits

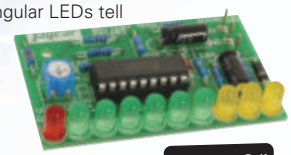
Car Battery Monitor Kit

Don't get caught with a flat battery! This simple electronic voltmeter lets you monitor the condition of your car's battery so you can act before getting stranded. 10 rectangular LEDs tell you your battery's condition.

- Kit includes PC board and all components
- PCB: 62 x 39mm

Cat. KA-1683

$13.00*



Smart Fuel Mixture Display Kit for Fuel Injected Cars

This improved model has an emergency lean-out alarm, better circuit protection and an auto dimming display. Another great feature is the 'dancing' display which operates when the ECU is operating in closed loop. Kit supplied with PCB and all electronic components.

- Car must be fitted with air flow and EGO sensors (standard on all EFI systems) for full functionality
- PCB: 121 x 59mm

Cat. KC-5374

$21.75*

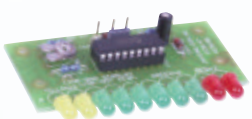
Mixture Display Kit For Fuel Injected Cars

This very simple kit will allow you to monitor the fuel mixtures being run by your car. This type of sensor is also known as an E.G.O. (exhaust, gas, oxygen) monitor. The circuit connects to the EGO sensor mounted in the exhaust manifold and the cars battery. PCB, LEDs and components supplied.

- PCB: 74 x 36mm

Cat. KC-5195

$12.25*



Digital Pulse Adjuster Kit

A huge step up in DIY automotive performance upgrades. It allows you to control and tune the operation of any solenoid that is run by the engine management system. This means that you could control turbo boost without an expensive boost controller, or alter automatic transmission line pressures for better shifts. Alternatively, it can be used to drive and control an extra fuel injector.

- Kit supplied with a quality solder masked PCB with overlay, machined case with processed panels, 2 programmed micro and all electronic components
- Kit requires the Hand-held Digital Controller **(KC-5386 $49 shown below)** and connecting cable **(WC-7502 $12)** 25 pin extension cable with all pins connected)

Cat. KC-5384

$57.75*

Hand Controller Kit

This hand controller is used for the mapping/ programming the Digital Adjuster kits above. It features a two line LCD, and easy to use pushbuttons. It can be used to program the adjuster then removed, or left permanently connected to display the adjuster's operation. It is designed as an interface and display, and is not required for general adjuster functions after they have been programmed.

- Kit supplied with silk screened and machined case, PCB, LCD, and all electronic components
- Must have all D25 pins connected

Cat. KC-5386

$49.00*

USB Port Voltage Checker Kit

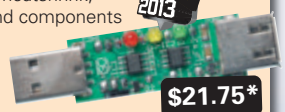
An easy way to test a USB port to see if it is dead, faulty or incorrectly wired to help prevent damaging a valuable USB device you plan to connect. Voltage is indicated using three LEDs. Kit supplied with double sided, soldermasked and screen-printed PCB with SMDs pre-soldered, clear heatshrink, USB connectors and components for USB 2.0 & 3.0.

NEW 2013

- PCB: 44 x 17mm

Cat. KC-5522

$21.75*



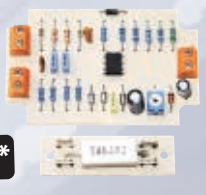
Peak Hold Injector Adaptor

Conventional 'saturated' fuel injectors are operated by applying a pulse waveform, however there is one different type known as peak hold injectors, which is found in some cars. This adaptor allows you to use the Digital Pulse Adjuster (shown on left), Independent Electronic Boost Controller and Digital Duty Cycle Meter circuits where you have these peak hold injectors.

- This kit is required to use the Digital Pulse Adjuster **(KC-5384 $57.75)**, Independent Electronic Boost Controller **(KC-5387 $57.75)**, and Digital Duty Cycle Meter **(KC-5375 $39.75)** projects with Peak Hold injectors
- Kit supplied with PCB and all electronic components

Cat. KC-5395

$14.50*



Garbage & Recycling Reminder Kit

Easy to build kit that reminds you when to put which bin out by flashing the corresponding brightly colored LED. Up to four bins can be individually set to weekly, fortnightly or alternate week or fortnight cycle. Kit supplied with silk-screened PCB, black enclosure (83 x 54 x 31mm), pre-programmed PIC, battery and PCB mount components.

- PCB: 75 x 47mm

Cat. KC-5518

$21.75*

Capacitor Discharge Ignition Kit for Motor Bikes

Many modern motor bikes use a Capacitor Discharge Ignition (CDI) to improve performance and enhance reliability. However, if the CDI ignition module fails, a replacement can be very expensive. This kit will replace many failed factory units and is suitable for engines that provide a positive capacitor voltage and have a separate trigger coil. Supplied with solder masked PCB and overlay, case and components. Some mounting hardware required.

- PCB: 45 x 64mm

Cat. KC-5466

$16.00*

For more details on each kit see our website

HOW TO ORDER

PHONE: 1800 784 0263*
FAX: +61 2 8832 3118*
EMAIL: techstore@jaycar.com
POST: P.O. Box 7172 Silverwater DC NSW 1811 Australia

*Australian Eastern Standard Time (Monday – Friday 6.30am – 5.30pm)
* US Eastern Standard Time (Monday – Friday 4.30pm – 3.30am)

Prices valid until 31/08/2013

*ALL PRICES IN USD & EXCLUDE POSTAGE & PACKING

NOW SHIPPING VIA DHL

5 - 10 day working delivery

• FAST DELIVERY • TRACK SHIPMENT

Note: Products are dispatched from Australia, so local customs duty & taxes may apply.

DHL

Don't Just Sit There...Build Something!

DEVELOPING PERSPECTIVES

Dedicated Chip vs. Microcontroller

Given the capabilities of the Raspberry PI, Parallax Propeller, and revved-up Arduino clones, why bother with dedicated chips? After all, when was the last time you worked with, say, a 7472 J-K flip-flop chip? I can recall a time when I had a plastic parts box filled with 7400-series chips, but I don't own a single 7400 device today. Have I replaced the dozen parts boxes full of NE-555s and 7400 chips with a single Arduino Uno? No way. Those boxes are still full of dedicated chips.

What's changed in my arsenal of devices is the nature of those dedicated chip assortments. I use the microcontroller for what it does best — for logic and control. The dedicated chips tend to be peripherals for the microcontroller — an assortment of sensors, actuators, and power-handling components.

I have a box of IR and ultrasonic range finders, digital compasses, accelerometers, servos, and electronic speed controllers for my quadcopter and ground robotics projects. Then, there are the high voltage optoisolators, diodes, and vacuum tubes for my guitar amp projects. There's a box dedicated to laser diodes and optical detectors for an ongoing robotics project, as well as microphone modules and vibration sensors for an energy-harvesting project. You get the idea — dedicated chips are still a necessity, it's just the nature of the chips has changed.

I mention my inventory of parts because there's a small but continuous stream of readers who write in asking for more component-level articles and that they don't want to make the leap to microcontrollers. I try to make the case that there can be just as much wiring and soldering and use of dedicated components with a microcontroller project as there is with a traditional circuit. It's up to you.

There are exceptions, of course, and some technologies have simply run their course. I get a certain sense of nostalgia when I wind my own toroidal transformers and use a capacitance meter to identify matching capacitors for an LC filter. I do miss the low level hands-on activity, and there's something to be said for simplicity.

Unfortunately today, time, money, and often space constraints dictate that I design the equivalent filters in software or consider pre-configured filter modules. Besides, now I can focus on the big picture — integrating a circuit with the Internet, for example.

So, if you're one of those readers more comfortable in the discrete component space, I suggest you step out of your comfort zone and at least try your hand at a hybrid microcontroller circuit. If you're into vacuum tube circuits, how about a microcontroller-based transconductance meter? If you're a ham radio enthusiast, what about a microcontroller-enhanced antenna tuner? Into lab equipment? Then, how about adding a digital readout to your Geiger counter? In short, there's no end to the ways you can break into the world of microcontrollers and still flex your soldering and discrete component muscles. **NV**

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER/ASSOCIATE PUBLISHER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Joe Pardue, Ron Hackett, Craig Lindley, Matthew Bates, Russ Kincaid, Fred Eady, Lou Frenzel, Thomas Henry, Ron Anderson

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@nutsvolts.com

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION
Sean Lemieux

READER FEEDBACK

Fluid Remarks

I loved Louis Dratwa's article, *The Lost Art of Strip Board Prototyping*, in the June 2013 issue. I also use strip board extensively for my prototyping — and even for multiple copies of a project — but I still learned a lot from his article. One thing I would like to add is to avoid using flux remover spray on the board.

I had a case of unexpected voltages in a circuit. I started removing parts one by one, and found that the supply voltage was leaking through the board material and appearing in the strangest places. I even had a resistor with parts no longer connected to it that measured three volts on one end and 0.2 on the other. Apparently, the board absorbed the fluid and became partly conductive.

Don Hicke
San Diego, CA

PIC Better Pick?

In the June 2013 article, *An Electronic Photocell for Lighting Control*, Gerry Shand uses a Maxim DS1302+ real time clock chip. Couldn't Gerry have used the Microchip PIC18F2520-I/SP's Timer 1 counter to implement a software real time clock and calendar (RTCC)? Microchip application note AN1303 shows how to implement an RTCC on a PIC16F1827 (see **ww1.microchip.com/downloads/en/AppNotes/01303A.pdf**) which also uses a 32.768 kHz crystal (the schematic in Figure 2 and Parts List description call out 32.767 kHz but the Parts List part number calls for 32.768 kHz).

Tim Brown PhD EE, PE
Honea Path, SC

Good catch on the crystal frequency.

For my part, I'd say yes, the PIC would have worked fine. Actually, just about every project in N&V can be implemented in some fashion on a microcontroller. The DS1302 is relatively inexpensive, assuming you have one on hand (shipping would wipe out any savings).

Bryan Bergeron, Editor

Thanks for your feedback, Don. Here is my answer on your two questions:

1. Yes the crystal frequency for the DS1302 on the schematic should read 32.768 kHz instead of 32.767 kHz.

2. We chose the DS-1302+ as the timekeeper chip because we were already familiar with the IC and we had already developed code from previous projects that we could transplant onto this project.

If you look at the circuit, you'll find that we have the math co-processor for doing the sunset and sunrise math calculations, the timekeeper chip for keeping time (and it also has a super capacitor for keeping the memory fresh), and the PIC as the "traffic cop" to marshall all the signals to and from everything else — plus, an independent LCD. This was the original design topology that was agreed upon when the design first began. This configuration uses a few more parts but it also made code testing, debugging, and troubleshooting much easier.

In my article, I also point out that the final PIC chosen was the third choice, so we minimized the change in the design cycle to keep the project on track. So, yes we could have taken advantage of more features in the PIC18F2520, but this would have led to more iterations during the design phase.

My experience with major changes like this during detailed design on any project I have undertaken only creates more design challenges and increased capital costs. It's better to stay the course and manage any changes efficiently in smaller chunks ... like the old Proverb goes: If the dog hadn't stopped to lift his leg, he'd have caught the rabbit.

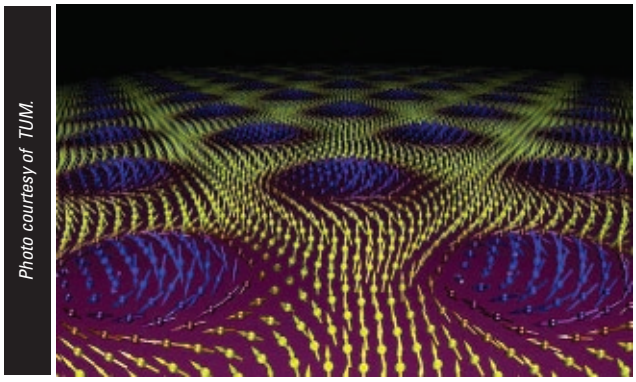
There are other design options including your suggestion, as well as creating a "one chip wonder." There is no one correct solution or answer to this design challenge.

Gerry

■ BY JEFF ECKERT

Discuss these topics at **http://forum.nutsvolts.com**.

ADVANCED TECHNOLOGY



■ Grid of magnetic vortex structures.

Yes, Virginia, There Is A Monopole

The magnets we deal with in real life are dipoles, meaning that they have both a north and south pole. A monopole, therefore, would take the form of a magnet that only has either a south or north pole but not both. The existence — or lack thereof — of magnetic monopoles has been an issue for a long time. Gauss's law for magnetism (which is really one of Maxwell's equations) indicates that they cannot exist. However, as early as 1894, Pierre Curie insisted that — like Santa and the Loch Ness monster — they could exist but perhaps just had not been observed yet. Apparently, Curie was right, because in 2009, physicists from the Technische Universität München (TUM, **www.tum.de**) accidentally found some of them swimming around in a sample of manganese silicon.

They also discovered that because the magnetic moments in that compound normally form a helix, if you arrange three of these helical structures on top of each other, the monopoles form magnetic vortices called skyrmions — but only at about -245°C. A skyrmion (in case you are wondering) is defined as "a topological soliton used in the mathematical modeling of baryons."

Got that? Well, never mind. The important thing is that the researchers later discovered that magnetic vortices exist in other unspecified materials as well, sometimes at room temperature. What we care about is that skyrmions can be used for long-lived data storage, and because they can use as few as 15 atoms to store a data bit (compared to about a million atoms on your hard drive), they could be used to create extremely compact storage media. Plus, shifting the state of a skyrmion takes 100,000 times less current than resetting a magnetic memory bit, so such storage media could operate on ultra low power.

Don't expect to order a Seagate skyrmion drive anytime soon. However, according to Prof. Christian Pfleiderer (one of the original team leaders), "... we finally have a method at hand that allows us for the first time to observe skyrmions in systems that are relevant for applications. This is a decisive step in the direction of a real technical use." ▲

A Bug In Your Ear?

With their compound eyes, flies react pretty well to perceived movements. That's why whacking them with a swatter is something of a challenge. On the other hand, most of them "hear" rather poorly because they do not have human-like ears and sense sound waves only through their antennae. An exception is *Ormia ochracea*, a housefly-sized insect that annoys people all over the southeast USA and Central America.

According to Prof. Ronald Miles of Binghamton University (**www.binghamton.edu**), Ormia have eardrums that operate like our own, and "can hear quite well." This is important to the flies, because the females use hearing to locate singing male crickets, upon which they deposit their larvae. Moreover, the flies have "remarkable" directional hearing, which is a tough thing to accomplish when your "ears" are that close together.

Now — based on how the fly does it — Miles and colleagues have designed a microphone that cleverly employs a tiny 1 x 3 mm diaphragm that rotates around a central pivot in response to sound pressure gradients. It also uses an electronic feedback system ("active Q control") to impose electronic damping on the diaphragm which achieves a noise floor that is 17 dB lower than the best hearing aid microphones currently on the market.

According to Miles, the device is promising for applications in areas including hearing aids, cell phones, surveillance, and noise control systems, and it "could easily be made as small as the fly's ear." ▲

■ An *Ormia ochracea* atop a human fingernail.

COMPUTERS and NETWORKING

Is It An AIO Or Tablet?

If you can't decide whether you need an all-in-one desktop or a tablet, maybe you don't have to. One of the latest from Dell (**www.dell.com**) is the XPS 18 which is pretty much both. When propped up on its (optional) powered stand, you can use it with a mouse and keyboard. You can also yank it off the stand and use it in tablet mode. At 4.85 lb (2.15 kg), it's about 3.5 times as heavy as an iPad, so it won't be as easy to carry around. However, it sports a full HD 1.84 inch capacitive touch display which nearly doubles the iPad's diagonal measurement.

The XPS can be had with a choice of Intel Core i3, i5, or i7 processors, up to 8 GB of SDRAM, and a 500 GB drive. A 32 GB Flash module is optional. You get up to five hours on a battery charge, and it comes with Windows 8 installed. The list price (depending on options) runs from $999.99 to $1,499.99. ▲

■ Dell's XPS 18 can be used in desktop or tablet modes.

Is It A Smartphone Or Tablet?

In another effort to straddle the space between two popular device configurations, some manufacturers are selling larger-screen smartphones inelegantly dubbed "phablets." The one that first comes to mind is probably Samsung's Galaxy Note, but a comparable — and cheaper — alternative has been introduced by LG in the form of its Android-based Optimus G Pro.

Both come with quad-core processors and 2 GB of RAM, but the Optimus specs are better in most categories. It's 6g lighter, has a better display (401 vs. 267 pixels per inch), and comes with higher-res cameras. The main drawbacks seem to be that it doesn't have a stylus and is available only through AT&T. It does have a built-in "infrared blaster," so you can use it as a remote control for your TV, if you care about things like that. Look for a price tag of $199.99 with a two year agreement. ▲

■ LG's Optimus G Pro, designed to give Galaxy Note a run for the (less) money.

Rugged SSD Offers Self-Encryption

If you need a solid-state drive system with maximized data security, reliability, ruggedization, and data transfer rates, a new self-encrypting SSD line from Microsemi (**www.microsemi.com**) may be of interest. The first offering in the company's TRRUST-Stor 200 series is a 256 GB, 2.5 inch drive that — according to company specs — provides 200 MB/s sustained data rates, making them "ideal for high bandwidth land-, air-, and ship-based video streaming and surveillance applications where recording every second of data is important."

The drive "also provides ample storage for imaging applications such as those used in UAVs and vetronics [vehicle electronics]." Data is protected via a hardware-based implementation of AES-256 encryption with XTS mode, flexible key management, and a purge function that renders data forensically unrecoverable in less than 30 ms.

The unit can withstand up to 3,000 G of shock and 30 Grms vibration. Company literature omits any mention of price, so price-conscious folks probably need not inquire. ▲

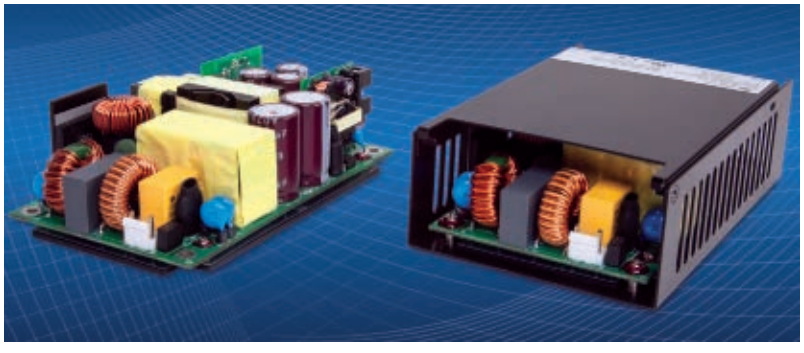
■ Microsemi's TRRUST-Stor 200 offers 256 GB of storage and 200 MB/s data rates.

CIRCUITS and DEVICES

93.5% Efficient AC-DC Power Supplies

If your project demands a beefy but quiet AC-DC power supply, the new VBM-360 series from CUI, Inc. (**www.cui.com**), may be a good option. Both open and enclosed versions are available in a standard 5 x 3 x 1.6 inch (12.7 x 7.6 x 4 mm) footprint, offering 14.8 W/in3 power density. Both offer baseplate cooling and feature typical efficiency of better than 93 percent. All models have a universal 90-264 VAC input for global operation, and are available with single outputs of 12, 24, and 48 VDC.

Open-frame models are rated for operation at 80 percent from -20°C to 40°C ambient, whereas enclosed versions are rated for operation at 100 percent load from -20°C to 40°C (both derating to 40 percent load at 70°C). Series features include active power factor correction and protections for overvoltage, overcurrent, short-circuit, and overtemperature. Connections for remote on/off control, remote sense, 5V standby, and a 12V fan output are also included. The price tag is $183 in quantities of 100. ▲



■ CUI's 355W power supply — open-frame and enclosed versions.

Catch Those Phantoms

For some years, there has been considerable grinding of teeth and yammering about phantom power which (in this context) refers to the power consumed by various appliances when they are switched off or put in standby mode. Also known as vampire power, phantom load, and other monikers, the phenomenon led to the 1998 One-Watt Initiative by the International Energy Agency, intended to reduce standby power consumption to 1W per appliance by 2010. Governments in most developed countries have followed the lead with regulations, and today the average LCD television falls within that limit (and half that in California). Power from my local plant costs roughly $0.09/kWh, so we're looking at a yawn-inducing penalty of about $0.79/year if I don't unplug the TV whenever I'm not using it.

Older appliances may drain up to 15W, though, so if you're miserly or just plain curious, you might consider the Save A Watt Phantom Power Indicator from P3 International (**www.p3international.com**). You just stick it in the wall, plug the appliance in question into it, and check the LED readout to see how much it's sucking out of you. The LEDs indicate power drain at 2W, 5W, 10W, and 15W levels. The units are available online for about $15. Don't forget to unplug yours, though, as they eat up 1W of power all by themselves. ▲

■ P3 International's Save A Watt phantom power measurement device.

Dippers For Tippers

So, you decided to indulge your lifelong fantasy of being a lounge musician, bought yourself a $79 Walmart guitar and a Mel Bay instruction book, and learned to play C, F, and G chords. You picked up a white sequined jumpsuit at Donny Osmond's last garage sale and changed your name to Rod Thruster, so it's show time. Soon you find yourself onstage at the St. Pia Zadora Golden Buckeye Supper Club,* bringing down the house with a soulful rendition of "Feelings." However, there's a problem: The club isn't paying you, so you're just working for tips. But inexplicably, your tip jar remains stone cold empty. It couldn't be because you suck as an entertainer, so what's going on?

Well, ding dong, it's very simple. Nobody carries money anymore. What you need is a digital tip jar from DipJar (**www.dipjar.com**). Just set it in front of you, and grateful audience members can swipe their credit or debit cards to show their appreciation. DipJar tips are preset to $1 per swipe, so really appreciative folks will have to do multiple swipes. That shouldn't be a big problem, though. All you have to do is pay a "small, one-time" charge to buy one, register it, and plug it in. Your tips will be automatically logged on one of the company's servers for later disbursement. Well, 92 percent of your tips, that is. DipJar keeps eight percent for its time and trouble — a rate that would even make American Express blush. Hey, that's show biz. ▲

** Apologies to Gary Burbank.*

■ DipJar, the tip jar for people who don't use cash anymore.

INDUSTRY and the PROFESSION

Bill Introduced To Protect Electronic Privacy

Earlier this year, a Freedom of Information Act inquiry turned up the Search Warrant Handbook, prepared by the Office of Chief Counsel for the Criminal Tax Division of the IRS in 2009, in which it is argued that "emails and other transmissions generally lose their reasonable expectation of privacy, and thus their Fourth Amendment protection once they have been sent from an individual's computer." Thus, the agency believes that Americans should expect "generally no privacy" in emails, Facebook entries, Tweets, and other digital data on the servers of third parties. Quite a few people and organizations have openly disagreed with that interpretation, including the ACLU, the Electronic Freedom Foundation, Amazon, Apple, AT&T, Intel, Microsoft, and many members of Congress. In response, Sen. Rand Paul recently introduced the Fourth Amendment Preservation and Protection Act of 2013, which would require specific warrants granted by judges before the IRS and others can obtain such information. California's Rep. Zoe Lofgren previously introduced similar legislation in the House of Representatives. If you feel one way or the other about the subject, you might drop your representatives a note expressing your opinion. The full text (three concise pages) can be found with a little searching at **www.eff.org**. **NV**

■ BY RON HACKETT
www.jrhackett.net

Have A Piece Of PICAXE Pi

Judging from the email response to my parting question last time, there's a fair amount of interest in exploring the possibilities of interfacing PICAXE processors with the Raspberry Pi (RPi), so this month that's exactly what we're going to do! It would probably take a couple dozen Primer articles to thoroughly discuss the various aspects of working with the RPi, but fortunately, there's a wealth of information available online, so there's no need for us to get into all those details here. Rather, I'll just suggest an online resource that I found to be especially helpful when I first started experimenting with the RPi, so you can begin your Pi explorations right away. (Of course, you may already have done that!)

Naturally, our first Pi-related project will be to construct a simple stripboard circuit that will allow us to easily access several of the Pi's 17 GPIO pins. If you have already begun investigating the Pi on your own, you know that the GPIO pins are all 3.3V level digital pins. You have probably also read something about "level-shifting" and why it's needed in order to safely interface the Pi with any 5V devices, including other microprocessors. However, we won't get involved with that particular complication.

In keeping with the Raspberry Pi (and PICAXE) philosophy that "simple" is preferable to "complicated," we're just going to run both systems (Pi and PICAXE) at 3.3V. As you may remember, all M2-class PICAXE processors can operate with a supply voltage as low as 1.8V, so 3.3V will work well for our purposes.

Of course, we're going to need a language to program the Pi side of things. When we get to that point (in the next installment of the Primer), I'll explain why we'll be using the Python programming language, and give some suggestions for getting started with Python, as well.

Before we begin this month, I want to thank the readers who took the time to respond to my question — I really do appreciate the feedback. In fact, I didn't get a single "no" vote. That doesn't mean, however, that there aren't some readers out there who have no interest in exploring the Pi. If you include yourself in that group, don't despair. Some of the material that we'll be covering will also be applicable to non-Pi related PICAXE projects — especially the discussion of Python programming.

Why PICAXE Pi?

Here's the very first question we need to address: "If the Raspberry Pi is such a powerful little computer, why do we need to interface it with a PICAXE processor in the first place?" The answer is simple: PICAXE processors have several hardware capabilities that are either missing entirely on the Pi, or are much more difficult to implement.

For example, there are no ADC and/or touch inputs on the Pi; its 17 GPIO pins are digital only. Also, the Pi has only one hardware PWM output and one serial I/O interface.

Pi users who need additional I/O capabilities have two basic options: Add additional hardware devices (e.g., ADC chips, touch sensors, PWM chips) which are sometimes expensive and frequently require specific software interfaces (e.g., serial, I^2C, or SPI); or implement the missing function(s) in software (if possible), which greatly increases the software complexity of a project.

Also, software solutions are frequently not as accurate as their hardware counterparts (software PWM is a good example.)

On the other hand, the Raspberry Pi includes many capabilities that are far beyond those of any PICAXE processor. A complete discussion of the Pi's capabilities would require far too much space. So, we'll get into the relevant details as we need them in future Primer installments. Right now, though, I just want to briefly mention four Pi capabilities that I think are the most relevant to PICAXE projects:

- **Graphical User Interface:** Create an onscreen interactive GUI for any PICAXE project, including buttons, dialog boxes, menus, scroll-bars, etc.
- **File Manipulation And Storage:** Store and retrieve huge amounts of data for a data-collection system.
- **Text To Speech:** Produce high quality speech from any text file; verbally report real time data values to the user.
- **Network Accessibility:** The interactive GUI can be accessible

from your LAN and/or the Web. Any modern computer, tablet, or smartphone can be used to interact with your project.

I hope the above list has whet your appetite, and got you started thinking about possible PICAXE-Pi projects of your own. However, before we get to that point, we need to implement a simple way of accessing the Pi's GPIO pins so that we can begin our experiments.

There are several commercial printed circuit boards (PCBs) on the market that we could use for this purpose, but it's not easy to know what features you need until you have had some actual experience. So (as usual), our first approach to interfacing with the Pi will involve a simple stripboard circuit.

Before we can actually design and build a basic interfacing circuit, however, we first need a good understanding of the Pi's power capabilities (and limitations) and its GPIO pin arrangement.

Setting Up A New Raspberry Pi

There are currently two available models of the Raspberry Pi. We will focus entirely on the newer version (model B) because it's much more powerful than the older model A, and only costs an extra $10.

When you purchase a new Pi, you will receive the assembled unit in a small cardboard box. Absolutely nothing else is included (not even the operating system), so let's begin by discussing the additional items you need to get your Pi up and running.

In order to install and configure the operating system, you will initially need to connect the Pi to a monitor, USB keyboard, and USB mouse. Once you have completed the initial setup, there are two other options for connecting to the Pi that don't require any of the above three peripherals.

Assuming you can connect your Pi to a local network by using the Pi's Ethernet connector or an inexpensive Wi-Fi adapter (more on that at some point in the future), you can use any Mac or PC on your network to remotely access the Pi for programming and/or interacting with any software you have installed on the Pi. Again, we will eventually discuss those two possibilities (SSH and VNC, if you want to do your own research).

I'm only mentioning these two possibilities so that you don't think you have to run out and purchase a $200 monitor for your little $35 computer. If you don't have a spare USB keyboard and USB mouse on hand, you can temporarily borrow them from your Mac or PC, then return them once you have installed and configured the Pi's operating system.

In addition, any TV or monitor that has an HDMI or DVI connector can be used to initially set up your Pi. In fact, the Pi has a built-in RCA composite video connector, so that ancient analog display that's been collecting dust in your closet can also get the job done.

In addition, you will need a 5V power adapter with a micro-USB connector to supply power to your Pi. Since the B model consumes about 700 mA, that's the minimum size you should use. I'll explain shortly why you also don't want to use an adapter that can supply much more than 1A at 5V.

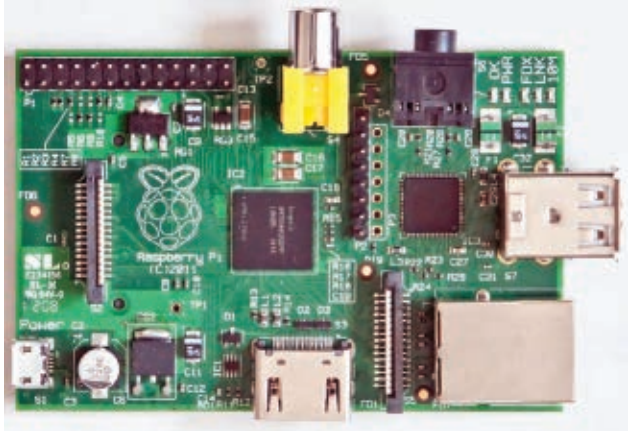
For now, just look for a good quality adapter that can supply 1A at 5V to your Pi. Fortunately, that size is readily available; you may already have a spare cell phone adapter that will do the job.

As I mentioned at the beginning of this article, I'm not going to provide a step-by-step guide to setting up your Pi because it would require much more space than we have available. Instead, as the need arises, I'll simply point you in the direction of an online resource that I think will provide the necessary information.

For the initial installation and configuration of the Pi's operating system, the best resources that I found are the tutorials available at **Adafruit.com http://learn.adafruit.com/category/learn-raspberry-pi**). Dr. Simon Monk and Limor Fried (Ladyada) provide a wealth of Pi-related information. In fact, the first seven lessons in the Adafruit series probably contain all the information you will need to understand just about everything we will be discussing about the Pi in this and future columns.

To get started, however, just scroll down to the bottom of the Adafruit link, and locate Lesson 1 (*Preparing an SD card for your Raspberry Pi*) which contains all the information you need to set up your Pi. I would recommend that you install the Raspian operating system; that's the version installed by the vast majority of Pi users.

Later, when you gain more experience, you may want to experiment with other Linux distributions for the Pi.



■ FIGURE 1. The Raspberry Pi, Model B.

RPi GPIO Pin Summary

Once you have

installed and configured the operating system, we're ready to discuss some of the details of the Pi's 26-pin I/O header. **Figure 1** is a photo of model B, in which you can see the I/O header in the upper right corner.

The pins are numbered sequentially from the top left to the bottom right, so the 13 pins on the left side of the connector have odd numbers and the 13 pins on the right side of the connector have even numbers.

Figure 2 presents a table of the pin functions for model B. (If you have the older model A, you already know that a couple of the pin functions differ, so you will need to make the appropriate adjustments to any software that we'll be discussing.)

The two innermost columns in **Figure 2** contain the physical pin numbers (in bold); the two GPIO columns show the corresponding GPIO number for each of the Pi's 17 GPIO pins; and the two Function columns indicate the optional "special function" of the pin where available. A dash in the Function column indicates that there is no special function attached to that particular pin.

Each DNC entry stands for "Do Not Connect," which requires a brief explanation. When you search for RPi information on the Web, you will see many pinout charts that indicate a function for each of the six DNC pins (either a 5V, 3V3, or Gnd). In fact, if you (carefully) use a multimeter to test the voltage at each of the six DNC pins, it will be the value that those charts indicate.

However, the Raspberry Pi Foundation reserves the right to implement other functions on those pins in future revisions of the RPi, and strongly recommends that they not be used at this point. We will be following that recommendation which is why the DNC label is included in **Figure 2**.

Function	GPIO	Pin #	Pin #	GPIO	Function
3V3	-	**01**	**02**	-	5V
I2C-SDA	2	**03**	**04**	-	*DNC
I2C-SCL	3	**05**	**06**	-	Gnd
-	4	**07**	**08**	14	TxD
*DNC	-	**09**	**10**	15	RxD
-	17	**11**	**12**	18	PWM
	27	**13**	**14**	-	DNC
-	22	**15**	**16**	23	-
*DNC	-	**17**	**18**	24	-
SPI-MOSI	10	**19**	**20**	-	*DNC
SPI-MISO	9	**21**	**22**	25	-
SPI-SCLK	11	**23**	**24**	8	SPI-CS0
*DNC	-	**25**	**26**	7	SPI-CS1
*See Text					

Figure 2. Pinout of the RPi GPIO connector.

RPi Power Considerations

Power is supplied to the RPi via its micro-USB connector. As I already mentioned, the model B requires about 700 mA at 5V to operate correctly, so a direct connection to a computer's USB port will not usually provide sufficient power for reliable operation.

This is why a separate USB power adapter is recommended. Also, as a safety feature, the internal circuitry in the Pi includes an automatically resetting fuse. When excessive current is being drawn, the resistance of the fuse automatically increases in order to reduce the current flow and protect the Pi from possible damage. If the current increases above 1.1A, the fuse "trips," which disconnects the power source and shuts down the Pi.

Once this happens, it takes some time (usually minutes, but possibly hours) for the fuse to automatically reset. The point is, if your Pi suddenly stops working, it may not be dead; just disconnect the power source and wait a while before attempting to re-boot it.

The internal fuse is also the reason why it's not advisable to use a USB power adapter that's capable of supplying more than 1A. If you use a 2A supply and accidentally create a short circuit, it's possible for a 2A current to damage the internal fuse. The moral of the story is, stick with a 1A power supply — you and your Pi will both be happier!

The Pi's internal power circuitry also includes an internal voltage regulator that reduces the 5V power supply down to 3.3V. That voltage is used to power the GPIO pins (in addition to other components), and 3.3V is also available on pin 1 of the GPIO connector.

It's important to be aware of the fact that the 3.3V line can only source a maximum of 50 mA. When the Pi is first powered up, the GPIO pins are each configured to source a maximum current of 8 mA. However, if you configured all 17 GPIO pins as outputs and tried to source 8 mA from each one of them, that would be a total current draw of 17 * 8 mA = 136 mA.

Since that amount of current far exceeds the capability of the 3.3V circuitry, either the regulator would shut down or internal damage could occur. Of course, these limitations are important to keep in mind when connecting any external circuitry to the Pi.

Designing A Stripboard Interface For The RPi

As you would expect, the purpose of our stripboard circuit is to enable us to easily connect the Pi's GPIO pins to a breadboard, so that we can experiment with interfacing the Pi with various breadboard circuits that include a PICAXE processor and other devices. As I was developing our stripboard circuit, I had the following two design objectives in mind:

- Keep it simple but still useful for a variety of projects.

• Don't exceed the Pi's current limits that we just discussed.

Let's briefly discuss these two objectives.

Keep it simple: The vast majority of breadboard-based interface circuits that I have seen can be divided into two general approaches: 1) Design a PCB that sits on top of the Pi and includes a downward-facing 2x13-pin female header that plugs directly into the Pi's 2x13-pin male I/O header, and one or more upward-facing female headers that allow the user to easily make connections to a breadboard circuit; or 2) Place the interface board next to the Pi, include an upward-facing 2x13-pin male header, and connect the Pi to the board with a short 26-wire ribbon cable that has a 2x13-pin female IDC connector on each end. Also, include one or more upward-facing female headers for connecting to the breadboard circuit.

I've already designed and tested a couple of PCBs that use the first approach, and they work well. However, in the interest of simplicity, I definitely wanted to use a stripboard for our first circuit. Unfortunately, a stripboard circuit is far from easy to construct using the first approach, so I settled on the "ribbon cable" approach for our first interface circuit.

Doing it that way, it's a relatively simple matter to interface all 17 of the Pi's GPIO pins to our breadboard circuits. However, for our purposes, being able to access all 17 I/O pins would be "overkill" — we can certainly accomplish a wide variety of interface projects with half as many pins. (As it turns out, I finally settled on a 12-pin I/O interface circuit for our breadboard circuits; we'll discuss that decision momentarily.)

Limiting the number of available I/O pins also helps us to accomplish our second objective, as we are about to see.

Don't exceed the RPi current limits: In order to conduct our experiments, there are three basic types of I/O connections we will need to make. The first is to connect one or more Pi output pins to one or more PICAXE input pins and/or one or more Pi input pins to one or more PICAXE output pins so that we can establish data communications between the two systems. The second is to connect some other device (e.g., a sensor) that can output data directly to one or more Pi input pins.

As long as we set up the directionality of the I/O pins correctly, both these situations require negligible amounts of current because inputs on both the Pi and the PICAXE are high impedance. However, we do need to make sure that excessive current isn't drawn if we accidentally connect a high level output on one device to a low level output on the other device.

We may also want to connect one or more Pi output pins to drive other devices — especially various LEDs that we may want to use as program status indicators. As you know, it can easily require 10 mA or more to drive a single LED, so we need to be especially careful in this situation. Fortunately, driving an LED in a 3.3V circuit requires less current than doing the same thing in a 5V circuit because the voltage drop across the current-limiting resistor is lower.

In order to accurately determine how much current is needed for this purpose, I tested five types of red LEDs that I had on hand by connecting each of them between the Pi's 3.3V and ground in series with a 470Ω current-limiting resistor (which actually measured at 465Ω), and measuring the voltage drop across the resistor in each case. The results are presented in **Figure 3**.

Using the nominal 470Ω current-limiting resistor, all five types of LEDs exhibited a reasonable level of brightness, and the most power-hungry among them (the 3 mm non-resistorized LED) only drew 3.4 mA. Even if we were to connect one of these resistors to each of the 12 I/O lines that we will be using and light all 12 of them at the same time (which is very unlikely), the total current draw would be 12 * 3.4 mA = 40.8 mA, which is within the safe operating limit of the Pi's GPIO pins.

Also, if we were to connect the same 465Ω resistor by itself between +3.3V and ground, the resulting current draw would be 3.3V/465Ω = 7.1 mA, which is again within the default maximum current draw of 8 mA for each Pi I/O pin.

Of course, this arrangement is equivalent to accidentally connecting a high level output from the Pi to a low level output from the PICAXE (or vice versa). Consequently, I decided to include a 470Ω resistor in series with each of the GPIO pins, in order to minimize the possibility of exceeding the Pi's current limitations.

There are two more current-related points that I want to mention before we move on. First, resistorized LEDs are so convenient that I frequently find myself inserting one into a circuit as a simple test device. For example, if I want to know if data is being received on the PICAXE *serin* line, I simply connect a resistorized LED between the *serin* line and ground; if the LED flickers, I know there's data coming in on the line.

I would like to be able to safely do the same thing with a Pi GPIO pin, so I also wanted to know the current draw of a resistorized LED

LED	Voltage Drop Across 465Ω	Current Draw
Non-Resistorized 3 mm	1.58V	1.58V/465 Ω = 3.40 mA
1 LED in 10-LED Bar Display	1.49V	1.49V/465 Ω = 3.20 mA
Non-Resistorized 5 mm	1.39V	1.39V/465 Ω = 2.99 mA
Resistorized 3 mm	1.01V	1.01V/465 Ω = 2.17 mA
Non-Resistorized 5 mm	.091V	0.91V/465 Ω = 1.96 mA

Figure 3. Current consumption of selected LEDs.

without the addition of the external 470Ω resistor. Since it's not possible to measure the voltage across the internal resistor on the LED, I set up my multimeter to directly measure the current flowing through a lit resistorized LED.

I found that the 5 mm resistorized LED draws about 5 mA, and the 3 mm draws about 6 mA — again within safe operational limits. Of course, it wouldn't be a good idea to connect 12 resistorized LEDs in this manner, but it does seem safe to use one of them as a simple testing device now and then.

Finally, it's worth mentioning one interesting fact that I discovered as I was investigating the current limitations of the Pi's GPIO pins: The source current limitations do not apply to sink currents. As we just discussed, individual outputs can only source 8 mA each, and the total source capability is limited to 50 mA. However, any number of individual outputs can sink 16 mA simultaneously.

In other words, the 12 GPIO pins that we will be using are capable of sinking a total of 12 * 16 mA = 192 mA. This is an important factor to keep in mind if we ever need to manage higher current loads.

Constructing The RPi Stripboard Interface Circuit

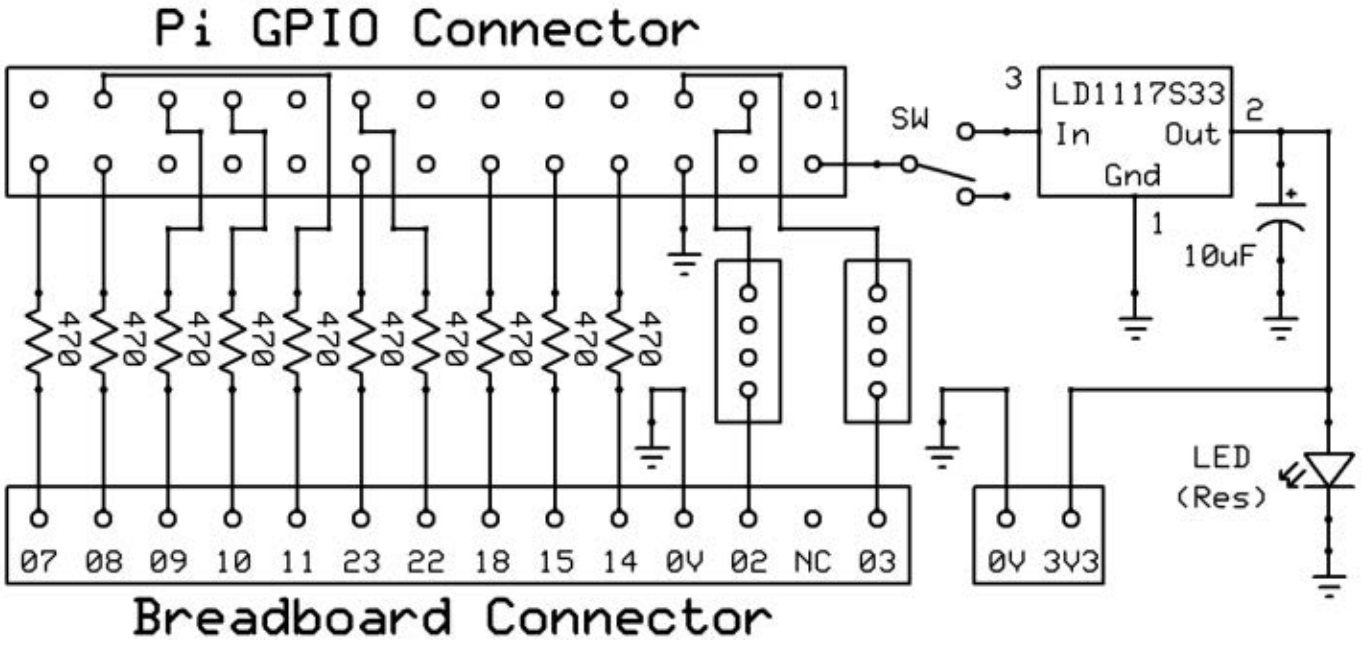
From the above discussion, it's very clear that the Pi's 3.3V current capabilities are rather limited. In comparison, the 5V supply line is much more robust. As we already know, model B requires about 700 mA to operate correctly. If we follow the recommendations and use a 1A USB power adapter, that leaves about 300 mA available on the 5V pin (#2) of the 26-pin GPIO connector which is more than enough to power our PICAXE processor, as well as any sensors or other devices that we may need in our various projects.

Of course, we want to power all our projects with a 3.3V supply to avoid the necessity of including level shifters in our circuitry. The solution is obvious — we'll just include a 3.3V regulator in our circuit to step down the voltage from the Pi's 5V power pin. An LD1117S33 regulator is well suited for our purposes because it has the following attributes:

- It can supply up to 800 mA which is more than enough for our purposes.
- The dropout voltage is about 1V which makes it suitable to drop 5V to 3.3V.

Capacitor, 10 µF, Electrolytic
Header, Female, two Pins
Header, Female, four Pins (two pieces)
Header, Female, 14 Pins
Header, Male, 26 Pins (2x13)
LED, Resistorized, 3 mm
Resistor, 470 Ω, 1/4 W (two pieces)
Power Switch, Small (300 mA)
Resistor Array, 16-Pin DIP (8 470Ω)
Voltage Regulator, LD1117S33, SOT-223

Figure 5. Parts List for interface board.



■ FIGURE 4. Schematic for interface board.

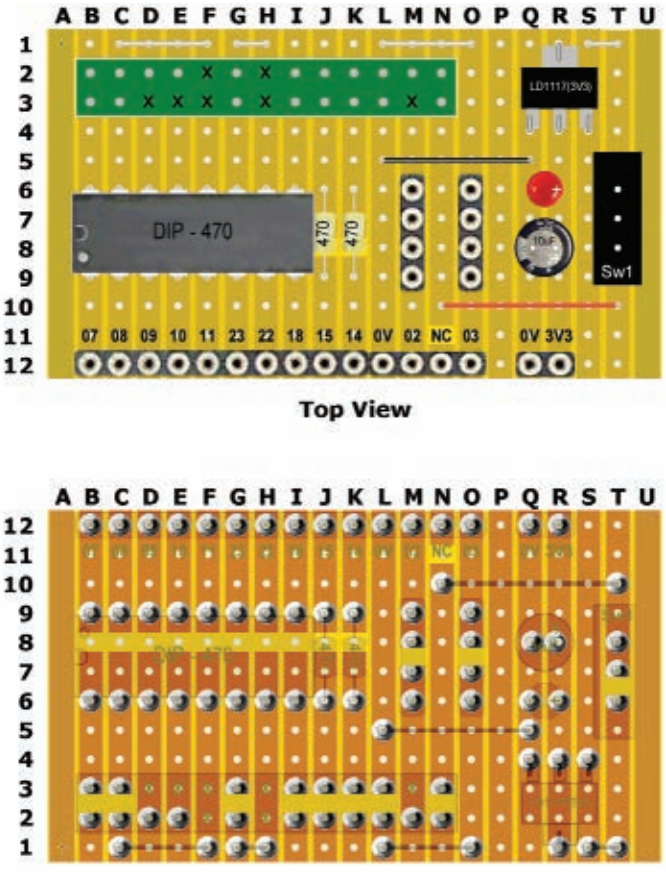
• It includes an internal thermal current-limit circuit.

• Only one additional component is required (a 10 µF capacitor).

The schematic for our stripboard interface circuit is presented in **Figure 4**; the complete parts list is shown in **Figure 5**. (All the parts for the board are available on my website.) I will also be carrying 26-pin IDC ribbon cable connectors which you can use to construct your own ribbon cable, assuming you have accumulated a few ribbon cables from old hard drives or other equipment. If not, a suitable cable (with connectors) is available at Adafruit (ID #862) and elsewhere.

As I already mentioned, we're interfacing 12 of the Pi's 17 GPIO pins. If you look back at the Pi's GPIO pinout in **Figure 2**, you will see that 10 of the GPIO pins can also implement special functions: I²C (2 and 3), SPI (7-11), serial I/O (14 and 15), and PWM (18). My original plan for the board only included those 10 pins because I wanted to be able to experiment with the Pi's special functions. However, when I designed the stripboard circuit, GPIO 22 and 23 just fell into place on the board, so I included them.

At this point, you may be wondering why the 470Ω current-limiting resistors are not included on the GPIO 2 and 3 lines. The reason is that the I²C interface will not function correctly if a current-limiting resistor is included. (We'll discuss that if and when we experiment with the I²C interface.) For now, just insert a 470Ω resistor into each of the four-pin female headers so we can use I/O pins 2 and 3 as regular GPIO pins.



■ FIGURE 6. Stripboard layout for interface board.

The stripboard layout for our interface board is presented in **Figure 6**; a full-size version is available on the *N&V* website. There are four points that I need to clarify:

1) An eight-resistor DIP IC is shown, but you certainly can replace it with eight discrete resistors if you prefer.

2) I like to keep a little distance between the power supply pins and the data pins, so I have separated the two supply pins from the other 14 pins along the lower edge of the board (where we will abut a breadboard so that we can use jumper wires to make the necessary connections).

As you can see, there's no circuitry in column P, so you could move all the power supply components to the left by one row and just use a single 16-pin female header if you would rather do it that way.

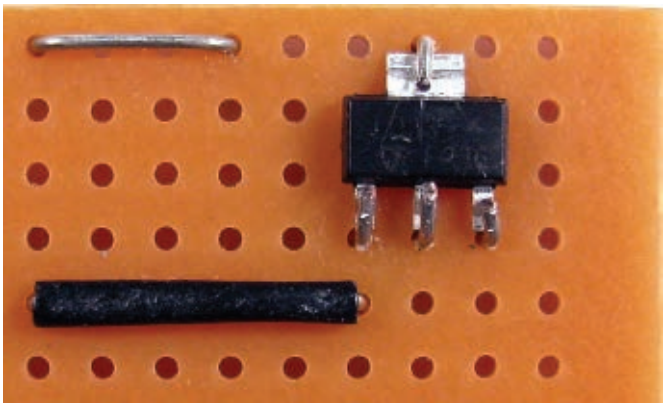
3) The seven "x" symbols on the 26-pin male connector (which mates with the ribbon cable) indicate that those pins should be **entirely** pulled out of the black plastic before soldering the connector in place. (They can easily be removed with a pair of needle-nose pliers.)

Also, double-check your work in this area — make sure you remove the correct pins; that you insert the connector in the correct orientation; and that you have cut the correct traces between the two rows of pins on the connector. **A mistake in this area can permanently damage your Pi!**

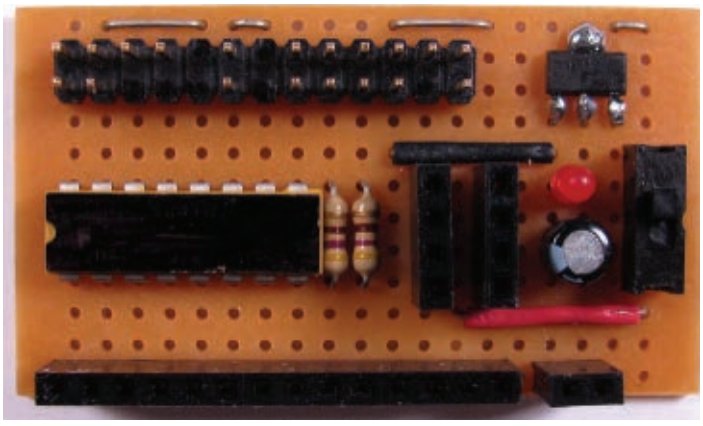
4) As you can see, the LD1117S33 looks a little weird! As I mentioned earlier, I had previously constructed a couple versions of this project that plugged directly into the Pi's GPIO connector, so that the stripboard sat directly on top of my Pi (which is in a plastic case). On those boards, it was a simple matter to surface-mount the regulator on the bottom of the stripboard.

However, in the present version of the project (which is much simpler to construct), the stripboard and the breadboard are both placed on the same flat surface. Soldering the regulator on the bottom of the stripboard would result in the board not sitting flat on the surface, which would make it slightly angled and somewhat awkward to use. As a result, I needed to find a way to solder the regulator on the top of the stripboard; hence, the weirdness!

I think it's easiest to solder the regulator in place first. When you



■ FIGURE 7. LD1117S33 mounted on top of stripboard.



■ FIGURE 8. Completed interface board.

have prepared the board and you're ready to start soldering, here's how I did it:

1) Cut four 1" pieces of bare jumper wire.

2) Place the stripboard (traces up) on top of a breadboard.

3) Insert one of the jumpers through hole R1, and adjust the stripboard so that the jumper can be fully inserted into one of the holes on the breadboard.

4) Similarly, insert the remaining three jumpers through holes Q1, R1, and S1, and fully into the breadboard hole that's underneath.

5) Solder and snip the four jumpers.

6) Turn the stripboard over and place the regulator between the four wires as shown in the layout; secure the regulator in place with a small spring clamp.

7) Snip the four wires to length, so that they can be bent over the regulator pins as shown in **Figure 7**.

8) Use a small screwdriver to bend each wire down on top of the pin.

9) Solder the regulator in place.

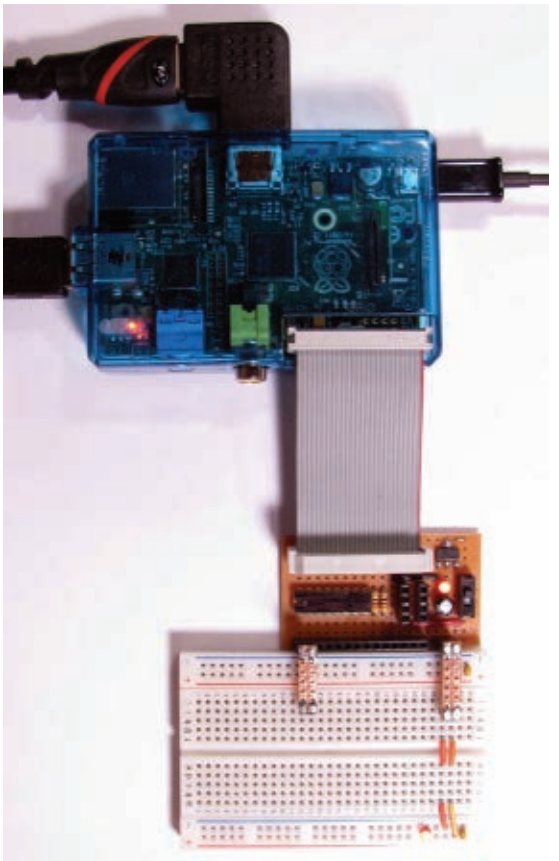
The remainder of the board is easy; as usual, just work from the smallest to the largest component. **Figure 8** is a photo of my completed stripboard, and **Figure 9** shows the interface board attached to my Pi. (If you're interested in the blue plastic case, it's ID #1144 from Adafruit.)

In **Figure 9**, you can see that I have connected the interface board to a breadboard with two small pieces of stripboard — each with a two-pin header soldered on each end.

I did that to make the connection between the stripboard and the breadboard more rigid, but it's not really necessary. If you prefer, you can simply use jumper wires for the connections.

Before you connect the stripboard to your Pi, it would be a good idea to use a continuity checker to thoroughly test the board for accidental shorts or unwanted connections. Most importantly, on the 26-pin male connector, make sure that no pin is shorted to any other pin. (I realize that's a lot of testing, but your Pi will thank you!)

Once again, we're completely out of space this month. However, I do need to include one final cautionary note. The switch on our interface board only turns off the power to the breadboard. As long as your Pi is up and running, all the GPIO pins are still powered. In fact, even if you shut down the Pi, it still remains powered.



■ FIGURE 9. Interface board connected to Pi and breadboard.

The only way to turn it off is to unplug the USB power cable. If you are going to be doing any hardware experiments, be sure to keep this caution in mind.

I had hoped to conclude this installment of the Primer with a simple "Hello World!" program, but that will have to wait until next time. In the interim, you may want to work your way through the first three or four Adafruit tutorials that I mentioned near the beginning of this article.

Also, if you do a web search for *raspberry pi python for beginners* (without quotes), you can get a jump-start on some of the material we will be covering next time.

As usual, have fun! **NV**

In this column, Russ answers questions about all aspects of electronics, including computer hardware, software, circuits, electronic theory, troubleshooting, and anything else of interest to the hobbyist. Feel free to participate with your questions, comments, or suggestions. **Send all questions and comments to: Q&A@nutsvolts.com**

- **A Q For Q&A**
- **Cat Latch Fever**
- **Walkie-Talkie This Way**

You can discuss these topics at **http://forum.nutsvolts.com**.

Ampere-Hour Meter

Q I (like many others probably) am interested in using rechargeable batteries to power many power-hungry projects. A useful device to have would be a simple and cheap ampere-hour meter that could be put between a battery and its charger or load to record how many ampere-hours had been input or output from a typical battery.

The device could perhaps also display instantaneous current and/or voltage at any desired time by pressing a button. Perhaps a simple three-character LED type display could be used instead of an alphanumeric one due to the simple nature of its use, plus the device could run off of the actual battery power being measured if its own power requirements were taken into account. If this complicates the project too much, then ignore it.

I'd like a device without a lot of bells and whistles, and that is cheap enough so that many could be used simultaneously on different battery projects. A device that dropped only 10 to 20 millivolts in its current measuring capability and could work with batteries of 3.6 through 14.8 or 24 volts at up to five amperes would cover many practical requirements.

External shunts and voltage dividers could be added to work with really powerful demands (like perhaps an electric bicycle), but keep the basic design simple and easy to use.

I am not yet versed in microcontroller use or I would do this myself.

— Ken Knaell

A *Ken and I collaborated on the analog design and I am quite sure it works. Unfortunately, neither of us is capable of programming the micro to get the amp-hours displayed. One purpose of presenting this incomplete design is to get some feedback so I can make it work.*

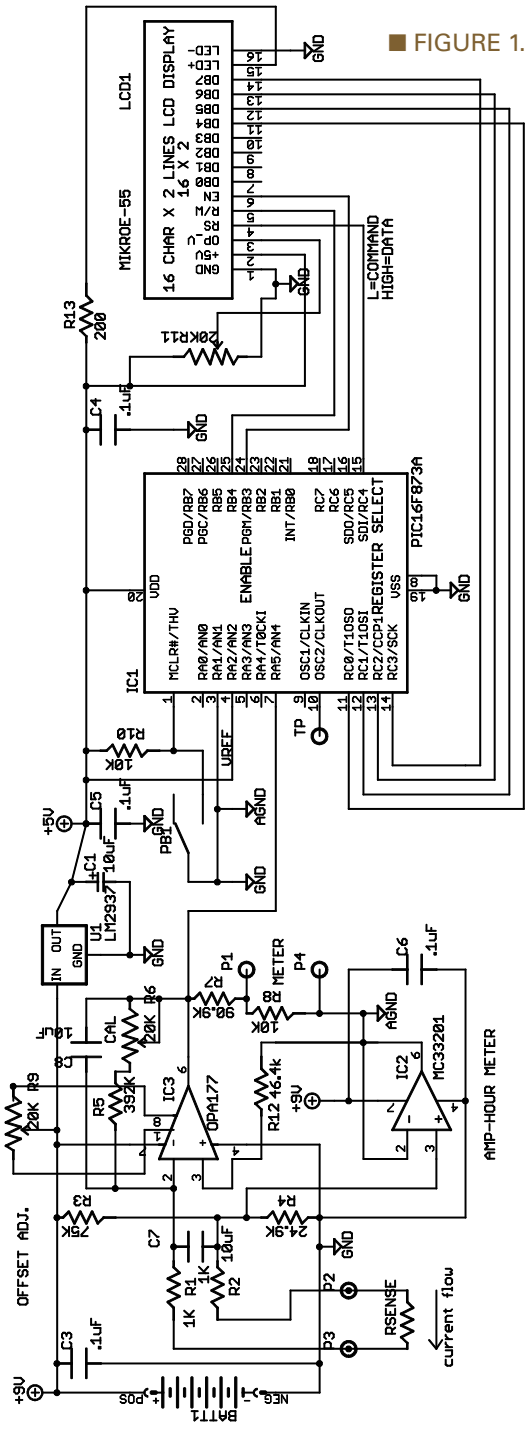
The schematic is **Figure 1**. The analog design requires battery operation so it can float at the sense resistor potential — whatever that may be. GND, in this case, is just the negative end of the battery and is unrelated to actual earth ground. AGND is another artifice such that zero volts difference between the op-amp output and AGND will indicate zero current. The offset adjustment, R9, is able to make that happen.

The sense resistor will drop .01 volts at the maximum current, so the sense resistor value is:

Rsense = .01/Imax

The OPA177 op-amp is low offset and low noise, and the noise is further reduced by C7 and C8

■ FIGURE 1.



which reduce the bandwidth to 0.1 Hz. The voltage divider, R7 and R8, provides an output for a multimeter that can display the current in millivolts (200 mV full scale).

Now the non-functional display: Initially, I wired it up as shown in the PICBASIC PRO Compiler manual, in the LCDOUT command description. It didn't work. I was not sure that the LCD display that I had chosen was Hitachi 44780 compatible, so I found another which claimed to be.

This one did not work either, but in re-reading the manual and the display datasheet I noticed the manual says that if LCDOUT is not used, the R/W pin can be grounded (and the schematic shows it grounded).

Since I plan to use LCDOUT, then R/W cannot be grounded, but where do I connect it? Evidently, R/W is a busy signal from the display, so I connected it to an input pin on the micro (RB4) and made a subroutine (see **Figure 2**) to wait until the display was not busy. Still nothing.

The example LCDOUT command sends two commands at once; perhaps the display can't handle that, so I separated them. Still nothing. I wish the system could give error messages. Evidently, there is some condition that is not satisfied but I have not figured it out.

Figure 2 is my test program (not intended to actually display amp-hours; that is another problem). It seems to me that my program agrees with the PICBASIC PRO manual and Chuck Hellebuyck's book *Programming PIC Microcontrollers With PICBASIC*, so I don't know why it doesn't work. Do you?

```
'*****************************************************************
'*  Name    : AMPERE-HOUR METER.BAS                              *
'*  Author  : RUSSELL KINCAID                                    *
'*  Notice  :                                                    *
'*          :                                                    *
'*  Date    : 10/20/2012                                         *
'*  Version : 1.1                                                *
'*  Notes   : RC0 THRU RC3 ARE DATA (PINS 11-14); RC4 (PIN 15) IS *
'*          : REGISTER SELECT TO DISPLAY; AN4 (PIN7) IS ANALOG   *
'*          : INPUT; RB3 IS ENABLE TO DISPLAY (PIN 24); RB2 IS   *
'*          : RESET TO DISPLAY (PIN 23); RB1 IS SERIAL/PARALLEL  *
'*          : CONTROL, PERMANENTLY HIGH (PIN 22).                *
'*****************************************************************
        REM DEVICE = 16F883      'LINE 14

        HIGH PORTB.1 : HIGH PORTB.2 : HIGH PORTB.3   'INITIAL CONDITIONS
        REM CONFIG1 = %1110000011100101 'DEBUG/EVP/FCMEN/IESO/BOREN DISABLED
                                        'PIN 1 IS MCLR, INT OSC CLKOUT
        REM CONFIG2 = %111111111111111 'PROT OFF, BROWNOUT OFF
        TRISA = %100110      ''AN2 IS Vref+, AN1 IS Vref- ,AN4 IS INPUT
        ADCON1 = %10110000   'ADC RIGHT JUSTIFIED,EXT VREF

        ADCON0 = %10010001   'AN4 INPUT, A/D IS ON  (LINE19)
        OSCCON = %01110111   '8MHZ INTERNAL OSC, CLKOUT
        PORTB = 0
        PORTC = 0

         DEFINE LCD_DREG PORTC
         DEFINE LCD_RSREG PORTC
         DEFINE LCD_DBIT 0
         DEFINE LCD_RSBIT 4
         DEFINE LCD_EREG PORTB
         DEFINE LCD_EBIT 3
         DEFINE LCD_BITS 4
         DEFINE LCD_LINES 2
         DEFINE LCD_COMMANDUS 2000     '20uS MAYBE?
         DEFINE LCD_DATAUS 50
         AMPS VAR WORD
         HOUR VAR WORD
         AMPHR VAR WORD
         HOUR = 0         'INITIAL CONDITION

         PAUSE 500       '1/2 SECOND WARMUP TIME
            TEST:
                 AMPS = 1
                 AMPHR = AMPS*HOUR + AMPHR    'ACCUMULATED AMP-HOURS
                 LCDOUT $FE,1
                 GOSUB WATE
                 LCDOUT $FE,"AMPERES = ",#AMPS
                 PAUSE 2500        '2.5 SECONDS TO READ THE DISPLAY
                 LCDOUT $FE,$C0
                 GOSUB WATE
                 LCDOUT $FE,"AMP-HOURS = ",#(AMPHR)
                 PAUSE 2500
                 HOUR = (1/3600)*5       '5 SECONDS TIME  (LINE 40)
                 GOTO TEST
        WATE:
                 IF PORTB.4=0 THEN RETURN
                 GOTO WATE

                 END
```

■ FIGURE 2.

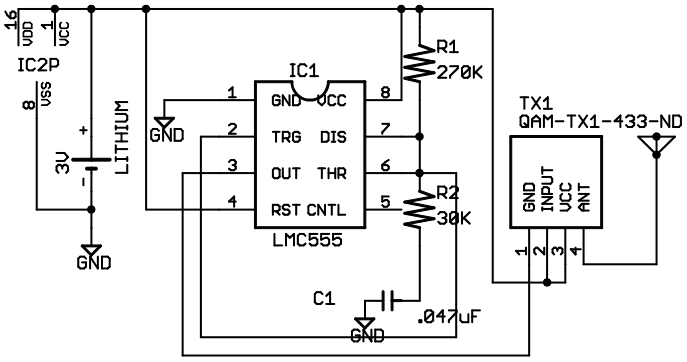
Cat Alarm

Q I have a cat that we keep inside all the time and I am having problems with the little guy getting out. Someone had him de-clawed and we

worry that he will not be able to defend himself. I would like to build a cat alarm that would sound a buzzer if he passes through one of two doors. I was thinking of something similar to the "theft protect" used in stores, but do not know enough about these systems and they are likely too expensive.

Another thought was to use a low power transmitter on the cat's collar, then each door would have a receiver and alarm buzzer. If the cat goes out the door, the alarm would sound, warning us he has escaped so we would have time to grab him before he got too far. Any ideas on a design that would work for this application? Thanks for your help.

— Bill Blackburn



■ FIGURE 3.

An RFID system will require research and experimentation, and unfortunately, I don't have time for that. A transmitter on the cat will require a battery that will last at least 24 hours.

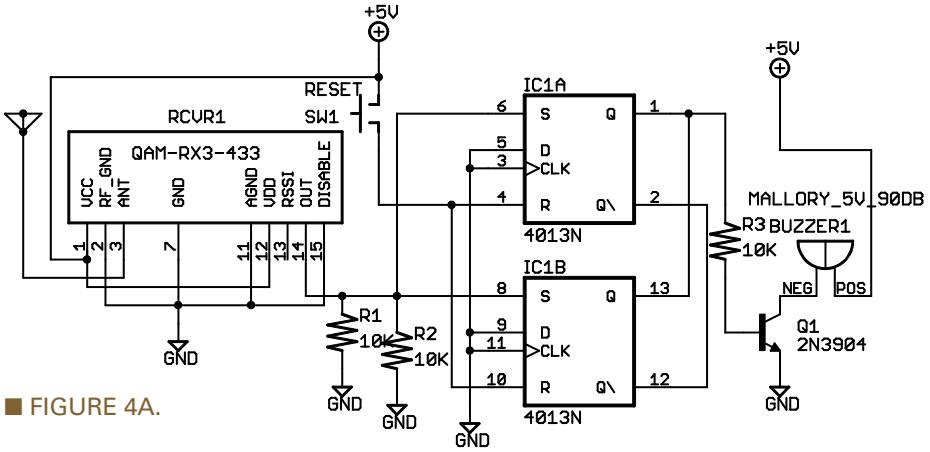
A three volt lithium coin cell has a rating of 350 mAh at 25 deg C and 1 mA load. The battery will last two weeks and costs $6 — that's $72 per year; do you feel the cat is worth it?

Now to find a transmitter that will run on 1 mA. Unfortunately, the low cost transmitter vendors do not supply a datasheet, but even if the transmitter draws 10 mA it could be turned on and off with a 10% duty ratio for a 1 mA average.

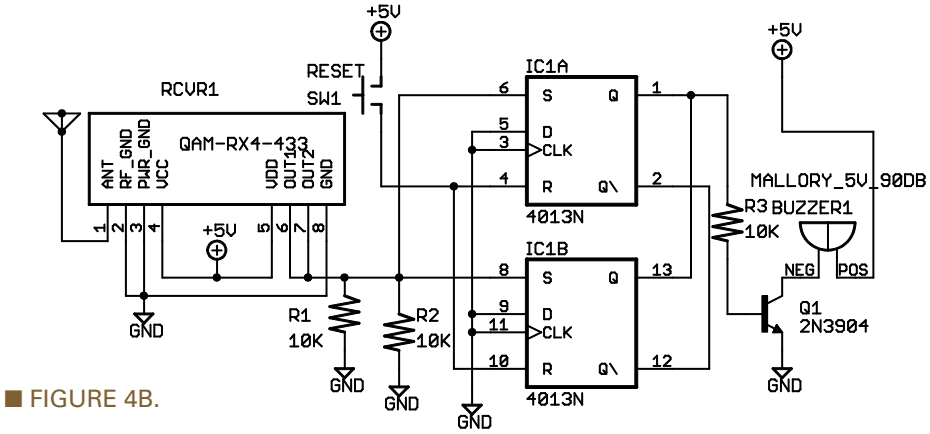
A very efficient antenna can be had that is inexpensive and about one inch by 1/2 inch. However, since the cat will be within three feet of the receiver at the door, an efficient antenna will not be required. A short piece of wire should work. My idea for the transmitter is shown in **Figure 3**. The 433 MHz transmitter module is available from several sources.

A Google search of the part number will find it for you.

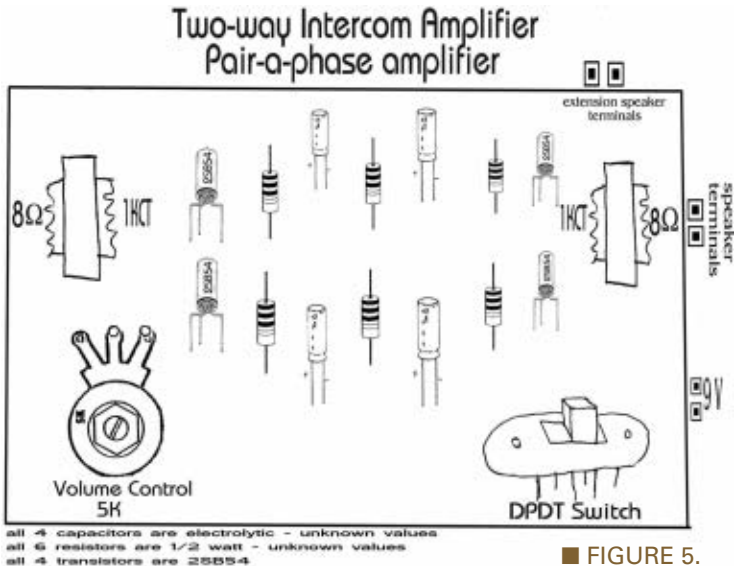
The QAM-TX1-433 and TLP434A are thru-hole, but I chose the QAM-TX1 because it is available from Digi-Key. The QAM-TX3-433-S and –ND are surface-mount for reflow soldering, but you can hand-solder it if the pads are made large enough to extend beyond the substrate so that a soldering iron can reach them. I don't know how the ASK is implemented or whether it is linear or not, so I tied the input high and turned the power on and off to implement two level ASK. It makes no difference to the battery. The 555 output is high for 90 mS, turning the transmitter off; and low for 10 mA, turning the transmitter on (100 Hz frequency). The receiver is an inexpensive AM type available from Quasar UK (part number QAM-RX4-



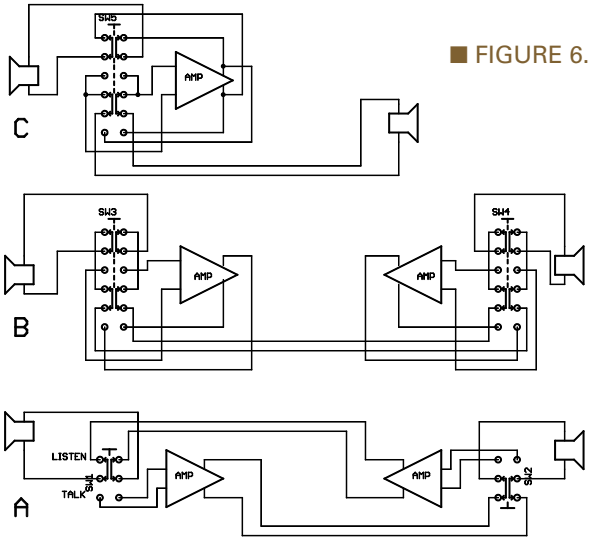
■ FIGURE 4A.



■ FIGURE 4B.



■ FIGURE 5.



■ FIGURE 6.

433) and Digi-Key, just add –ND to the part number (see **Figure 4A**).

ALTelectronics also has a receiver (part number RLP434A), but it has a different pinout; refer to **Figure 4B**.

It operates at the license free 433.92 MHz frequency — same as the transmitter. The receiver will operate on five volts; the transmitter will operate on a three volt coin cell (Mouser part number 658-CR2032; $0.45). The output sets the flip-flop which stays set until you reset it. The flip-flop energizes Q1, which turns on a 90 dB pulsing alarm. This will no doubt annoy the cat enough that it will stay away from the door.

You will want to trim the antenna such that it only alarms when the cat is within four feet of the door.

'**parts-placement diagram**' shown in **Figure 5**? I remember that it was powered from a nine volt battery, but don't remember the electrolytic capacitor's values or how the rest of the parts were connected.

I don't remember what I did with the finished kit, but I'd like to rebuild it — especially with the mini-audio transformers and multitude of germanium transistors I found in a pair of GE walkie-talkies bought very cheaply at a Goodwill store.

— Don Franklin

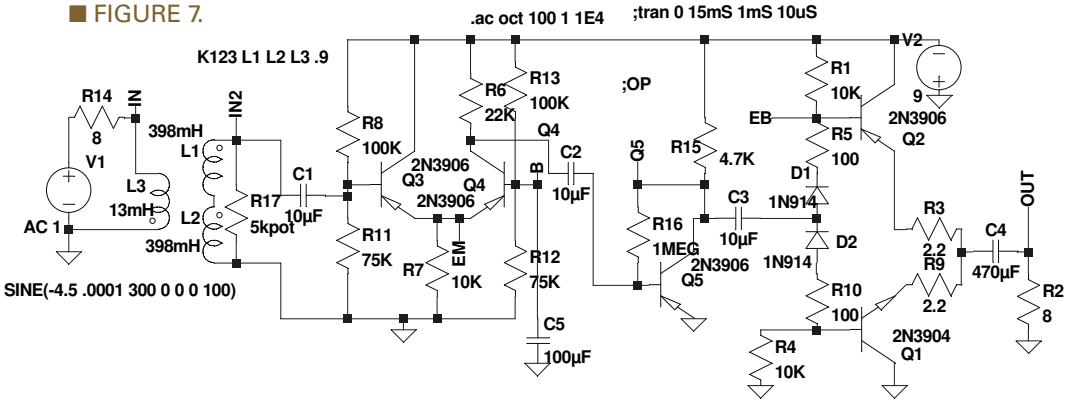
Two-Way Intercom

Q I read the request from a reader for an intercom diagram (June 2013; page 26) and I basically have the same request. Can you please re-create a schematic from the

A In **Figure 6**, I show several ways to hook up an intercom. **Figure 6A** uses the DPDT switch and requires four wires between units. **Figure 6B** uses a 4PDT switch and requires only two wires between units. **Figure 6C** is an option that only needs one amplifier, although the main unit is always in receive mode so the remote can initiate a conversation by speaking into the mic/speaker. In **Figures 6A** and **6B**, both units are in receive mode, so the system is quiet until one pushes the talk button. In **Figure 6C**, however, the main unit is always listening to the remote — however irritating that might be. In any case, the amplifier is shown in **Figure 7**.

A transformer at the input is an expensive gain stage compared to a transistor, but since it is free (part of

■ FIGURE 7.

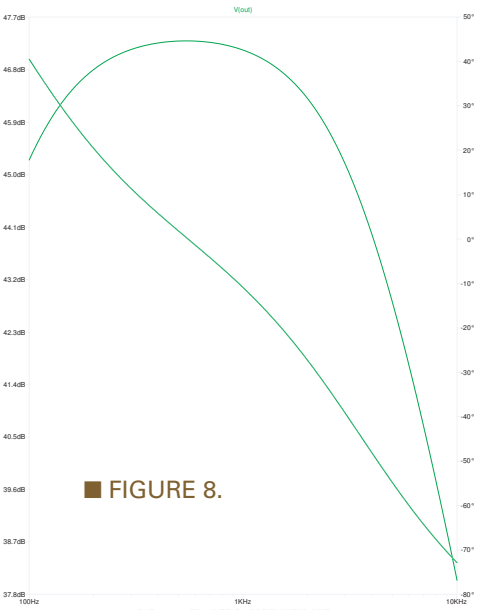


the walkie-talkies), I included it in the **schematic**. However, I think there is enough gain for the system to work without the transformer.

The first stage is a "long tail pair" which has wide bandwidth and low distortion, but the voltage swing is limited due to the necessity to bias the bases. Q3 is an emitter-follower driving the Q4 grounded base. The signal at Q5 can swing nearly rail to rail, but with some distortion.

The bias is highly dependent on the transistor beta (hfe), so you will need to tweak R16 to get between 4 and 4.5 volts at the collector. Q2 is an NPN which is not as common as PNP in germanium; Q1 and Q2 should be complementary pairs. AC128 (PNP and AC127 (NPN) have been suggested for the output transistors; you can find them on eBay. In **Figure 7**, I wanted to simulate the audio transformer, which required some guesstimates. If the high impedance side is rated 1K, the inductive reactance should be higher, say 10K.

The inductance — measured at 1 kHz — will be: L = 10K/(2*PI*1 kHz) = 1.59 Henries. The transformer is center-tapped with two equal windings. Since the inductance varies as the square of the turns, each winding will be 1.59/4 = 398 mH. Using the same logic on the low impedance side, if the rating is eight ohms, the inductive reactance should be 80 ohms; the inductance is: Ls = 80 Ω/(6.28*1K) = 13 mH. The simulated gain is **Figure 8**.



■ FIGURE 8.

My simulation used silicon transistors and diodes because there are no models for obsolete germanium transistors. I don't anticipate any problem in using either type in the circuit. If you use germanium transistors, be sure to use germanium diodes. **NV**

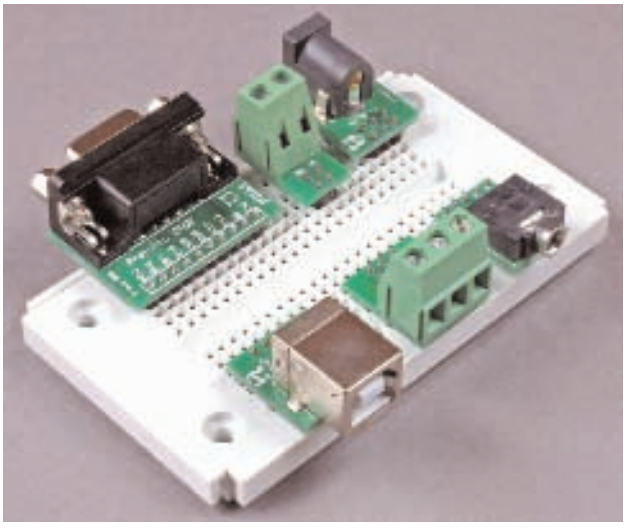
NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

CONNECTOR BREAKOUT MODULES FOR SOLDERLESS BREADBOARDS

Aztec MCU Prototyping now offers a growing line of connector breakout modules for use with solderless breadboards. Engineered in Canada and carrying their house brand — Omega MCU Systems (OMS) — these breakout modules bring the user a host of tangible benefits that help speed up the prototyping process, and make the use of solderless breadboards more reliable and the process more repeatable.

All modules are manufactured on high quality 1.6 mm thick double-sided FR4 fiberglass printed circuit boards with 1 oz copper traces, and use superior quality connectors. They plug directly into solderless breadboards to provide reliable connections to external devices and circuits instantly. There is no learning curve to deal with the benefits provided by these modules.



Each module is based on industry standard connectors, so they allow for the use of commonly available standard configuration cabling to connect user's prototypes. This means that interaction points such as computers, power supplies, USB devices, modems, amplifiers, or anything else a prototype needs to be able to work with can be hooked up quickly and reliably without the use of jury-rigged cables or the undependable rat's nest that inevitably forms with pigtailed connections.

The compact design of each module ensures the most efficient use of space, and uses flat-faced headers for insertion into the solderless breadboard which ensures reliable and low resistance connections.

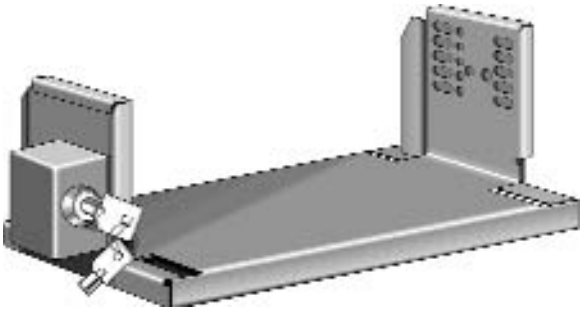
The current line-up consists of six modules, including a DB9F, DC power jack (5.5 mm x 2.1 mm), 3.5 mm stereo jack, USB "B", a three-position screw terminal, and a two-position screw terminal. A datasheet showing each module and connection diagrams is available on Aztec's website. Prices range from $2.19 to $3.49.

For more information, contact:
Aztec MCU Prototyping
Web: **www.aztecmcu.com**

SLIDE MOUNT FOR MOBILE RADIOS

The ET-850 universal slide mount for two-way radios from E Tip, Inc., provides a standardized mobile radio mounting platform for vehicles and other equipment applications, while allowing the operator with the key to remove the radio easily and quickly. Equipped with a tubular lock cylinder, the ET-850 helps to prevent unauthorized removal and transfers, tampering, and thefts of opportunity.

Different makes and models of radio can be fitted to the slide tray, allowing a variety of radios to be installed in any location with a locking base. Adapters are supplied to fit each radio to the slide tray.



Technicians for fleet operations save labor because the radio can be mounted onto the slide tray at the work bench. The locking base is securely mounted in the vehicle. The radio on the slide tray is then easily inserted into the base, eliminating blind fumbling with tools to secure hidden fasteners. Technician labor can be maximized with the ET-85.

The black powder coat finish is durable and attractive for use in a luxury automobile or a work vehicle.

Many fleet organizations such as

fire, law enforcement, utilities, schools, marine operations, trucking, contractors, service, and commercial organizations, etc., encounter expensive and disruptive loss of communications because of crimes aimed at two-way radios. Fleet communications are an important infrastructure component that is vital to providing superior service.

Using the ET-850 helps lessen the potential for disruption of a fleet communications network. The ET-850 fits most of the mobile two-way radios on the market today. For pricing. call (630) 966-8992.

For more information, contact:
E Tip, Inc.
Web: **www.etipinc.com**

WATERPROOF RED/BLACK PAIR CABLE

The J2 LED Lighting, LLC red/black pair cable is a bulk electrical/electronic grade black jacketed flexible interconnect cable with 18 AWG red/black conductors for DC power applications. It is a SJTW rated indoor/outdoor use cable that is waterproof and sunlight resistant. It has filler/separator material of cotton to maintain a round shape with flexibility and to aid in stripping.

The SJTW 18 AWG two conductor red/black pair cable type is found as the low voltage output cable on many UL8750/IP67 rated LED driver power supplies. These drivers are for both constant voltage (CV) and constant current (CC) LED circuit types.

The J2 RBC-18AWG-SJTW bulk cable provides a means for manufacturers and installers to extend the cables of LED drivers out to the luminaire head.

Specifications include:

- AWG: 18 (two PVC insulated conductors red and black) bare copper flexible 41 strand
- Amp: 10 max
- Volts: 300 max AC
- System volts: 12-24 typical
- Flammability: VW-1 UL and CSA/FT2 CSA
- Certifications: UL component (cable is marked UL)
- Maximum operating temperature: 105°C dry/60°C wet
- Composition: RoHS/Pb-lead free
- Filler/separator: Cotton based thread and paper
- SJ = Hard usage flexible cord (Trade Name = junior hard service cord)
- T = Thermoplastic insulation and jacket (PVC)
- W = Suitable for use in wet locations, and sunlight resistant

The RBC-18AWG-SJTW is designed and manufactured to the specifications for NEC (National Electrical Code) SJTW type cable. With this design, the scope of intended applications extends to many low voltage applications of the non-fixed variety including: portable lighting, power distribution, solar panels, and small wind generator uses. Applications include:

- Outdoor/indoor LED lighting
- Commercial lighting
- LED stage, studio, and theater lighting
- LED decorative lighting
- Water feature low voltage LED lights
- Stand-alone portable 12 volt solar panel systems and small wind turbine generators
- Portable 12 volt battery packs
- Low voltage DC extension power cables
- Connecting low voltage luminaries
- Power supply output cable assemblies
- Equipment interconnect
- Dock and pier LED lighting

Prices are:
1) 20 feet = $16.99
2) 100 feet = $67.99
3) 500 feet = $299.00

For more information, contact:
J2 LED Lighting, LLC
Web: **www.j2ledlighting.com**

DIGITAL AC/DC POWER SUPPLY

Global Specialties has introduced a new rugged general-purpose AC/DC power supply with variable outputs of 0-25 volts at five amps with digital LED (green) displays for motoring both voltage and current outputs. The power supply offers a

low cost space saving solution for applications requiring both AC/DC voltages. The 1315D is specifically designed for educational usage and prototyping, covering a wide range of circuit testing applications. This simple AC/DC power source can be used to demonstrate the concept of alternating and non-alternating voltages. It can also be used for simple experiments using lamps, resistors, etc. The unit has been provided with a voltmeter and ammeter for AC/DC voltage and current readings. List price is $590.

The new Model 1315D features include:

- Variable Output: 0-25V at 5A
- AC or DC Selection Switch
- Separate Circuit Breakers for AC and DC
- Output Voltage Three-Digit LED Display (Green)
- Output Current Three-Digit LED Display (Green)
- Separate Power Switch with LED Indicator
- Heavy Duty Transformer
- Rugged Metal Case
- 115V at 60 Hz or 230 VAC, 50 Hz, Single Phase
- Three Year Warranty

For more information, contact:
Global Specialties
Web:
www.globalspecialties.com

BUILD A LOW COST, HIGH PERFORMANCE 12 WATT AMPLIFIER FOR EIGHT OHM SPEAKERS

By Ronald Anderson
ronwande@bellsouth.net

Go to **www.nutsvolts.com/index.php?/magazine/article/august2013_Anderson** for any additional files and/or downloads associated with this article. You can also discuss this topic at **http://forum.nutsvolts.com**.

With this unit, you can have a high performance power amplifier for a low cost. The performance actually depends more on the design than on expensive components. Since the time when solid-state amplifiers were first designed using early power transistors, a great deal has been done to improve them. This article will show you how to construct an inexpensive amplifier; the power supply described here will run two channels.

Safety First!

If you have been working with digital circuits and processors, you are probably used to working with five volt power supplies and know the basic precautions to take. This amplifier will have about 35 volts from heatsink to heatsink — enough for a fair shock if you have wet or sweaty hands. So, don't touch both heatsinks at once, in particular one with each hand!

Also, don't work with a ring or metal wristband on. Filter capacitors hold considerable energy and getting a metal band across one might cause considerable heating. Always wrap the fuse and power switch connections with tape or use heat shrink tubing on them before applying any power. This at least limits your exposure to only the AC from the transformer and the DC power supply output voltages.

Parts And Assembly

You may question the use of carbon film resistors since metal film ones are quieter. This amplifier checks out at 105 dB signal to noise. To obtain 12 watts out, we will use a 25 volt CT transformer rated at two amps. You can increase the power output to 30 watts by using a higher voltage and current power transformer. We'll discuss this later. The amplifier in **Figure 1** is a 30 watt version.

The power transistors — leads down, label facing you — have the base, collector, and emitter in that order. The small signal transistors — flat facing you and leads down — are emitter, base, and collector in that order.

Heatsinks might come with a graphite impregnated fiberglass backing. This is a heat conducting layer that conforms to the transistor mounting surface and is NOT an insulator. With it, you don't need heatsink compound. Be sure to debur the hole at the mounting surface.

Sanken transistors have a mounting hole that will clear a 4-40 screw or an M2.5 metric. The plastic case is extremely hard (I suspect it might be ceramic). I tried to drill out the hole just a little for a 6-32 screw and the case cracked; there was no damage to the working parts, it just doesn't look very nice. I used a separate heatsink for each of the two output transistors.

The bias adjust network has the sensing transistor connected on leads as short as possible and the transistor — a Phillips BD139 or a 2SD669 — is mounted stacked with one of the output transistors, with the mounting surface to the face of the output transistors.

Point the leads in the same direction as those of the power transistor. Choose the one that allows the shortest leads. Heatsink compound between the two is desirable. Use sleeving on the three leads of the transistor.

The temperature on the outside face of the power transistor follows the junction temperature more closely and quickly than the mounting surface and heatsink, so this is the best place to measure the temperature.

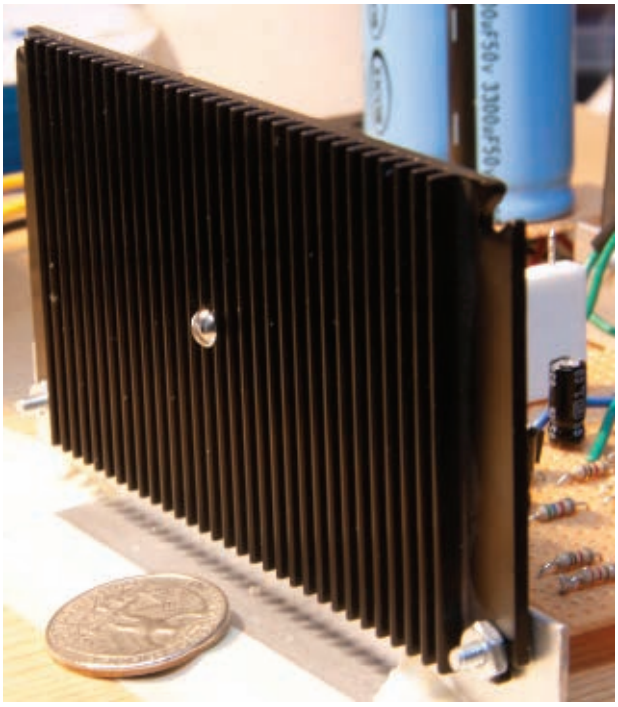
The BD139 is available from BG Micro for 40 cents. The 2SD669 is available from Tayda. They are equally usable. (Hopefully, you can buy one or the other along with an order for other parts.) Any TO-126 size NPN transistor ought to work well here, but I've tested with the two above, with the label facing you; the leads down have connections left to right ECB.

The output transistors should have their collectors wired directly to the power supply filter capacitors using #16 stranded wire. Twist these wires tightly for as much of their length as is practical. Run the ground wire separately; that is, don't twist it with the other two. These power wires each carry essentially half wave representations of the output signal. This signal can couple to the input of the amplifier, greatly increasing the distortion at high frequencies. Twisting the supply wires together as far as possible greatly reduces this problem.

Place the output transistors and heatsinks as far away from the input as possible, but keep the leads other than the power to the collectors as short as possible. The power to the rest of the circuit should be connected to the power supply filter capacitors with ordinary hookup wire. Don't connect the circuit power to the output transistor collector power leads.

It is VERY important to connect the speaker output ground back to the junction of the two filter capacitors in the power supply. Ground to the amplifier must be run separately back to that same "quality ground" point. Also, there are two bypass capacitors; each power voltage to

FIGURE 1. Power amplifier in wood case.



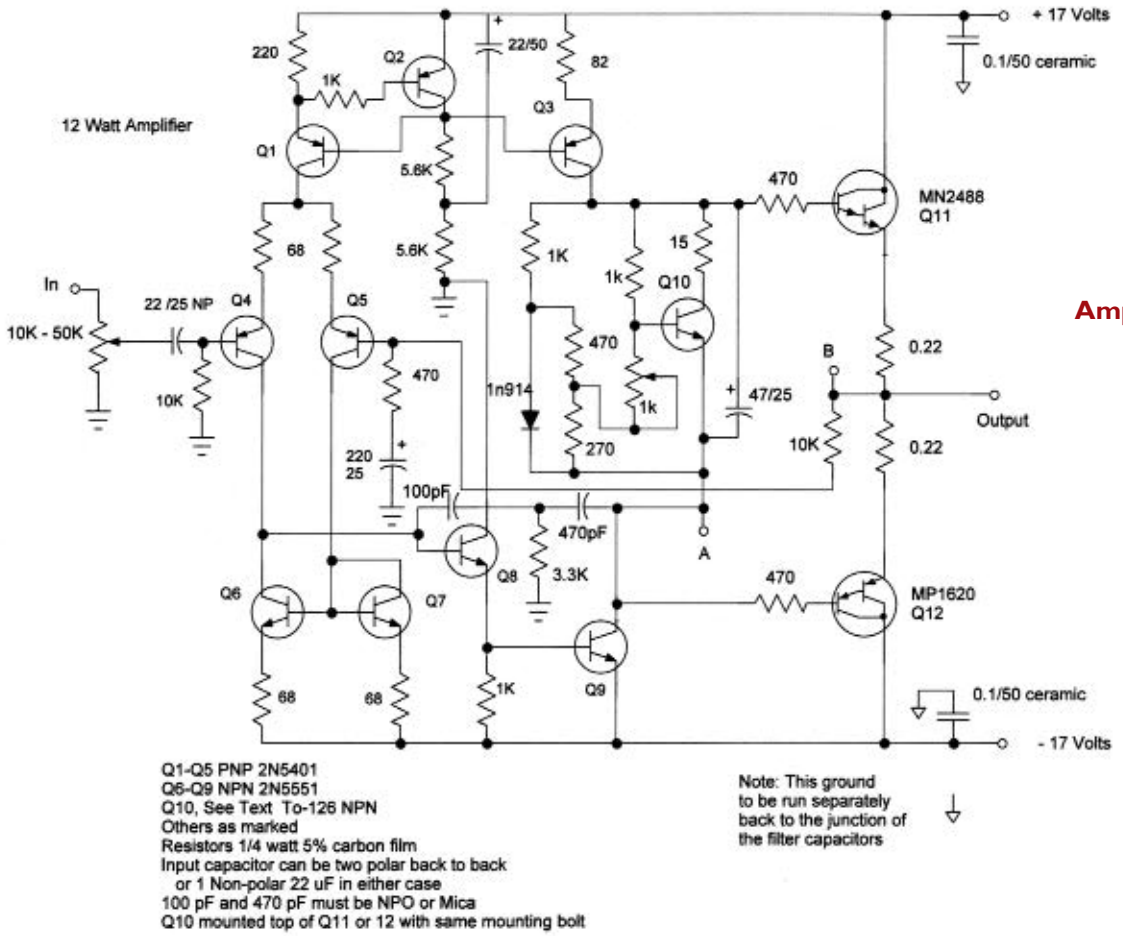
■ **FIGURE 2.This heatsink is adequate.**

ground (0.1 µF ceramic). The ground for these should be run separately back to that quality ground point, as well.

My perf board assembly has an added RCA jack and potentiometer stuck on the corner for testing (**Figure 4**). The 0.22 ohm resistors are in one package with leads running out both ends for convenience in measuring the voltages. Standard single wire wound resistors may be used as well, of course. Mouser has five watt units for 44 cents each.

I placed parts roughly as they appear on the schematic. This makes circuit tracing much easier. The top of the board in **Figure 6** corresponds to the top of the schematic. You will want a power switch and fuse on the primary side of the power transformer; 3,300 µF capacitors are adequate for both versions of the amplifier. By using the larger ones, you will get a watt or two more out of the amplifier before clipping. You will need to use a slow blow fuse because at turn-on, the filter capacitors essentially look like a shortcircuit as they charge. I suggest two amps. A fuse will not protect the output transistors against an output shortcircuit, but will blow if the power supply is shorted.

This power amplifier is Class B. Class B amplifiers introduce what is called crossover distortion. When the output signal is negative, the PNP output device conducts. When it is positive, the NPN conducts. Crossover distortion is generated at the point where the conduction crosses over from one output transistor to

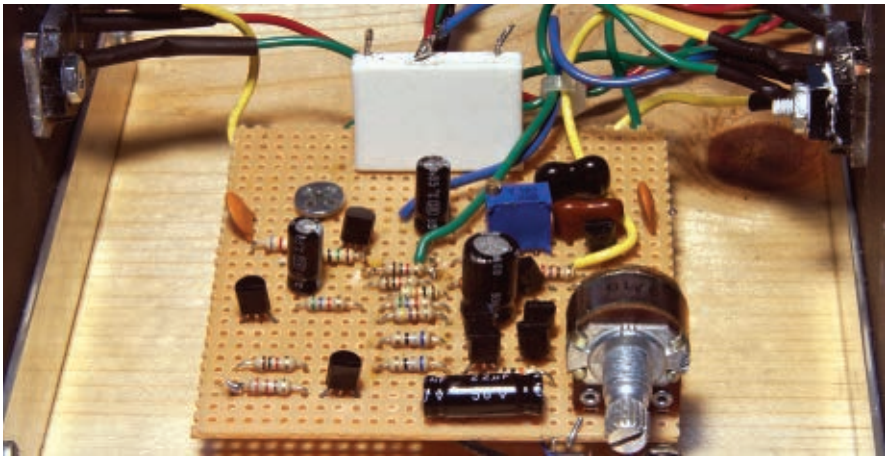


■ **FIGURE 3.**
Amplifier schematic.

the other. Each transistor requires a small forward bias voltage base to the emitter to make it start to conduct — roughly 0.7 volts. There is an optimum quiescent current through the output stage that minimizes distortion. No setting of this current can eliminate the crossover distortion completely, but the remaining distortion is reduced by the overall global negative feedback.

This current is controlled by the bias voltage applied to the two output transistors. The bias control circuit includes a temperature sensing transistor that reduces the bias voltage as temperature increases, attempting to maintain the bias current at a constant level. If this is not done, as the output transistors get hot, the current increases and can reach a condition called thermal runaway. When that happens, the current increases until one or both of the output transistors is destroyed. These output transistors need a sensing transistor that is closely coupled mechanically. (Refer to **Figure 5** again.)

Note that the output transistors have 470 ohm resistors in series with the base connection. These are called stopping resistors. They prevent the output stage from oscillating. I place these resistors right at the base lead of the output transistors and use sleeving over them. Delay connecting the output transistors until preliminary testing is done.



■ **FIGURE 4. Breadboard assembly.**

Preliminary Test

With the output transistors NOT connected, connect point A to point B (see **schematic**). This will be the output point for testing. Before turning on the power, trace your wiring against the **schematic** wire for wire. Measure resistance from the power connections to ground to be sure there is not a shortcircuit in the wiring.

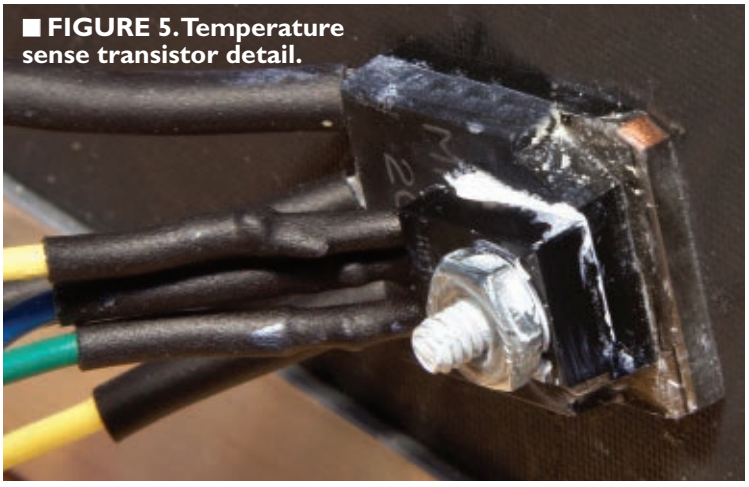
Turn the volume control down all the way. For a first test, don't connect a load of any kind. Turn on the power and measure DC voltage from output to ground. It will either be at one of the power supply voltages or nearly so indicating a problem in the wiring; or, it will be within 10 or 20 millivolts of ground indicating proper operation. If it is high, turn off the power and trace the circuit again. By not connecting the output transistors yet, you have probably avoided destroying any of the low level transistors.

Final Test

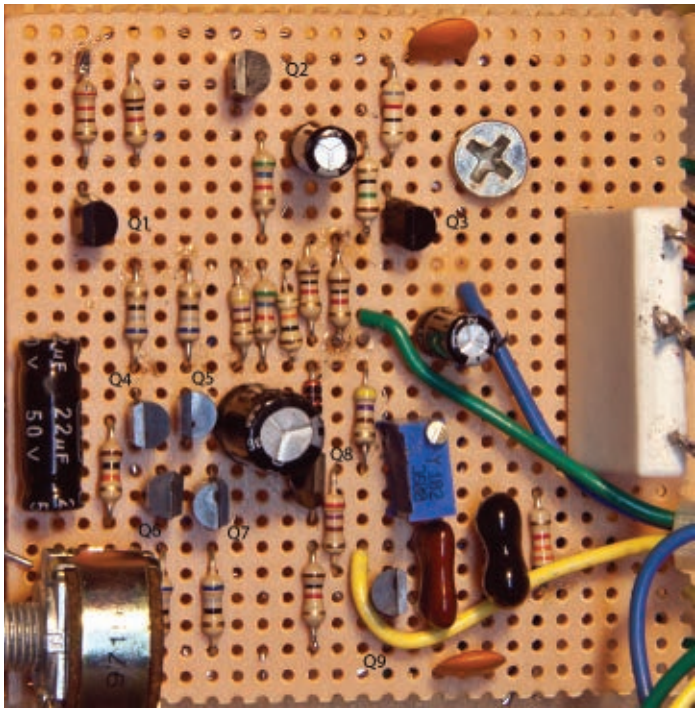
When the voltage checks out okay here, turn off the power and carefully connect the output transistors. Be sure to remove the connection of point A to point B. Measure and set the 1K bias adjustment potentiometer to maximum resistance. Measure because you might have gotten the CW and CCW ends of the potentiometer reversed. Maximum resistance insures a small bias voltage. Now, turn the power on again and check that the output voltage is on the order of millivolts.

Distortion is reconciled for minimum by adjusting the bias voltage potentiometer while measuring the DC voltage across the two 0.22 ohm resistors in series. Setting the voltage to 26 millivolts, the theoretical best value works fine. This adjustment gets my prototype very close to the minimum distortion point. This is a no signal current of about 59 mA (0.026/0.44 = 0.059 amps).

If you use a 20 turn potentiometer (recommended), it will take several turns of the adjusting screw before you see any voltage across the 0.44 ohms. Don't turn too rapidly! Wait a while for the transistors to heat up and readjust the bias potentiometer for 26 millivolts again. The heatsink may get warm but shouldn't get hot with no signal present. Connect the potentiometer as shown back



■ **FIGURE 5. Temperature sense transistor detail.**



■ **FIGURE 6. Perf board with identification of transistors.**

in the **schematic**. The most common potentiometer failure is that the adjustable contact can open. When connected as shown, this only results in a decrease of the bias voltage to the output stage, reducing the quiescent current to zero. Distortion will increase but no other damage will occur. Important: You must measure this voltage with a "floating" voltmeter such as a battery-operated digital multimeter. If you use a bench voltmeter that has the common lead grounded, you will destroy the output transistors or at least one of them.

Distortion (%) at 12 watts	
1 kHz	0.0016
10 kHz	0.0025
20 kHz	0.0042

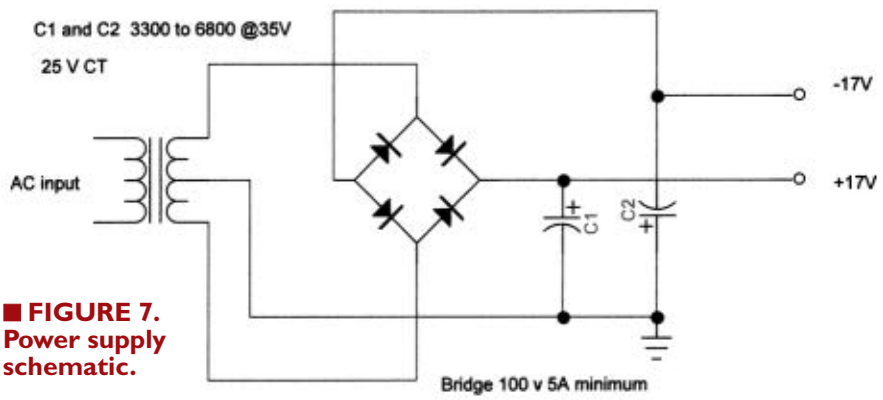
Distortion values will vary somewhat with the exact bias setting, but will very likely be below 0.007% at 20 kHz. This is about 1/16 of what is presently considered audible. Sorry, but space here doesn't permit a discussion of why we are worried about harmonics of 20 kHz (which we can't hear anyway).

The heatsinks get warmer but not uncomfortable to the touch running the amplifier at eight watts output. The transistors actually dissipate more power at four watts output than at 12 watts — which is near clipping. If you have a signal generator and can run a signal at 1 kHz into the amplifier input — adjusting the level for 5.6 volts output into an eight ohm load — set this up and let it sit for half an hour, testing it with a finger now and then. You can make an eight ohm dummy load by using an eight ohm 10 watt resistor.

I found some 10 watt 24 ohm resistors and paralleled three for a 30 watt load. You can test using a 50 or 60 Hz power line frequency by connecting a transformer — perhaps 6.3 volts centertapped — using only half of the winding and turning the input down with the input level control for the four watt output (5.6 volts into an eight ohm load equals four watts) if you don't have a signal generator.

Maximum output before clipping is 12 watts 9.8 volts RMS into an eight ohm load. Depending on your filter capacitors and your local line voltage, the maximum output may vary a little. Smaller filter capacitors allow greater ripple voltage, so the signal can run into the ripple minimum points and cause a sudden and sharp rise in distortion as the signal level increases beyond that point.

You will find the output enough for very loud room volume with any fairly efficient speaker. My JBL speakers produce 86 dB sound pressure level at one meter at one watt. My better speakers are 93 dB. The 12 watt amplifier on the more sensitive speaker sounds as loud as a 50 watt amplifier on the JBL. Frequency response of this amplifier is 3 dB down at 400 kHz. This implies negative feedback of about 26 dB at 20 kHz. This is a safe level for stability of the amplifier. A bit more open loop gain would increase the feedback to 30 dB and reduce the distortion by about 1/3 at the expense of less stability.



■ **FIGURE 7. Power supply schematic.**

More Power?

Does 12 watts seem too puny? How about 30 watts? Replace the power transformer with a Triad 36-T8 from Allied Electronics. It is rated at 2.8 amps 36 volts centertapped, and the supply voltage will be on the order of plus and minus 26 volts. Heatsink to heatsink voltage will be 52, which is more than enough to feel. Before

changing the power transformer, turn the bias control to reduce the bias voltage to zero. Connecting a higher power supply voltage would increase the bias. After connecting, readjust for 26 millivolts. Use a rectifier bridge rated at 100 volts and five amps or more, and 35 volt or higher filter capacitors.

Rectifier bridges are quite inexpensive. There is no reason not to use one that has a substantially higher current rating than necessary. Orient the heatsink with the fins vertically (this is more important with the 30 watt version) and mount the transistor about an inch up from the bottom at the center of the heatsink. Be sure not to place the heatsink on the bottom of the box; that is, leave half an inch UNDER the heatsink so air can flow directly up through the fins. This makes a significant difference in the maximum heatsink temperature.

I have mounted my heatsinks in previous projects on a piece of 5/8 x 5/8 aluminum angle from the local hardware store. Remember that these heatsinks have the supply voltages on them. Don't mount the pair to a metal chassis or to a single angle bracket! Because of the voltage present, you should put the amplifier with heatsinks in an enclosed box. Be sure there are ventilation holes below the fins and also at the top of the box. Be sure to fit the box with feet so air can get to the vent holes at the bottom of the box. With this power supply, you will get close to 32 watts out before clipping.

Distortion (%) at 30 watts	
1 kHz	0.0011
10 kHz	0.0026
20 kHz	0.0042

The distortion values depend a little on the temperature of the heatsinks; that is, whether the amplifier was just turned on or has been running at 20 watts for a while. In this design, distortion remains low from 1 kHz on down to 20 Hz. If you have inefficient speakers or listen to your music really loud, you should use larger heatsinks.

Manufacturers typically use heatsinks that won't take maximum output continuously since music peaks are very far above the average. I've used these smaller heatsinks in my 30 watt version amplifier successfully. It can be run at 10 or 12 watts, which causes maximum power dissipation in the output transistors for 15 minutes without problems.

The listening level will be a fraction of a watt with reasonably efficient speakers. The heatsinks will barely reach body temperature. The extra power is needed for peaks in the music loudness. Note that with this power transformer at full output on one channel, the power supply voltage is 25.5; at 30 watts, the peak collector current is 2.7 amps. The RMS current is just under two amps. The power supply will run one channel at 30 watts but not both simultaneously. This is normal design practice among amplifier companies. Average output power playing music will be well below any problem level.

Last Notes

My home stereo system uses an amplifier of this design. The most usual comment I get from listeners to my system is that it sounds "clean." I've substituted my breadboard of this amplifier for one channel of my stereo. It sounds just as clean as my previously built amplifier. And that's music to my ears. **NV**

12 Watt Amplifier

PARTS LIST

Resistors 1/4 watt 5% except as noted

R1	220
R2, R12, R14,R15	1K
R3	82
R4,R5	5.6K
R6,R7, R9,R10	68
R8,R21	10K
R11,R16	470
R13	3.3K
R17	270
R18	15
R19, R20	470
R23,R24	0.22 (three or five watt)

Capacitors Aluminum Electrolytic

C1	22/25 nonpolar or a pair of 22/25 polar back to back
C2	22/50
C3	220/25
C6	47/25

Ceramic NPO or Mica

C4	100 pF
C5	470 pF

Ceramic

C8,C9	0.1 µF 50

Potentiometers

P1	50K dual ganged for two channels
P2	1K 15 or 20 turn trimmer

Diode

D1	1N914 or 1N4148

Transistors

Q1-Q5	2N5401
Q6-Q9	2N5551
Q10	BD139 or 2SD669
Q11	Sanken MN2488
Q12	Sanken MP1620

Parts Suppliers:

BG Micro in Garland, TX	www.bgmicro.com
Marlin P Jones in Florida	www.mpja.com
Allied Radio	www.alliedelec.com
Jameco in San Jose, CA	www.jameco.com
All Electronics	www.allelectronics.com
MCM Electronics	www.mcmelectronics.com
Mouser	www.mouser.com
Tayda Electronics	www.taydaelectronics.com

BUILD YOUR OWN INDUCTION CHARGER

By Matthew Bates
mattscottbates@gmail.com

Go to **www.nutsvolts.com/index.php?/magazine/article/august2013_Bates** for any additional files and/or downloads associated with this article. You can also discuss this topic at **http://forum.nutsvolts.com**.

As an avid hobbyist, I would like to have a convenient way to recharge my battery powered projects without having to tie up the USB ports on my computer. Borrowing from the concept of the wireless chargers on the market, I decided to build my own. So if you like the idea of having a wireless replacement for your USB port, open up your surplus parts drawer and let's start the induction process.

How Does Inductive Coupling Work?

Wikipedia defines resonant inductive coupling as "the near field wireless transmission of energy between two coils that are tuned to resonate at the same frequency."

The formula for calculating the resonant frequency is:

$$f_r = 1/(2*pi*\sqrt{(LC)})$$

You can use a meter to determine inductance, but not for the distributed capacitance that accumulates between windings. You can use the following formula to determine the self-capacitance or the interwinding capacitance:

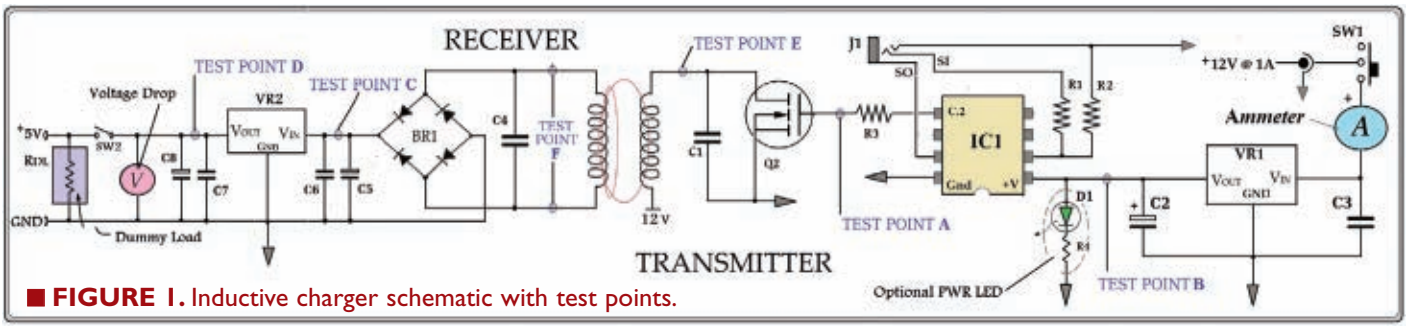
$$C = .29L + 0.41R + 1.94[\ \sqrt{\ }\ (R\text{^}3/L)\ ]$$

Where
C = Self capacitance in picofarads
R = Radius of coil in inches
L = Length of coil in inches

The prototype coil for this project was wound using some surplus wire I had left over from a previous project. The size of the coil was based on a dimension that was indicative of most of my average size projects. The coil was a flat, single layer spiral coil created with 26 AWG enameled magnet wire that had a 1" inner diameter and an outer diameter of 2.5".

The coil was wound with 44 turns and had an inductance of 152 uH with a parasitic capacitance of 1 µF. Using the resonant frequency formula just given, I discovered that the coil would resonate at 12.9 kHz. If you want to use your own coil design, you will need to find the resonant frequency for it.

There are online sites that serve as calculators that can make the job a lot easier; there is one such calculator located at **www.1728.org/resfreq.htm** that can solve for frequency, capacitance, or inductance as long as you have two of the three variables. You may want to start with the coils used in this project before you try using coils of your own design.



■ **FIGURE 1.** Inductive charger schematic with test points.

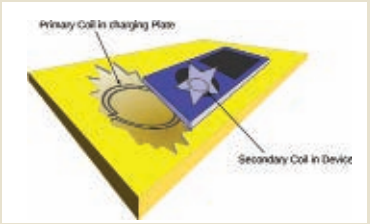
A wireless charging system needs to contain the following circuit elements:

- Any type of oscillator capable of producing the resonant frequency.
- A power transistor to serve as an amplifier for driving the primary coil.
- A set of coils that serve as a primary transmitter and secondary for the receiver.
- A full wave rectifier to convert the incoming AC to a DC value.
- A voltage regulator to create a useable voltage for charging depleted batteries.
- A circuit to manage the charging process for Li-Ion or NiMH battery chemistries.

The schematic shown in **Figure 1** is an example system with test points for troubleshooting possible problems, plus the meter placement that is necessary to calculate power efficiency.

Building The Circuit

Before you can fully test the operation of the transmitter and receiver circuits, you will need to construct a set of coils.



Inductive charging (also known as "wireless charging") uses an electromagnetic field to transfer energy between two objects. This is usually done with a charging station. Energy is sent through an inductive coupling to an electrical device, which can then use that energy to charge batteries or run the device.

Induction chargers typically use an induction coil to create an alternating electromagnetic field from within a charging base station, and a second induction coil in the portable device takes power from the electromagnetic field and converts it back into electrical current to charge the battery. The two induction coils in proximity combine to form an electrical transformer.

Greater distances between sender and receiver coils can be achieved when the inductive charging system uses resonant inductive coupling. *Courtesy of Wikipedia.*

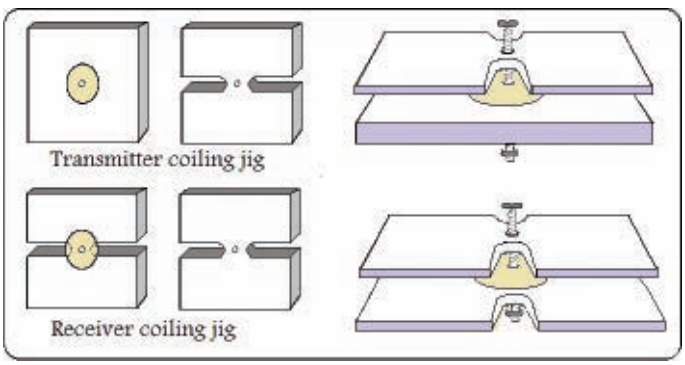
Creating The Coils

If you are going to design your own coils, try experimenting with varying diameters of wire, coil geometries, and different coil sizes. The following is a description of a coil design technique that is the culmination and the distillation of many moons of effort in the application of a single method.

The coil construction can be the most difficult part of this project. The suggested coils for this project are a flat pancake style that is reminiscent of the old Tesla primary coil design. They can be almost impossible to fabricate without a specific technique. I have tried numerous ways to create these coils; the method I discuss here provides the most consistent results.

You will need two acrylic blocks per coil. The blocks should be of a thickness that makes it difficult to deform them. I find that about 1/4 ″ thick acrylic is fairly rigid under stress. You can find prefab blocks at most well stocked craft stores; they are typically used for making stamping tools. I found the ones I used at Michaels craft supply, but they can be ordered from various places online.

The only problem with prefab blocks is the lack of variety of dimension. The blocks I used are 2.5″ square which works fine given the dimension of the circuits that I would like to make wirelessly rechargeable. For the transmitter and receiver coil, you will need two sets of block configurations illustrated in **Figure 2**.



■ **FIGURE 2.** Winding jigs for transmitter and receiver coils.

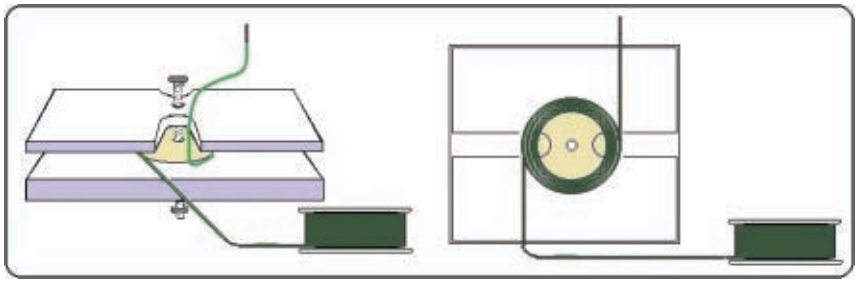


FIGURE 3. Winding the transmitter coil.

Cut a 1″ diameter disc from any mylar material. The disc thickness needs to be the same thickness as your wire. I had some 26 AWG enameled magnet wire from a previous project, but any gauge wire (within reason) will work. Drill a 3/16″ hole in the center of the two acrylic blocks and in the center of the 1″ mylar disc. To make the "U" shaped cut-outs, drill a 1/4″ hole that straddles a portion of the 1″ disc as shown. With a dremel cutting wheel or hacksaw, cut the block from the edges to the 1/4″ hole so that it matches the shape in **Figure 2**. Using a machine screw, make sure the pieces can be assembled (refer again to **Figure 2**). Insert one end of the wire as shown leaving about 6″, and wind the coil as illustrated in **Figure 3**; keep a slight tension on the wire as you wind.

Wind the coil until it just reaches the edge of the block. Cut the wire leaving

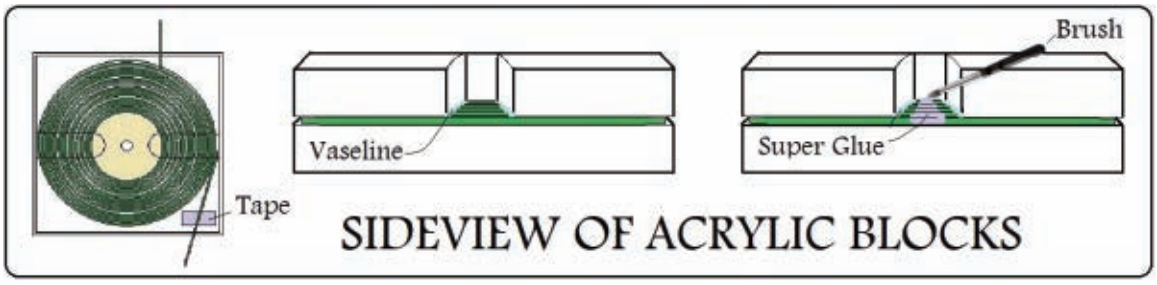


FIGURE 4. Applying glue to freeze finished coil design.

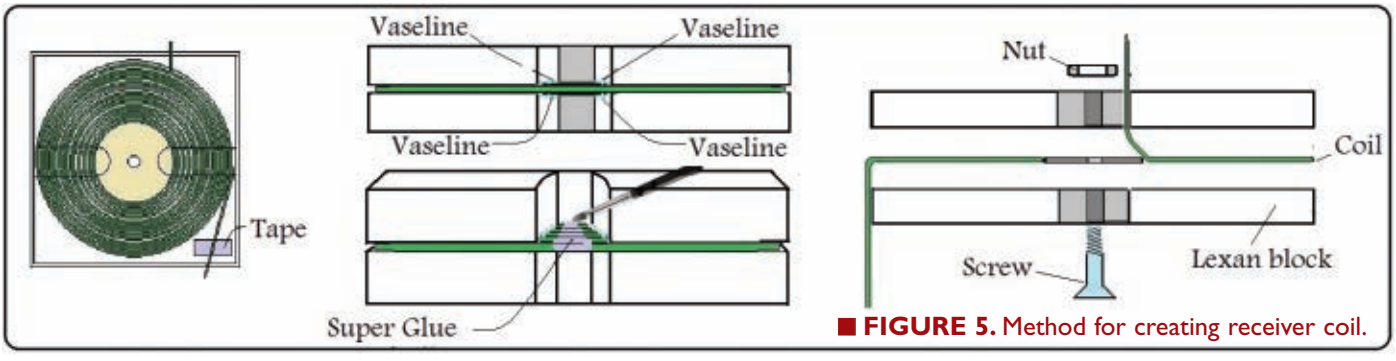


FIGURE 5. Method for creating receiver coil.

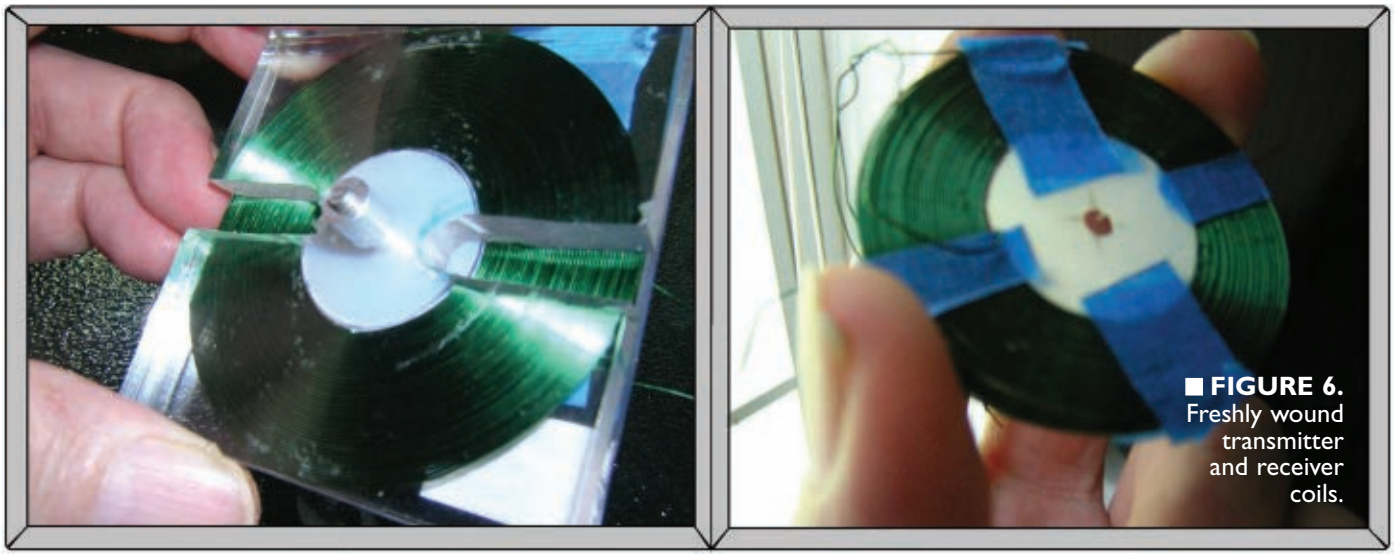


FIGURE 6. Freshly wound transmitter and receiver coils.

six inches on this end, as well. Tape the end of the wire to one of the blocks to keep the coil from unraveling. With a small brush or toothpick, apply Vaseline to the intersection of the cut-outs in the plastic block to the coil as shown in **Figure 4**.

Apply Super glue between the edges of the U shaped cut-outs with the glue applicator brush, also shown in **Figure 4**. The Vaseline will prevent the glue from attaching to the edges of the plastic block cut-outs.

Once the glue dries, disassemble the jig and you are left with the coil glued to the block. This will serve as the transmitter coil in the charging base.

The receiver coil is fabricated much the same way except that you will use cut-out acrylic blocks on the top and bottom as shown in **Figure 5**. Vaseline all four intersections of the coil to the acrylic block and glue the coil the same way as the transmitter coil. Once dried, disassemble the jig as shown in **Figure 5** and you will be left with just a flat pancake coil. Leave the disc in the center of the coil.

You might want to glue more of the coil area after separating it to make it more stable. This coil will mount onto the receiver board, along with the rectifying parts and voltage regulating electronics.

When finished, you should end up with a transmitter coil glued onto the top of one of your acrylic blocks (refer to **Figure 6**). The receiver coil should not be attached to either of the acrylic blocks, and the 1" diameter mylar disc should remain in the center of the coil for easy mounting to the receiver card. Both coils should measure approximately one ohm of resistance.

Once you have finished with the coils, we will begin by breaking the **schematic** (**Figure 1**) into the construction of individual transmitter and receiver circuits. I do recommend building both circuits on separate breadboards before committing your design to a final circuit board.

Building The Transmitter Circuit

The transmitter requires a 12V power source capable of delivering one amp. The PICAXE operates from 2.4V to 5V, and will require a voltage regulator to produce a voltage in this range. Use either a 3.3V or 5V regulator such as the LM2950 or LM7805. The PICAXE 08M2 microcontroller serves as the oscillator that generates the resonant frequency. The output of the 08M2 is fed to the gate of a MOSFET power transistor that drives a coil directly from its drain. A snubber capacitor from the drain side of the MOSFET to ground is included to prevent damaging the MOSFET from inductive kickback during turn-off transitions. The back-EMF can be quite considerable (10x the input voltage), even with air core transformers. The best capacitor to have here is an MKP grade that is often used when generating high current pulses, but a higher voltage metalized film capacitor (MPF) will suffice. An ammeter should be placed as shown in the **schematic** to measure the input current consumed by the circuit in order to calculate efficiency.

The PICAXE will need to be programmed to generate the resonant frequency. To do this, add two resistors to your breadboard as shown back in **Figure 1**. Connect your programming cable to the audio jack and load the following lines of code to generate a 12 kHz output with a 50% duty cycle:

BASIC CODE TO GENERATE 12 kHz

```
setfreq m8              'REM sets the operating speed to 8 MHz
do                      'REM beginning of loop
pauseus 1200            'REM creates a 1200 µS pause
pwmout c.2, 153, 308    'REM generates a 12 kHz output
                        '@ 50% duty cycle
pauseus 1200            'REM creates a 1200 µS pause
loop                    'REM End of loop
```

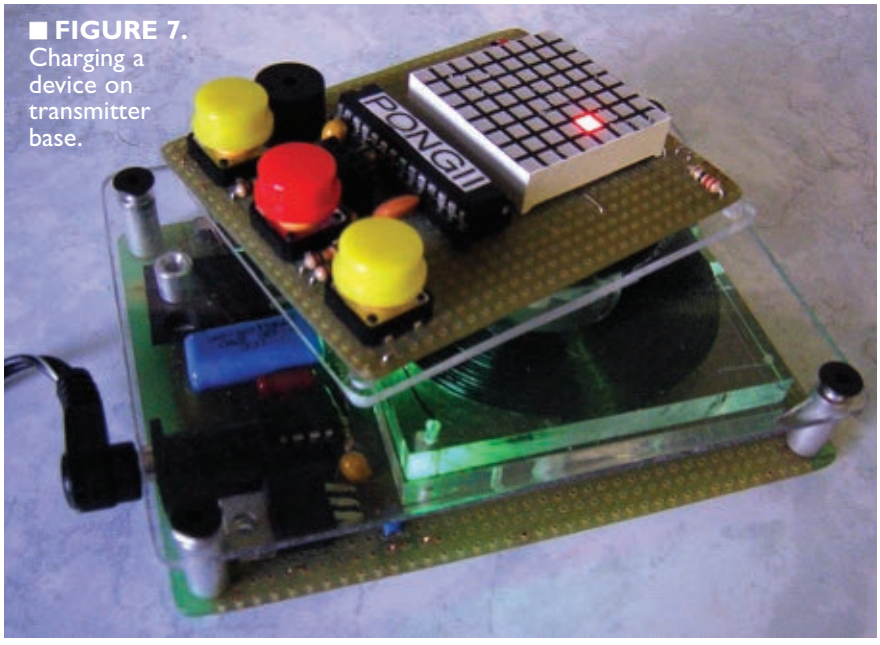
The code to produce any frequency with a set duty cycle can be generated by using the *pwmout* wizard of the compiler and is invoked from the program menu. In the prototype circuit, I placed the "PWR ON" LED in the side of the 1/4" acrylic coil platform. It creates an interesting effect when the circuit is turned on.

Building The Receiver Circuit

Once energy has been coupled to the secondary, a rectifier converts the incoming AC to a DC value. The output may not follow the normal turn ratio and be higher than the input voltage. This is due to ringing on the outgoing wave that gets attenuated on the secondary, causing a rise in voltage. This is not a problem unless it exceeds the 35V input limit of most regulators.

A snubber capacitor of 0.1 µF should be placed between the outputs of the secondary coil to block inductive kickback. Feel free to use either discrete diodes or a packaged bridge rectifier in the design. Be sure that the devices you implement can withstand one amp of current at 50V. The DC output is regulated to 5V using an LDO regulator such as an LM78L05. It is very important to use the LDO version regulator to provide a source of constant current and constant voltage, much like USB output.

To measure the output power of the receiving circuit, place a resistive short across the regulated 5V output that can be switched in using a SPST slide switch as shown in **Figure 1**. Use a meter to read the voltage drop across the resistor. Using Ohm's Law, you can compute the power output by I = E / R. Use a resistance value with a base of 10 to make calculations easier. Be



■ **FIGURE 7.** Charging a device on transmitter base.

CHARACTERIZING POWER OUTPUT & EFFICIENCY

FDH055N15A — N-Channel Power Trench MOSFET 150V, 167A, 5.9 mΩ
COIL DIAMETER = 2.5 | APERATURE = 1" | SEPARATION = 0.25"
FREQUENCY = 12.9 kHz DUTY CYCLE = 50%
Input Voltage = 12V Output Voltage 5.06V (31V unregulated)
Voltage Drop across 10 ohm load shorted = 0.710V (I = E / R) 710 mA
Input = 900 mA Output = 710 mA Efficiency = 710 mA / 900 mA * 100 = 78%

sure to use an adequate wattage resistor for the dummy load. By generating current values close to one amp, you will need a resistor rated at 5W.

Testing Your Circuit

When breadboarding certain power transistors, it may be necessary to attach smaller diameter wires to the leads in order to plug into your breadboard. You will also need a way to intercept the (+) lead from your power supply to attach the ammeter.

Connect the coils to your breadboard circuits and attach the meters as shown back in **Figure 1**. Place the receiver coil over the transmitter coil, separating them with one of the acrylic blocks to act as an insulator. Apply power to the transmitter circuit and record the values from both meters. Close SW1 to short the dummy load across the output of the regulator.

You should notice the input current value rise due to the short being reflected back to the primary. You may need to heatsink your power transistor. If it becomes excessively hot at resonance, you need to check your work. Try the suggestions given in the Troubleshooting section first.

It's very simple to add a recharging receiver to your projects. The following is an example of a battery powered project that I converted to be wirelessly rechargeable. I took an existing project that is an 8 x 8 LED matrix Pong game that is powered by a lithium-polymer battery source. The game has a footprint of 3" x 2" with a battery supply on the back of the board. I mounted the receiver coil on a board with the same dimension as the game, leaving plenty of room for the electronics in the receiver.

I wanted to keep the receiver card as thin as possible to not add much more depth to the existing project. **Figure 7** is a photo of the charging receiver attached to this project design that I want to charge wirelessly.

The entire receiver board adds only an additional 1/4" to the depth of the project. A single IC battery charging manager shown in **Figure 8** is connected to the output of the 5V regulator. This chip (manufactured by Maxim Integrated) requires just a few external components, and will manage the charging for a single cell lithium battery. The MAX1811 has an LED that indicates when charging is complete.

I have been able to get a nominal service life of approximately 400 charges with this device. I even use it to charge my super capacitors.

Troubleshooting

This circuit was intentionally designed to be simple, so troubleshooting should be correspondingly easy. The following are voltages that should be present at the various test points shown in the schematic from **Figure 1**.

1) There should be 5V at TEST POINT B (check the 12V supply voltage if not 5V).

2) There should be approximately 2.5V at TEST POINT A (check the 08M2 power supply or the code).

3) There should be a minimum of 6V at TEST POINT C (check the rectifier or AC across the coil). Test the regulator by connecting the 12V supply to the input terminal.

4) You should have 5V at TEST POINT D (check the regulator connections).

5) There should be 12V AC or greater at TEST POINT E (check the coil connection if the test points in listings 1 and 2 are okay).

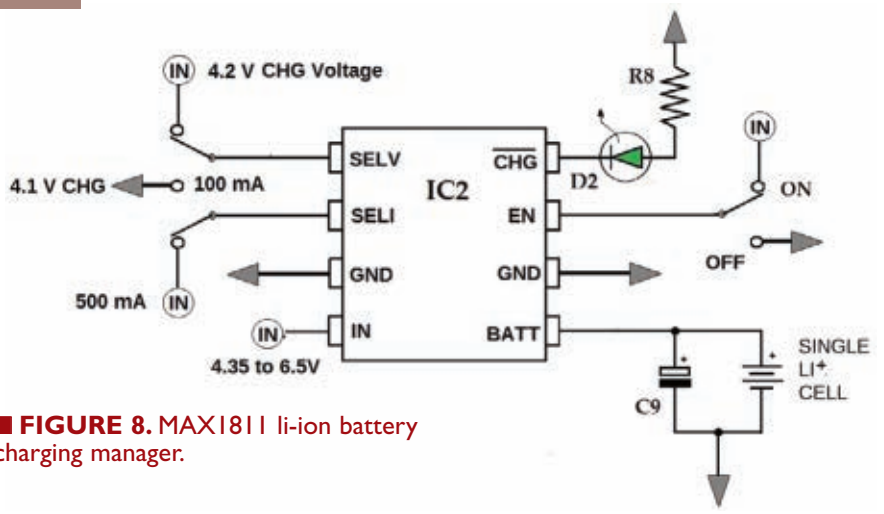
6) You should have an AC value at TEST POINT F (check the secondary coil connection if the test point in listing 5 is okay).

PARTS LIST

ITEM	QTY	DESCRIPTION	SOURCE/PART#
All surface-mounts are 805			
Q2	1	FDH055N15A N-Ch FET (any)	Digi-Key # FDH055N15A-ND
J1	1	1/8" Audio Jack (any)	Digi-Key # 2168131
R1	1	22K ohm 1/4W Resistor	Digi-Key # CF14JT22K0CT-ND
R2	1	10K ohm 1/4W Resistor	Digi-Key # S10KQCT-ND
R3	1	220 ohm 1/4W Resistor	Digi-Key # CF14JT220RCT-ND
R4	1	330 ohm 1/4W Resistor	Digi-Key # A105936CT-ND
RDL	1	10 ohm 5W	Digi-Key # ALSR5J-10-ND
C1	1	0.1 µF MPF Snubber Capacitor	Digi-Key # EF2105-ND
C3,C6	2	0.1 µF Bypass Capacitor	Digi-Key # 1493-3401-ND
C2,5,7	3	10 µF Electrolytic 50V	Digi-Key # P997-ND
C4	1	0.1 µF Mylar Snubber Capacitor	Digi-Key # 495-2435-ND
D1	1	3 mm Green LED	Digi-Key # 751-1101-ND
BR1	1	Bridge Rectifier	Digi-Key # DF005M-E3/45GI-ND
VR1,2	2	LM78L05 or LM2940-N Regulator	Digi-Key # LM2940T-5.0-ND
SW1	1	SPST Slide Switch	Digi-Key # CKN9924-ND
L1,L2	1	Asst Magnet Wire	RadioShack # 278-1345
IC1	1	08M2 PICAXE Micro	SparkFun COM-10803
Optional Parts			
IC2	1	Batt Manager (see text)	Digi-Key # MAX1811ESA+-ND
D2	1	3 mm Green LED	Digi-Key # 751-1101-ND
R8	1	220 ohm 1/4W Resistor	Digi-Key # CF14JT220RCT-ND
PCB	1	4.3 x 6.8" Gen Prototyping Board	Jameco # 206587

MISCELLANEOUS HARDWARE
Four 2.5" x 2.5" x 1/4" acrylic blocks from a craft supply store.
Threaded standoffs for building a transmitter base (RadioShack).
Super glue with brush applicator.
1/8" Lexan
Vaseline

MISCELLANEOUS TEST EQUIPMENT
VOM with 10A scale feature
Oscilloscope (optional)



■ **FIGURE 8.** MAX1811 li-ion battery charging manager.

Possible Enhancements

You will want to come up with a way to sense that an object has been placed on the charging base to keep the transmitter from running all the time. The most graceful way to do this would be to design a current-sensing circuit that trips when a load is introduced.

I currently take advantage of built-in IR commands with the 08M2, and use an IR circuit as a proximity detection system. By using a 08M2 in the receiver, a two-way communication may be desirable between the transmitter and receiver. You may also want to make a large charging surface area.

A simple way to accomplish this would be to arrange transmitter coils wired in parallel. If you make printed circuit boards, you may want to create an etched coil for the receiver that can be scaled to fit the application.

By utilizing surface-mount components, the receiver may take on a near credit card size footprint.

Conclusion

Whether you build this project just to study induction or actually apply it to some recharging end, it is guaranteed to be challenging for both novice builders and experienced ones alike. NV

AUVSI's Unmanned Systems 2013

WASHINGTON, D.C.

WALTER E. WASHINGTON CONVENTION CENTER

CONFERENCE: 12–15 AUGUST
TRADE SHOW: 13–15 AUGUST

With more than 40 product categories and 600 exhibitors, you will find no shortage of innovation, technology and products in the unmanned systems and robotics industry.

Imagine being in one place with the biggest influencers in the industry who all want to share their most valuable knowledge with YOU!

- **Learn** about important issues, advancements and opportunities in the unmanned systems and robotics industry.
- **Choose** from more than 100 technical sessions, panels and workshops featuring the industry's top speakers and leading experts.
- **Explore** the latest technology on the market, including live air, ground and maritime demonstrations, and much more.
- **Mingle** and network at daily receptions, coffee breaks, generous exhibit hall hours and our premier networking event, The Mix.

WILL YOU BE THERE?

Register today at **auvsishow.org**

A Mathematics Engine For Microcontrollers

By Thomas Henry
studs.kreitzer@gmail.com

Go to **www.nutsvolts.com/index.php?/magazine /article/august2013_Henry** for any additional files and/or downloads associated with this article. You can also discuss this topic at **http://forum.nutsvolts.com**.

Even though microcontrollers seem pretty magical, when it comes to computations they're really just simpletons (if fast). Most can only handle basic integer arithmetic, and only over a fairly limited range of numbers. What's needed is a "coprocessor" onto which we can offload any serious mathematical work. Would you believe such a contrivance is often available at garage sales?

In this article, we'll explore how to connect a typical microcontroller to the TI-83 Plus graphing calculator. It's not really a coprocessor in the usual sense, but rather an auxiliary outboard device the microcontroller can call upon for help when the numbers get nasty.

The TI-83 Plus — by some accounts — is the biggest seller in the Texas Instruments lineup. For better or for worse, just about every middle school or high school student owns one. As a consequence, it frequently shows up at flea markets, rummage sales, and second-hand resellers on the Internet. Even if the unit has a damaged keyboard or LCD screen, it is still usable in this super-cool project.

If you're not familiar with the TI-83 Plus graphing calculator, let me mention that it easily handles 14-digit floating point calculations, roots, exponentials, logarithms, trigonometric functions — even matrix algebra, complex numbers, and numerical calculus. All of these features become instantly available to your microcontroller projects. But there's more ...

Data logging is a piece of cake — thanks to the large memory of the calculator — and the data can be stored in vector or matrix form for further manipulations. If statistical analysis is up your alley, then look for the seven different probability distributions within the calculator. Maybe the most intriguing application of all is direct manipulation of the calculator's LCD screen by pixel or by character for meaningful visual displays.

If this all sounds inviting, then let's tuck in and see how to make a microcontroller and the TI-83 Plus talk nicely to each other.

The Connections

Here's some welcome news. It's not necessary to hack into the calculator or modify it in any way to take advantage of what it has to offer. The TI-83 Plus already sports a jack allowing us communication access. (The manufacturer intended this for downloading software, but we'll use it for much more.)

This port is a two-wire affair governed by a somewhat unusual hardware protocol to be explained in just a moment. But first, let jump in with both feet and see what's going on electrically.

Figure 1 is the schematic for a test circuit which will hold you in good stead for many explorations. I went with the common and inexpensive PIC16F88, but I see no reason why what follows here wouldn't work with just about any microcontroller sporting configurable bidirectional port lines.

The plug which connects to the TI-83 Plus is a small 2.5 mm stereo mini-phone type. Actually, the calculator comes with a patch cord terminated by a pair of these. Since it's not really needed for much of anything else with only one calculator, I cut the cable in two and used one piece to make the PIC connection. If you do likewise, you'll note that the red wire in the cable connects to the tip, while the white wire connects to the ring. Ground, of

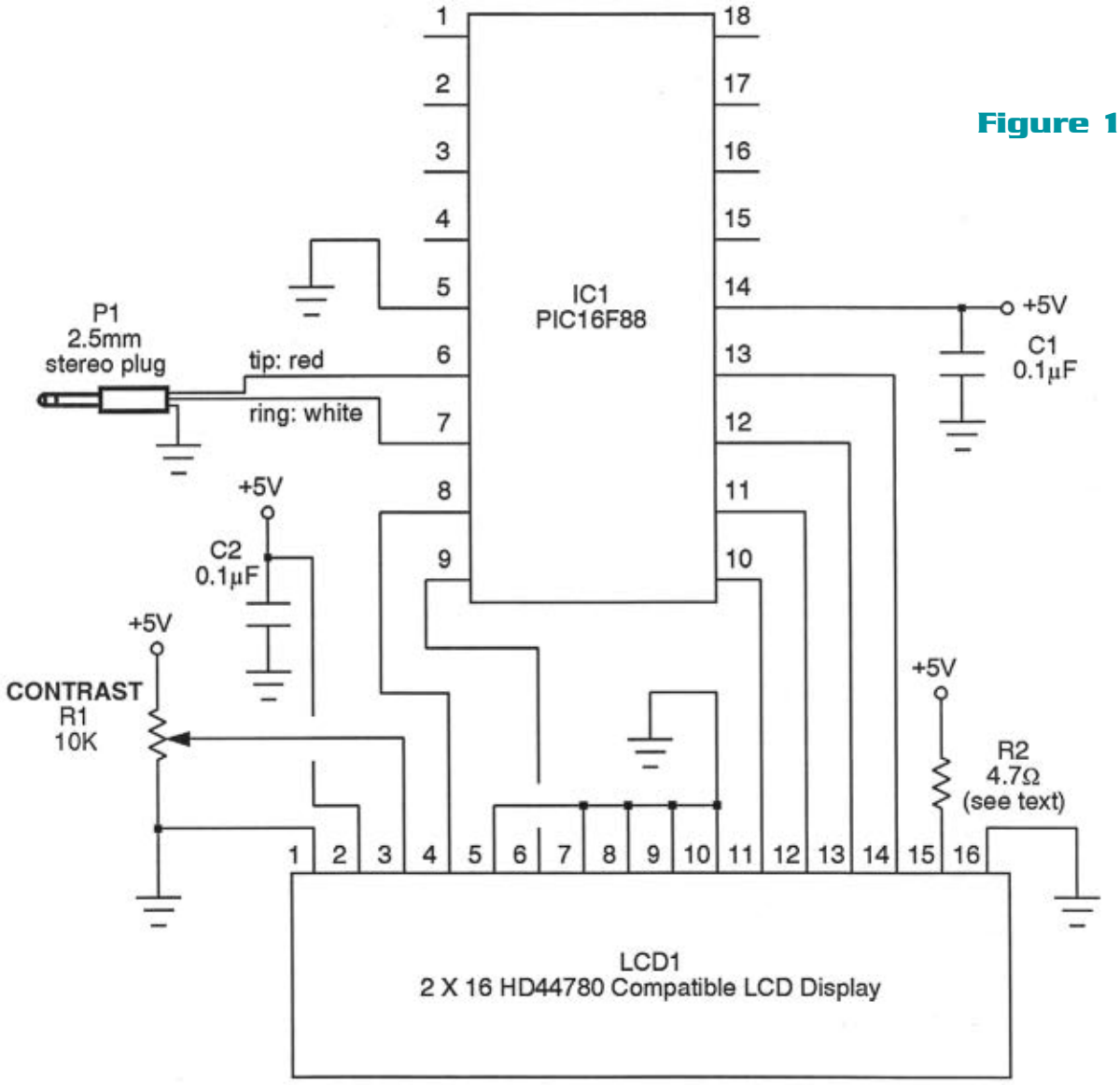


Figure 1.

course, is provided by the braided shield in the cable.

So, the tip connects to port line B.0 (pin 6), while the ring goes to B.1 (pin 7) of the PIC16F88. Observe how uncomplicated everything is. This is a direct connection and no additional circuitry is called for!

Just so we can see what's going on, let's also attach an LCD to the PIC as shown in the **schematic**. There's nothing new or unusual here, as this is the typical way to connect an LCD for four-bit operation. You'll note R2 which is the current-limiting resistor for the LED backlight of the LCD. Just use whatever value is specified in your unit's datasheet.

Well, that was pretty painless. If you've got it all breadboarded as shown in **Figure 2**, let's see how to make the bits flow unabated between the PIC and the TI-83 Plus calculator.

Low Level Communication

Since only two wires are used to communicate between the devices, you might guess there's some fancy footwork involved ... you'd be right. The data transmission is bidirectional; the PIC sends a computation to the calculator, which then returns the result amidst a flurry of handshaking. Plus, everything must happen asynchronously. That's a lot to ask of two wires, but it all works out quite handily. In what follows, we'll call these lines Data0 and Data1.

During quiescence, the data direction of Data0 and Data1 are both set to input mode and will idle in a high state — thanks to the electronics of the calculator. To send a zero from the PIC, make Data0 an output line and bring it low. Now, the PIC waits for the calculator to acknowledge this with a Data1 low response.

While it might sound a bit chatty and redundant, the PIC turns right around and acknowledges the acknowledgment by bringing Data0 high again. If the calculator is happy with this state of affairs, Data1 goes high in return. We're back where we started with both lines idling high as inputs.

To transmit a one from the PIC to the TI-83 Plus, mimic these steps but just reverse the roles of Data0 and

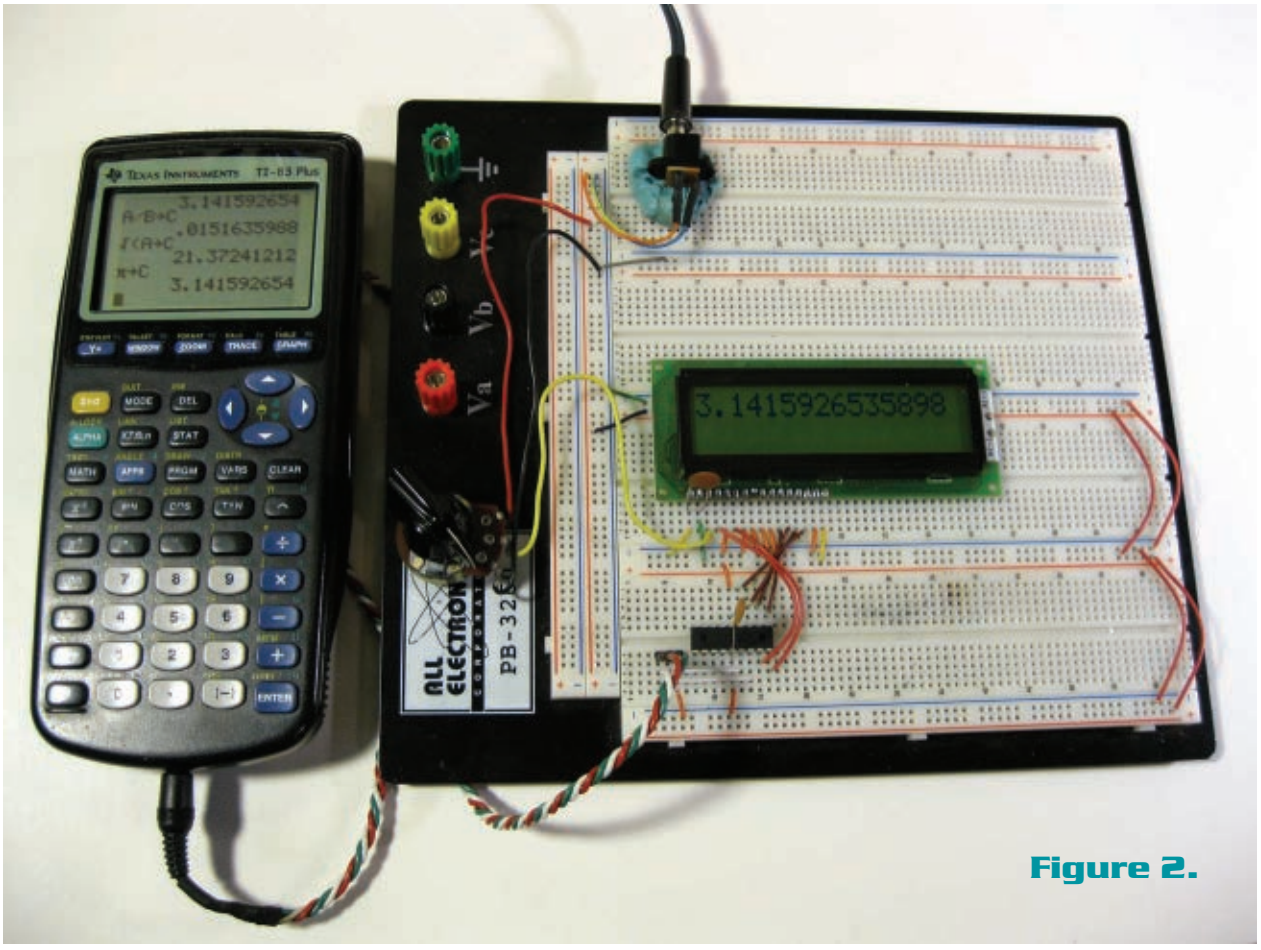


Figure 2.

Data1. To recapitulate, then, Data0 always transmits a zero while Data1 always transmits a one, with the alternate port line covering the hardware handshaking. Yes, it seems a trifle convoluted at first but the choreography is actually fairly straightforward and reliable.

To receive a zero from the calculator, just follow these next steps. As usual, Data0 and Data1 are idling as inputs and in a high state. When the TI-83 Plus is ready to send a zero, it will bring Data0 low. This alerts the PIC to make Data1 an output which it sets low as an acknowledgment. The calculator responds by making Data0 high again. Finally, when the PIC is content with how things have gone, it sets Data1 as an input. Once more, both port lines are idling in a high input condition. Receiving a one is very similar, with the roles of Data0 and Data1 reversed.

So, we know how to transmit and receive a single bit between the PIC16F88 and the TI-83 Plus calculator. All that's required now is a pair of software routines to bag up everything and allow entire eight-bit bytes to be communicated. To save you the travail involved in that task, I've written a collection of library routines which takes care of all of the messiness for you. You can get it (called TI-LIBRARY.GCB) in the download files for this article.

For now, take a moment to spy the two low level routines of interest: GETBYTE and SENDBYTE. These superintend the details for you. Just pass in or receive byte parameters and away you go. The source code (written for the free Great Cow Basic compiler) has been heavily commented, so understanding how it all works is straightforward. Remember, the syntax of Great Cow Basic is very similar to commercial compilers or, for that matter, the interpreted Basic of the PICAXE line of microcontrollers. Porting it to some other chip shouldn't be much of a botheration at all.

Time For A Test

We really should pause to test the essential send and receive functions before moving on to the more complicated business of transmitting entire commands. So, compile and burn the demonstration program DEMO1.GCB into a PIC16F88. You'll note that this program leans upon the library routines just mentioned. Then, wire up the circuit shown in **Figure 1** and plug it into the TI-83 Plus calculator. For this simple test, we don't really need the LCD, so you can leave that portion off if desired.

Here's the plan. The calculator will be running terminal software. When you press a key on it — say the letter A — that will be transmitted to the PIC, which then turns right around and sends it back to the TI-83 Plus.

You'll see the results on the calculator's LCD screen. So, it's really nothing more than a send-and-echo business, but will give you confidence that the one-byte character communication is working well in both directions. (See the **sidebar** for information on free terminal software for the TI-83 Plus which is great for this experiment.)

If everything has worked out for you, then we're finally ready to start sending complete commands and make that calculator obey our every wish.

High Level Communication

So, we now know how to send and receive single bytes. Strings of these are assembled to create entire message packets which convey values, variables, commands, and results. This is done by means of a rather complex language called the TI Link Protocol.

Consider that your calculator has dozens of menus and hundreds of commands within it, and you'll appreciate why the TI Link Protocol is so extensive. In a nutshell, every single feature of the TI-83 Plus is accessible via remote control now, so the associated language must necessarily be quite large. Fortunately, you really only need to know a handful of commands to get started having fun.

I gather that the TI Link Protocol was originally intended to be a proprietary, in-house affair. However, several enterprising sleuths put in a huge amount of work experimenting, observing consequences, collecting results, and writing them up in a final document available now to the general public. Entitled the *Link Protocol Guide v1.4*, the primary authors are Tim Singer and Romain Liévin. You'll find it available for free download at **www.ticalc.org/archives/files/fileinfo/247/24750.html**. This really is a massive treatment explaining the ins and outs of the language very well. I'll focus on a few of the commands needed to employ the TI-83 Plus as a mathematics engine with the PIC, but be sure to pour over the complete document to get even more ideas for some real microcontroller voodoo.

Let's begin with what's known as "silent mode remote control." All of the actions your calculator normally carries out by means of keystrokes or menu selections can also be initiated with the cable connection detailed previously.

A desired command is formatted as a packet made up of several bytes. Let's say we wanted to cause floating point division to occur. The packet would consist of 0x23, 0x87, 0x00, and 0x83 — four bytes total. (Transmit the individual bytes using the *SENDBYTE* routine described above). The first (0x23) tells the TI-83 Plus that an external microprocessor is originating the packet, while the second byte indicates that a command is coming in; not a number, variable name, or some other entity. The last two bytes — which really form a 16-bit word — signify that the PIC is requesting a division. This is followed up with a bit of software handshaking from the calculator, confirming the command was received properly.

To make this painless, the library I've written contains a

A Preliminary Test

Getting a calculator to communicate with a PIC is a moderately complex business with plenty of opportunities for things to act up. When they do, it can be difficult to determine whether it's the hardware or software that's at fault. For that reason, I began this project with several simple experiments just to convince myself I understood what was going on. Here's a great first test to give you confidence you're nailing the basics down.

In this experiment, we'll simply tie the TI-83 Plus calculator and a PC together, and confirm that they can send data back and forth. Both devices will be running terminal software and so will be talking ASCII to each other. Press a key on the calculator and it'll be reflected on the PC. Then, press a key on the PC and look for it to be echoed to the TI-83 Plus.

The connection is by means of the commercial TI-Graph Link serial gray cable. This is manufactured by Texas Instruments and available for about $15 from Amazon and other dealers. Attach the mini-phone plug to the calculator. On the other end, connect the DB-9 plug to an RS-232 COM port on your PC. If you don't have such a fitting, there are lots of inexpensive USB-to-COM adapters kicking around. Again, Amazon is a good source.

Now, you need to load up some terminal software on both the TI-83 Plus and the PC. Let's focus on the calculator first. Rather unbelievably, there exists a very nice and free terminal program for the TI-83 Plus. In order to run it, though, you need to first load in a software shell (also free), so let's begin there. Go to **joewing.net/projects/ti83/ion** and download the shell entitled ION 1.6. Just follow the easy installation instructions and you'll be set for the next step. Incidentally, ION 1.6 was written by Joe Wingbermuehle.

Next, return to the Web and download Telnet 83 Plus SE 1.9 from **www.ticalc.org/archives/files/fileinfo/178/17878.html**. This is a full-featured terminal program. It really is amazing to see a handheld calculator doing things that a mere 30 years ago were only being carried out on mainframes. Again, follow the included instructions to install and use Telnet 83 Plus. The authors in this case are Justin Karneges (who did the original implementation) and Hideaki Omuro (who ported it to run under the ION 1.6 shell).

Now, let's turn to the PC. I like using the terminal software written by Br@y++ (that's not a misprint) and available for free download at **https://sites.google.com/site/terminalbpp**. There's nothing to install here. Just run it and away you go.

Here's an easy-to-miss necessity. The TI-Graph Link cable is powered by the RTS line on your computer. (That's part of the RS-232 protocol). So, in the PC terminal program, be sure you've enabled that line. In the terminal of Br@y++ I'm using, there's a button to let you do exactly that. Conclude by making the communication settings 9600 baud, eight data bits, one start bit, one stop bit, and no handshaking.

If you made it past all that (and it took me several days to sort it out myself), then the calculator and the PC should be talking nicely to each other. Confirm that you can send ASCII text back and forth reliably. Just this experiment alone ought to give you some ideas for future projects.

routine called *SENDCMD* which accepts the 16-bit word representing the requested action and packages it appropriately for calculator consumption. You'll find a list of all the possible actions and their ID numbers in the TI *Link Protocol Guide* mentioned earlier.

So, what about variables and values? Well, the TI-83 Plus expects numbers to always be in a specific format. For example, the quantity that we would ordinarily write as 123.4567 is represented within the calculator as:

0x00, 0x82, 0x12, 0x34, 0x56, 0x70, 0x00, 0x00, 0x00

Do you see the pattern? The first byte gives the sign 0x00 for non-negative numbers, and 0x80 for negatives. The second specifies the exponent of the power of 10 if we're to write the number in scientific format, but normalized to 0x80. Thus, 0x80 means an exponent of 0, 0x81 is for 1, and so on up to 0xE9 for an exponent of 99 — the largest you can go. On the other hand, 0x7F would signify an exponent of –1, 0x7E would be –2, on down to 0x1D which means –99.

The remaining bytes are the packed binary coded decimals (BCDs) of the mantissa; 14 digits in all. The implied decimal point is between the first and second digit.

This is a great system for the calculator, but not necessarily for humans. But again, the library comes to our rescue. I thought it would be pleasant to be able to read and write numbers in everyday decimal fashion and as ordinary strings. (Yes, unlike many other Basic compilers, Great Cow Basic handles strings and does so very elegantly.) So, for the above example, you could simply specify the string "123.4567" in your PIC code and the library routines will automatically take care of massaging that into the form required by the TI-83 Plus.

As for loading and reading variables, look for the subroutines called *SENDNUM* and *GETNUM* in the software library for this article. To stuff a value into a variable, simply set the global variable *NAME* equal to the ASCII value of the name (0x41 for A, 0x42 for B, and so on), and set the global string variable called *VALUE* to the desired decimal number. Then, invoke *SENDNUM* and lo and behold! Your calculator has that variable and value all set to go.

The subroutine takes care of packing the BCDs, setting exponents, etc., as well as the extensive handshaking and checksum business required. It's all very easy to do now. *GETNUM* works in a similar way to retrieve a value from the TI-83 Plus, managing the software handshaking, unpacking the number to a string, and so on. Keep in mind that you'll be able to manipulate not only integers, but floating point numbers

spanning –9E99 to –9E99, and with up to 14 digits of precision. The PIC has become a mathematical savant now!

The Big Demonstration

So, we've met the required subroutines now:

SENDBYTE
GETBYTE
SENDCMD
SENDNUM
GETNUM

Let's see what they can do. If you haven't already beaten me to the punch, compile the DEMO2.GCB source code and burn it to the PIC16F88. (Or port the code over to your microcontroller of choice).

Here's what you'll witness when you fire up the circuit: 456.78 will be divided by 30123.456 (not the sort of thing you do every day on an integer based microcontroller), and the result of 0.01516359875838 is displayed on the PIC's LCD. Then, the square root of 456.78 is computed and exhibited. Next, a 14-digit approximation to the constant PI is displayed. The demonstration concludes by drawing a circle on the calculator LCD — all under PIC control.

Final Calculations

If you're used to being hamstrung by word-based integer arithmetic, then even this simple demonstration program seems pretty incredible. If you're not so easily impressed, however, remember we've only scratched the surface.

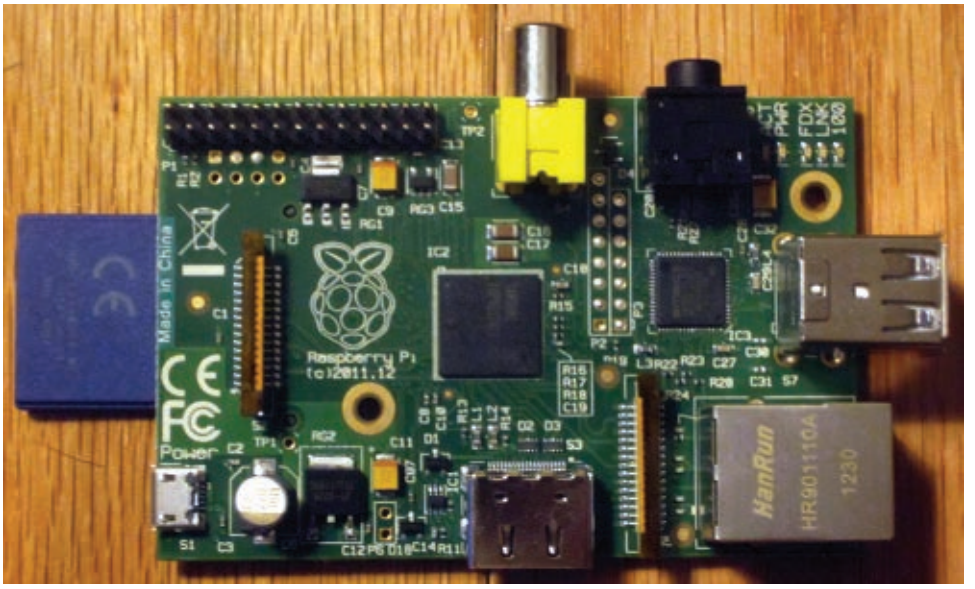
With the hardware and firmware described here — along with the encyclopedic TI *Link Protocol Guide* — a whole new world of trigonometry, exponentials, logarithms, complex numbers, statistics, matrix algebra, and more awaits you!

Now, go figure it out! NV

More Raspberry Pi, Anyone ? Part 2

By Craig A. Lindley
calhjh@gmail.com

Go to **www.nutsvolts.com/index.php?/magazine /article/august2013_Lindley** for any additional files and/or downloads associated with this article. You can also discuss this topic at **http://forum.nutsvolts.com**.



The Model B Raspberry Pi.

In the March 2012 issue of *Nuts & Volts,* I wrote an article entitled "Build Your Own Wi-Fi Internet Radio" about how to build an Internet radio using a repurposed wireless router. While this project worked fine (and still does), it had limitations driven mainly by the small amount of RAM available on the router. After publication of that article, I continued to evolve the design so that it could also play MP3 files stored on a Flash/thumb drive, as well as function as an Internet radio player. I liked the idea of having the music I was currently interested in on a Flash drive and being able to change it easily. Also, since the capacity of Flash drives was going up and the prices were going down, I could have a fairly large number of songs available at one time. Then, to put frosting on the proverbial cake, I could control the Internet radio/music file player with a new iPhone/iPod Touch app I was developing called MpdClient.

As time passed, I became frustrated with the fact I had to convert all of the songs I wanted to play on my Internet radio/music file player to MP3s to play them. So, I started looking for a better solution that would allow me to play many different types of music files. When the Raspberry Pi came on the scene, I figured out how to run a full featured version of MPD which solved all my problems. Now, while I'm sitting in my favorite chair, I can listen to Internet radio stations or play artists, albums, or individual songs from a Flash drive plugged into my Raspberry Pi.

In this article, I will show you step by step how to turn your Raspberry Pi into a remote-controllable Internet radio/music file player.

Hardware

With the high level of system integration on the Raspberry Pi (RPi), you can make an Internet radio/music file player with fewer parts than what would be required using a repurposed wireless router. **Figure 1** shows a block diagram of the proposed system; **Table 1** gives some specifics as to the components required. For our purposes here, I will assume your RPi will be connected to an existing audio system (through an appropriate stereo interconnect cable) or used with headphones, so we won't worry about discussing amplifiers or speakers.

I will show you how to connect the RPi to your local wireless network using an inexpensive (less than $6) USB Wi-Fi adapter. However, you could run an Ethernet cable from your RPi to your router and skip the Wi-Fi stuff altogether.

The two USB ports on the RPi Model B fit our needs exactly with one port for the USB adapter and the other for a Flash drive containing music. The Model A RPi would not work for this application because it only has a single USB port (unless you want to incorporate a USB hub into your system). In any case, the increased amount of RAM (512 Mbytes) on the Model B probably makes the MPD server run better. The hardware connections to/from the RPi are as follows:

1. Connection of the power adapter to the micro USB port on the card.
2. Connection of the 3.5 mm audio output to the music system via the appropriate cable.
3. Insertion of the 4 GB or larger SD memory card containing the operating system.
4. Connection of the USB Wi-Fi adapter to one of the USB ports.
5. Temporary connection of a USB hub to the other USB port. This will only be needed during the setup process and will be removed when setup is completed. Connect a USB keyboard and mouse to the hub at this time. The USB hub will be replaced with a USB Flash drive containing the music library when setup is finished.
6. Connection to either an HDMI or composite video monitor. The RPi will run headless; that is, with no monitor required once the installation procedure is competed.

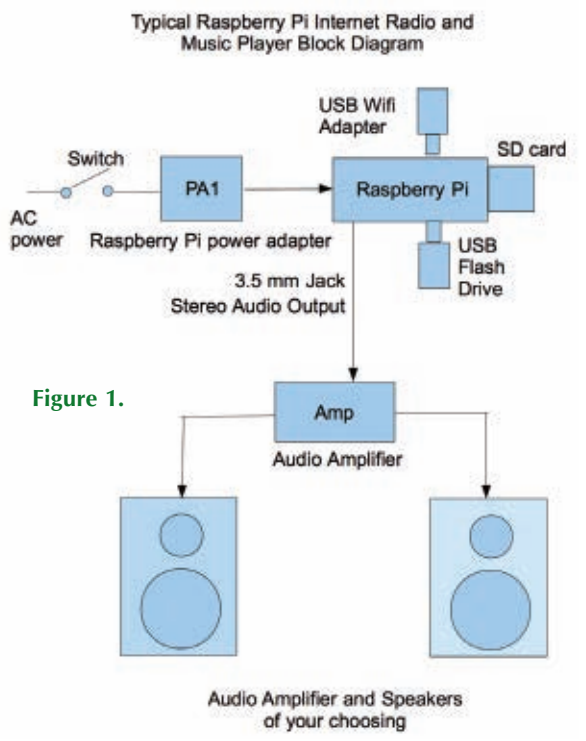


Figure 1.

TABLE 1. Required Hardware.

Item	Source
Raspberry Pi Model B with power supply	www.adafruit.com www.mcmelectronics.com www.sparkfun.com www.newark.com
SD Memory Card 4 GB or larger	Everywhere
Flash Drive Bigger = More Songs I used an 8 GB device.	Everywhere
Mini 150 Mbps USB Wi-Fi Wireless N LAN Network Adapter 802.11n/g/b	eBay
Audio Output Cable 3.5 mm stereo cable to whatever input (s) your amplifier requires	eBay, BestBuy, Amazon

Software

When I first began using the RPi for Internet radio, it had a real problem with loud pops when changing stations. So loud, in fact, as to almost make this arrangement unusable. Recently, however, I reloaded my RPi with the 02-09-2013 version of the Raspbian operating system (available from **raspberrypi.org**); this problem seems to have been fixed.

If you have an older version of the operating system software on your RPi, I strongly suggest reloading it with this new version. You can follow the directions in my March 2013 article, "Raspberry Pi, Anyone?" All of the initial configuration described in that article is still appropriate and should be done before proceeding.

Assuming we are starting with a newly loaded RPi image, we will first get the Wi-Fi working, then update the OS software with all the newest packages. Next, we will update the RPi firmware and install a series of packages required to run MPD. Finally, we will configure MPD for our purposes.

To get started, insert the SD memory card with the operating system into the RPi and make all of the connections listed in the hardware section above. Once power is applied, your RPi should boot and take you into *raspi-config* automatically if you haven't done your initial configuration already, or to the command prompt if you have. At this time, please log in with the username of *pi* and the password of *raspberry*.

1. At the command prompt, type *startx* to start up the Graphical User Interface (or GUI).
2. Click on the Wi-Fi Config icon to configure your RPi for wireless operation. If you are going to use a wired network connection, skip to step 6.
3. From the app's menu, click Network and then Add.
4. Input the SSID or name of your wireless network, and select the appropriate Authentication method for your network, if any. My wireless network uses WPA2 — Personal authentication, but yours may be different. Finally, type your network password into the PSK field.
5. Click the Add button and if all is well, your RPi should connect to your wireless network. If it doesn't, check to see that you have the correct SSID, Authentication, and PSK values set.
6. Click on the LXTerminal icon on the desktop to bring up a shell window.
7. Type *sudo apt-get update* to allow your RPi to determine all of its OS packages that are out of date.
8. Type *sudo apt-get upgrade* to install all of the new OS packages.
9. Once the upgrade is complete, type *sudo reboot* to restart your system. A reboot is necessary to utilize all of the new software you just installed.
10. Log back in and run *startx* again. This time, click the Midori icon to bring up the browser. Type *github.com/hexxeh/rpi-update* into the location bar.
11. Also click on the LXTerminal icon to bring up a shell window.
12. In the shell, type *sudo apt-get install git-core* to load some more software we will need.
13. We will now download and install the RPi firmware updater so that our system can be completely updated, both in terms of its firmware and its OS software. In the Midori window, scroll down to the Installing section and highlight the long command that begins with *sudo wget ...* Once highlighted, copy it to the clipboard and then paste it into the LXTerminal window to execute it.
14. To run the firmware updater, type *sudo rpi-update*.
15. After this process completes, reboot your system again and log back in.
16. Next, we will install software that will automount any USB Flash drives plugged into the RPi. This means that if you plug in a new Flash drive, the system will recognize it automatically without you having to do anything. Type *sudo apt-get install usbmount*. When completed, type *sudo service udev restart* to make the system begin using the *usbmount* software.
17. To test that automounting is working, list the content of the directory */media/usb0* using the command *ls /media/usb0*. The directory should be empty. Plug in a Flash drive with some files on it into the USB hub and run the command again. This time, the content of the Flash drive should be visible in the directory listing. If not, reboot and try again.
18. Finally, install MPD and MPC using the command *sudo apt-get install mpc mpd*. We install MPD because we need it; we install MPC to help in testing the MPD server.
19. Next, we need to edit the MPD configuration file that was just installed. We will invoke the vi program editor with the following command line: *sudo vi /etc/mpd.conf*. You can use another editor if you are more comfortable with it.
20. Find the music_directory line in the file and change its value in double quotes to "/media /usb0." This tells MPD where to look for music files. Next, find the bind_to_address line and replace "localhost" with "any." This allows us to interact with the MPD server remotely. Next, find the line #mixer_type "software" and remove the # sign. Finally, find the line #autoupdate "yes"; first remove the # sign, and then change "yes" to "no." Save the changes to the file with the command *:wq if using vi*.

21. Reboot once more so MPD will read its new configuration.
22. Log in and type *alsamixer* on the command line. This will bring up a GUI with a slider in the middle of the screen. Raise the value three quarters of the way up and then click the Esc key to terminate this program.
23. To make sure the kernel modules necessary to output sound from the RPi are loaded, type the following command in a shell window: *sudo modprobe snd_bcm2835*.
24. Lastly, type *amixer cset numid=3 1* to route sound to the 3.5 mm jack on the RPi.
25. Reboot one more time and look closely at the messages that stream by on the monitor. Make sure you don't see any errors when the MPD server automatically starts up. If you do, something is wrong and you will need to figure out what that is and fix it.

Now, it's time to find out if our Internet radio/music file player is working. For this test, make sure there are music files on the Flash drive which is plugged into the USB hub. Then, plug some headphones into the 3.5 mm jack on the RPi. Type the following command in a shell window: *mpc add http://64.40.99.2:8088*. Then, type *mpc play*. If all is well, you should hear the Beatles Internet radio station.

Next, type *mpc clear* to stop the server from playing the Internet radio station. Then, type *mpc update* to force the MPD server to build a database of the songs on the Flash drive. Type *mpc listall* and you should see all the songs listed that are on the Flash drive.

Then, type *mpc add* followed by one of the file names you saw with the *listall* command. Finally, type *mpc play* and you should hear the song you just added.

If these tests are successful, you are ready to go. You can power-down your RPi and remove the USB keyboard, mouse, and the USB hub. Remember to plug your USB Flash drive containing your songs back into the RPi and connect the audio cable to your stereo system.

To control your MPD server remotely, you can use my iPhone/iPod Touch app (MpdClient) or any other MPD client application you have available.

Songs On A USB Flash Drive

The MPD server used in conjunction with the RPi can play many different audio file formats. If you would like a complete list of supported files, type *mpd --version* at a command line prompt. This means you can put a variety of audio file types on your Flash drive, and MPD should have no trouble indexing and playing them all.

MPD does not require any kind of directory structure for music file storage like some other music library programs do (like iTunes). All files can be placed in the root directory of your Flash drive and MPD will sort things out. This is possible because MPD extracts song information from metadata stored in the music files themselves.

When MPD is commanded to do an update, it extracts information from the song files and stores it in its private database. This allows for some very powerful searching and sorting capabilities. Using the MPC client we installed on the RPi for testing, we can list all the available songs by typing *mpc listall*. Type *mpc list artist* to see a list of all the artists with songs on the Flash drive. Type *mpc list album* to see the albums the songs on the Flash drive are associated with. Finally, *type mpc list genre* to see the genres covered by the songs on the Flash drive.

While these capabilities might not seem all that powerful, consider the following example. Say you execute *mpc list genre* and get the following list retuned:

World
Rock
Unclassifiable
Holiday
Alternative & Punk

Say you only feel like listening to World music. You can do so using the following series of commands:

1. *mpc clear* — This clears the server's playlist of all songs which stops playback.
2. *mpc findadd genre World* — This creates a World music playlist on the server.
3. *mpc play* — This starts the playback of World music.

Of course, I only used genre here as an example, but you could use these same techniques to select songs from a specific album or to play back all songs by a specific artist. Pretty powerful stuff, really.

Conclusion

Having a remote-controllable Internet radio/music file player is very convenient. Building one yourself inexpensively makes it all that much sweeter. **NV**

Raspberry Pi Specs.

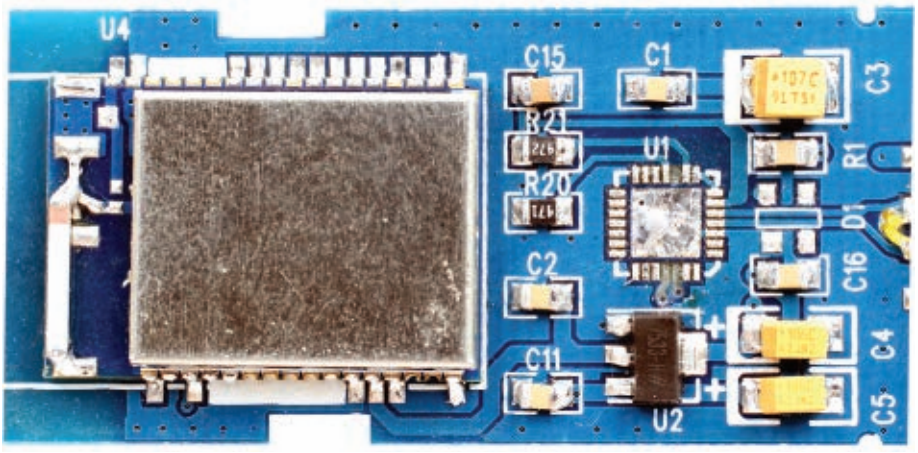
Feature	Model A	Model B
Suggested Retail Price	$25.00	$35.00
CPU	700 MHz ARM11 with hardware floating point	
GPU	Capable of 1080p video	
Memory	256 MB shared with GPU	512 MB shared with GPU
USB Ports	1	2
Video Outputs	HDMI and composite video	
Audio Outputs	Stereo 3.5 mm jack + HDMI	
Onboard Storage	SD memory card up to 128 GB	
Networking	None	10/100 Ethernet
Connectivity	8 x GPIO, UART, I²C, SPI	
Size	3.3" x 2.1"	
Power Requirement	300 mA – 1.5W	700 mA – 3.5W
Official OS	Raspbian version of Linux	

BY FRED EADY

Score Big With The Lemos LMZ ZigBee Module

ZigBee is still an exclusive club. However, nonmembers like us can still play a round of 18 on the pristine ZigBee greens. This month, we have a rare opportunity. We are going to scratch-write a 32-bit hardware driver library to support a brand new member of the ZigBee family. You will be able to start your ZigBee application development immediately as we will not employ the services of any specialized development hardware or software. We're going to build our ZigBee driver using easily obtained off-the-shelf hardware and development tools.

Go to **www.nutsvolts.com/index.php?/magazine/article/august2013_DesignCycle** for any additional files and/or downloads associated with this article. You can also discuss this topic at **http://forum.nutsvolts.com**.



■ PHOTO 1. The Lemos LMZ ZigBee module you see here is the USB variant. All of the native USB support is located to the far right of the module. The actual ZigBee radio is contained within the RF shield at the far left. The absence of U1 and D1 will become evident as we make our way through the project.

Driver Requirements

The functionality of the ZigBee driver is directly related to the characteristics of the ZigBee radio hardware. Our driver will address the radio portion of the Lemos LMZ ZigBee module smiling under the lights in **Photo 1**. The Lemos module is available as a USB dongle or a stand-alone embeddable ZigBee radio module.

The USB dongle variant of the Lemos module (LMZ-USB) communicates to a host computing device using the RS-232 protocol at LVTTL (Low Voltage Transistor-Transistor Logic) logic levels via a Silicon Labs CP2102 (U1). The CP2102 is Silicon Labs version of the FTDI FT232RL USB-to-UART bridge IC. Our embedded LMZ ZigBee module driver will target a microcontroller-based host that is equipped with an on-chip UART engine. Thus, the PC-oriented serial port USB conversion and virtual COM port functionally provided by the CP2102 and its PC driver are not required. We only need to directly connect the LMZ radio module's UART interface to the host microcontroller's UART I/O pins.

A typical CP2102 implementation is shown in **Schematic 1** which I built using an ohmmeter and printed circuit board (PCB) observation. If you take a peek at the CP2102 datasheet, you will see that the Lemos LMX-USB's CP2102 is implemented in its most basic manner.

That's a good thing as now we know that we can simply remove the CP2102 and protection diodes (D1) from the LMZ-USB and expose the Lemos module's UART interface.

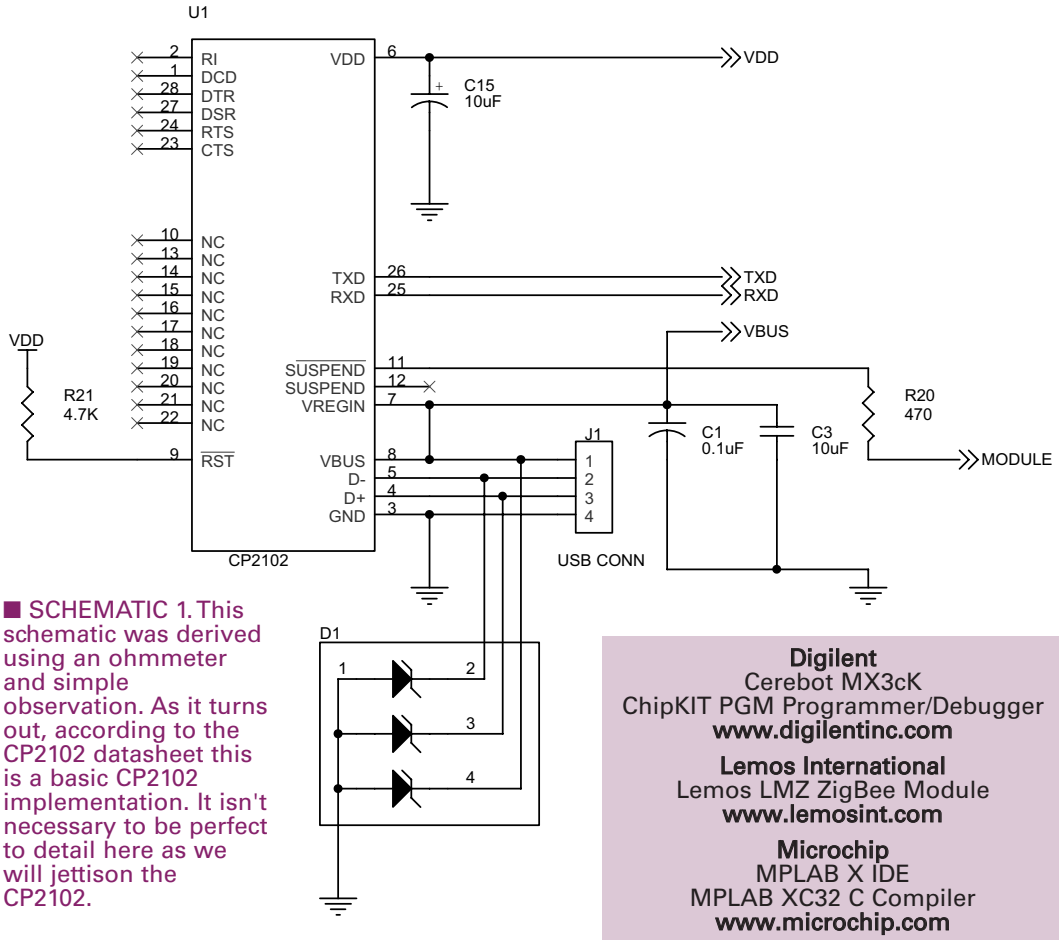
Take another look at **Photo 1**. The CP2102 (U1) and the protection diode array (D1) have been removed. We can now tap into the LMZ-USB's center-most USB connector pins and use them as RS-232 protocol I/O pins. With the CP2102's absence, we can directly connect the hijacked USB I/O pins to the LMZ module's UART interface.

Everything we left on the LMZ-USB PCB is related to converting the +5 volt VBUS signal to a +3.3 volt

power rail for the Zigbee module. U2 is a 3.3 volt linear regulator that is supported by filter capacitors C2, C4, C5, and C11. Resistors R20 and R12 are used by the CP2102 for SUSPEND and RESET functions, respectively.

Using **Figure 1** as a reference, **Photo 1** identifies the CP2102's D+/D- USB inputs and its TXD/RXD UART interface. With no CP2102 in the picture, we can bypass the originally intended USB functionality by simply connecting the CP2102's D+/D- pins to its TXD/RXD pins. This allows us to use the LMZ-USB's USB connector as a serial I/O connector instead.

Another look at the far right of **Photo 1** reveals a pair of insulated wires falling into a hole I drilled. The other ends of the pair of wires connected to the center-most pins of the LMZ-USB's USB connector terminate at the CP2102's TXD and RXD pins. Now, we have a clear path to the LMZ module's UART interface. However, the LMZ-USB's Type A USB connector was not designed to interface RS-232 signals. So, we've got an adapter to build.



■ SCHEMATIC 1. This schematic was derived using an ohmmeter and simple observation. As it turns out, according to the CP2102 datasheet this is a basic CP2102 implementation. It isn't necessary to be perfect to detail here as we will jettison the CP2102.

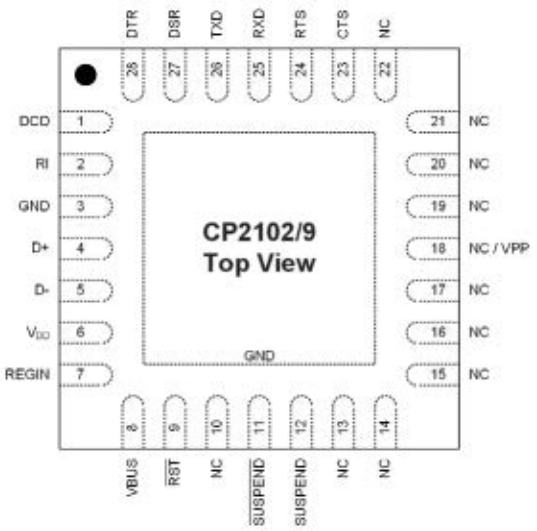
Digilent
Cerebot MX3cK
ChipKIT PGM Programmer/Debugger
www.digilentinc.com

Lemos International
Lemos LMZ ZigBee Module
www.lemosint.com

Microchip
MPLAB X IDE
MPLAB XC32 C Compiler
www.microchip.com

USB Type A To RS-232

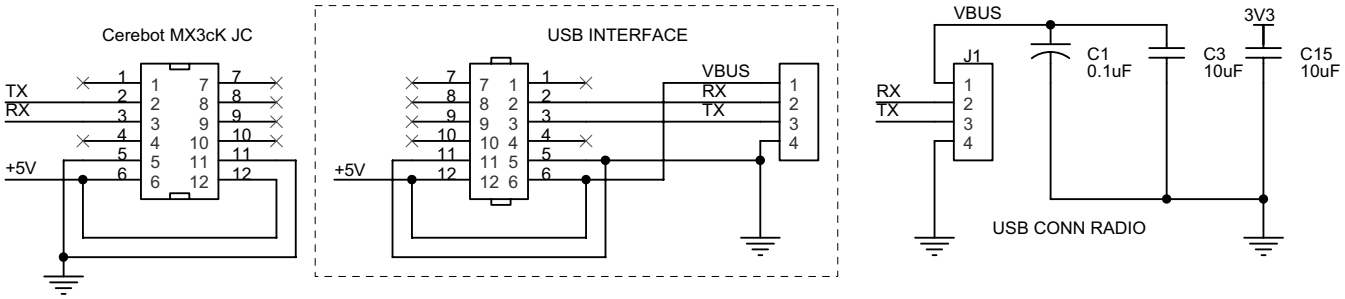
Naturally, the adapter design depends on the existing interfaces. We know that we have a Type A USB interface on the LMZ-USB dongle. Even though our Lemos module is really an LMZ-USB, we removed the original USB component. So, from now on our LMZ-USB dongle will be seen as a simple carrier board for the LMZ module. The interface on the other end has yet to be identified. A perfect host for our Zigbee module would provide VBUS voltage and a standard UART interface. In that we've removed the USB requirement, the host microcontroller need not support host USB mode. Should we want to develop a sensor or control/monitor host platform, our selection of a host microcontroller and development environment should support these possibilities.



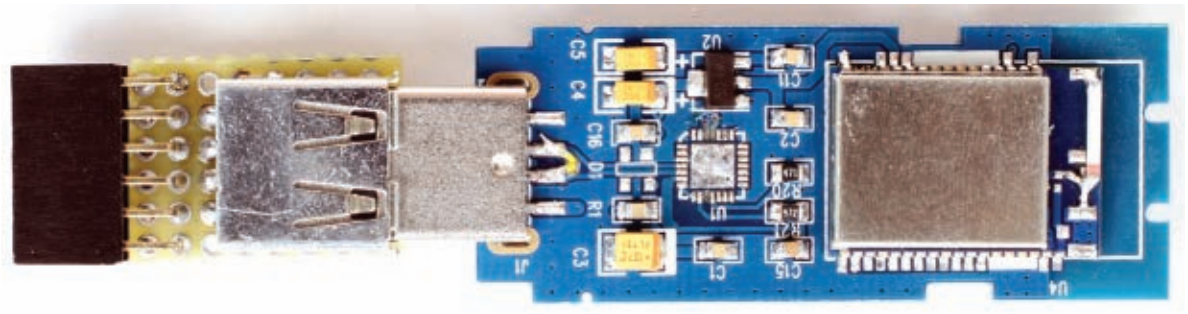
■ FIGURE 1. Beating this figure against Photo 1 uncovers the missing CP2102 TXD and RXD solder pads. The D+/D- to TXD/RXD physical connections take place on the opposite side of the printed circuit board.

Flight-tested sensors and I/O hardware can be instantly realized with Digilent's extensive line of Pmods. So, let's tap one of the latest Cerebot boards as the host for our ZigBee module. We will employ the services of the Digilent Cerebot MX3cK.

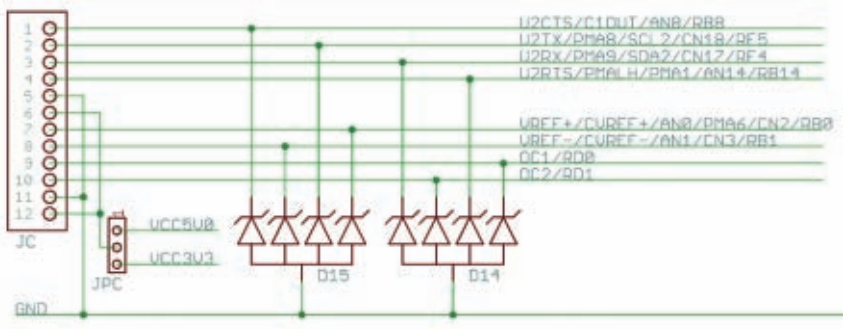
The MX3cK is specifically designed to interface to Digilent's family of Pmods. That means our host microcontroller interface will consist of 14-pin dual-row female headers. The MX3cK's UART1 portal can be accessed natively



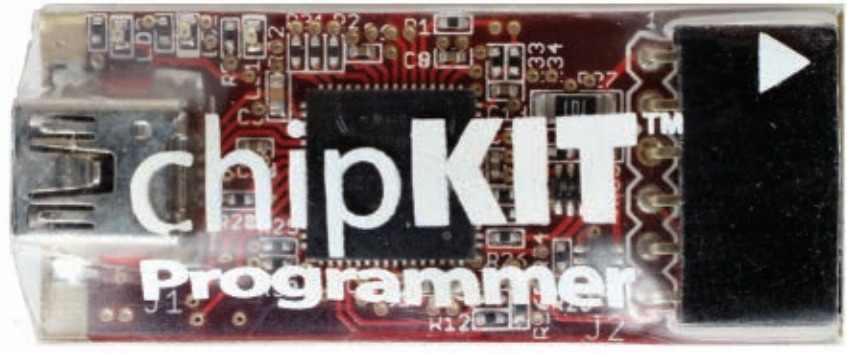
■ SCHEMATIC 2. With the CP2102 and protection diodes out of the way, we have a clear shot to the Cerebot MX3cK's JC connector. The MX3cK's JC connector houses all of the PIC32MX320F128H's UART2 signals.



■ PHOTO 2. Being a product of the 1960's space program, this shot kinda reminds me of the Apollo command module attached to the LEM (Lunar Excursion Module).



■ SCHEMATIC 3. The UART signals mate perfectly with the FTDI-based USB-to-UART Pmod. RTS and CTS signals are also present on interface JC. However, the LMZ ZigBee module only requires U2TX and U2RX.



■ PHOTO 3. The chipKIT programmer/debugger is designed to work with the chipKIT Max32, Uno32, uC32, and MX3cK. I've also used it to develop home-brewed PIC designs.

via an onboard FTDI FT232RQ. So, we'll wire our ZigBee module interface on the MX3cK side to match the USB-to-UART Pmod. All we need to know schematically about the interface is laid out in **Schematic 2**. Removing the CP2102 and protection diode array logically leaves us with J1, C1, and C3, presenting the MX3cK's VBUS power rail to the Lemos module's +3.3 volt regulator. Note that we cross the RS-232 protocol TX and RX signals in our hand-wired USB interface module.

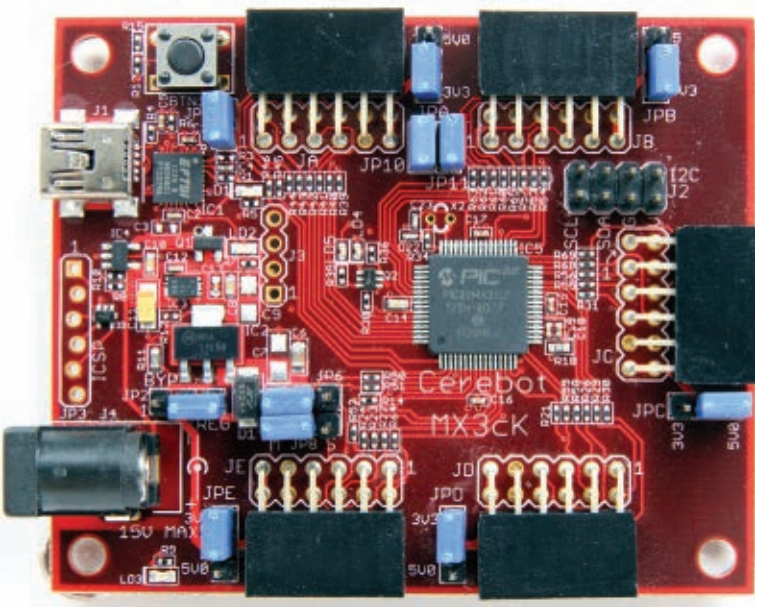
The USB-to-Cerebot MX3cK interface is attached to the Lemos module in **Photo 2**. What you don't see in this shot is the hand-wiring on the opposite side of the adapter. In that the MX3cK's interface connectors are all female, we'll use one of the Digilent 14-pin extended male headers to make electrical contact between the adapter's interface and the MX3cK JC interface. This interface is depicted graphically in **Schematic 3**.

The Baby Of The Bunch

The MX3cK is the smallest and least adorned Cerebot. However, that doesn't mean it can't hold its own. The MX3cK is based on the PIC32MX320F128H, which does not contain an on-chip USB host engine. The idea behind the design of the MX3cK is to add peripherals as needed with Pmod modules. The standard SPI, I²C, UART, and timer

peripherals are all there on-chip.

Arduino compatibility using a bootloader is also part of the MX3cK design philosophy. However, I overwrote the factory-installed bootloader by using a chipKIT PGM programmer/debugger to load and test our Lemos ZigBee module driver firmware. The chipKIT PGM programmer/debugger shown in **Photo 3** works seamlessly with MPLAB X and the X series of Microchip C compilers. An aerial view of the MX3cK is realized in **Photo 4**. It can be powered by an external power supply or via the FTDI-supported USB portal. We are particularly interested in interface connector JC, as is holds our Lemos module UART signals. The +5.0 volt VBUS power supplied to the module flows through the JPC jumper. The complete module driver development setup is being flown over in **Photo 5**.



■ PHOTO 4. This shot looks very busy. However, the MX3cK is a very basic implementation of the PIC32MX320F128H. Power for the ZigBee module is obtained from the JC connector by way of the JPC jumper.

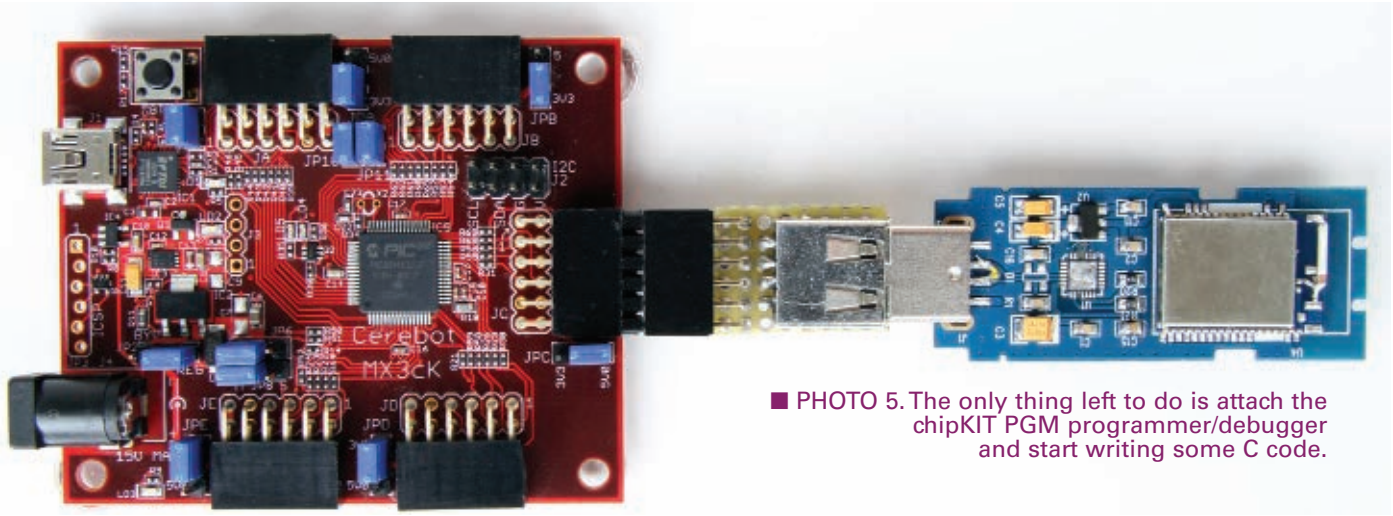
Writing The Lemos LMZ ZigBee Module Driver

The LMZ ZigBee module is no lightweight, either. The design goal for this module was ease of use. With that, little knowledge of the complexities of ZigBee networking is required to put it to work. Our ZigBee module is loaded with a copy of the EmberZnet ZigBee protocol stack. All the ZigBee developer has to do is issue some very simple commands to the module's UART via a standard three-wire serial connection (TX, RX, GND). Our driver development work will revolve around writing command and response code. For instance, to check the status of the module, we would issue an INFO command. The response to the INFO command provides us with the following node-related information:

- Firmware Version
- EUI (Extended Unique Identifier)
- Type of Node (Coordinator-Router-End Device)
- Sleep Status
- Network Status

Rather than show you the captured packet data with the MPLAB X Watch window, let's plant some embedded debug code into our LMZ module driver. The first order of business is to redirect all *stdio* (standard I/O) print requests (*printf*) to UART1. This is accomplished by overriding any calls to *putc*. The following code simply loads the UART1 transmit register with anything that was destined for UART2 via the standard I/O *putc* function:

```
//************************************************
//* UART1 REDIRECTOR
//************************************************
```



■ PHOTO 5. The only thing left to do is attach the chipKIT PGM programmer/debugger and start writing some C code.

```
void _mon_putc(char c)
{
       U1TXREG = c;
       while(U1STAbits.TRMT == 0);
}
```

We must also disable buffered outputs to *stdio*. This bit of code must reside in our *init* function and here's how that's done:

```
setbuf(stdout,NULL); //UART1 REDIRECT
```

The UART1 redirection code allows us to use *printf* statements to "print" debug information to a standard terminal program running on the host PC. The MX3cK's UART1 engine is supported by an FTDI FT232RQ USB-to-UART bridge IC. So, all we need is Tera Term Pro and a USB cable to capture and view our user-generated debug messages. All of the commands in our module driver will build command packets in the *txBuf* array. Let's tear down the *sendInfo* command packet:

```
//***********************************************
//* SEND INFO PACKET
//***********************************************
void sendInfo(void)
{
  BYTE i;

  txBuf[0] = 0x6A;          //preamble
  txBuf[1] = 0x95;          //preamble
  txBuf[2] = xmitSeqNum;    //sequence number
  txBuf[3] = cmdInfo;       //command
                            //(cmdInfo=0x00)
  txBuf[4] = 0x01;          //data length
  txBuf[5] = 0x00;          //data
  txBuf[6] = 0x00;          //chksum

  txMsgLen = 0x07;          //total msg len
  //compute chksum
  for(i=0;i<txMsgLen-1;++i)
  {
    txBuf[6] += txBuf[i];
  }
  xmitPkt(txMsgLen);        //send it
}
```

Every command packet sent to the module begins with a two-byte preamble (0x6A, 0x95). The preamble is followed by a user-generated sequence number. The sequence number helps us keep up with which response belongs to which command. The command code lies between the sequence number and data length bytes. The data field contents depend on the command. In any event, the length of the data field is limited to 80 bytes. All packets end with a CRC byte which is computed by adding all of the bytes in the packet, beginning with the first preamble byte and ending at the last data byte.

Building a command packet is academic if you don't apply it. So, we need to write a packet transmit routine. Debug messages will flow through UART1. UART2 handles communications with the ZigBee module. In that we've redirected the *stdio* traffic to UART1, we must manually handle data radio traffic moving through UART2:

```
//***********************************************
//* TRANSMIT PACKET -> UART2
//***********************************************
void xmitPkt(BYTE msglen)
{
    BYTE i;
       for(i=0;i<msglen;++i)
       {
              U2TXREG = txBuf[i];
              while(U2STAbits.TRMT == 0);
       }
}
```

The module will always respond to a command with an ACK packet, followed by the command response packet. You know what that means. We need to cobble up an ACK packet handler routine:

```
typedef struct
{
       BYTE preamble[2];
       BYTE seqNum;
       BYTE cmdID;
       BYTE dataLen;
       BYTE recvCmd;
       BYTE status;
       BYTE crc;
}ACKPACKET;
ACKPACKET ackPkt;
//***********************************************
//* ACK PACKET HANDLER
//***********************************************
BYTE ackHandler(void)
{
    BYTE rc;

    ackPkt.status = 0x01;

    ackPkt.preamble[0] = recvchar();
    if(ackPkt.preamble[0] == 0x6A)
    {
       ackPkt.preamble[1] = recvchar();
       if(ackPkt.preamble[1] == 0x95)
       {
              ackPkt.seqNum = recvchar();
              ackPkt.cmdID = recvchar();
              if(ackPkt.cmdID == cmdACK)
              {
                  ackPkt.dataLen = recvchar();
                  ackPkt.recvCmd = recvchar();
                  ackPkt.status = recvchar();
                  ackPkt.crc = recvchar();
              }
       }
    }

    return(ackPkt.status);
}
```

The *recvchar* function — which plucks received characters from the receive buffer (RxBuf) — is part of the UART2 interrupt handler engine. Thus, our ACK packet handler simply reads incoming ACK packet bytes and stuffs them into the *ackPkt* structure. Once we get an OK from the ACK packet, we can process the response. In this case, that would be the INFO command response:

```
typedef struct
{
    BYTE preamble[2];
    BYTE seqNum;
    BYTE cmdID;
```

```
    BYTE dataLen;
    BYTE version[2];
    BYTE EUI[8];
    BYTE type;
    BYTE heartBeat[2];
    BYTE networkFlag;
    BYTE crc;
}INFORSP;
INFORSP infoRsp;
//***********************************************
//* INFO RESPONSE HANDLER
//***********************************************
void infoRspHandler(void)
{
   BYTE i;

   infoRsp.preamble[0] = recvchar();
   if(infoRsp.preamble[0] == 0x6A)
   {
       infoRsp.preamble[1] = recvchar();
       if(infoRsp.preamble[1] == 0x95)
       {
           infoRsp.seqNum = recvchar();
           infoRsp.cmdID = recvchar();
           if(infoRsp.cmdID == cmdInfoRsp)
           {
               infoRsp.dataLen = recvchar();
               infoRsp.version[0] = recvchar();
               infoRsp.version[1] = recvchar();
               for(i=0;i<8;++i)
               {
                     infoRsp.EUI[i] = recvchar();
               }
               infoRsp.type = recvchar();
               infoRsp.heartBeat[0] = recvchar();
               infoRsp.heartBeat[1] = recvchar();
               infoRsp.networkFlag = recvchar();
               infoRsp.crc = recvchar();
           }
       }
   }
}
```

As you can see, the INFO command response handler looks a lot like the ACK handler. RxBuf is a circular buffer that uses head and tail index pointers. At this point, we are again servicing RxBuf, which is interrupt driven at its input and poll driven at its output. That is, an interrupt is fired when UART2 receives a character. The incoming character is placed at the next available slot of the RxBuf pointed to by the head pointer. The application program checks to see if there is a character to be retrieved from RxBuf (head pointer not equal to tail pointer). If a character is available, the application uses the *recvchar* function tail pointer to extract the character from the RxBuf array. Once the ACK packet and INFO command response are placed in their respective structures, we can use the MX3cK's UART1 and Tera Term Pro to display the structure content. Here are the debug display routines for the ACK packet and INFO command response:

```
void showAckPkt(void)
{
  printf("\r\n");
  printf("Preamble    %02X %02X\r\n",ackPkt.
                      preamble[0],ackPkt.
                      preamble[1]);
    printf("Sequence Number     %02X\r\n",ackPkt.
                                seqNum);
    printf("Command ID          %02X\r\n",ackPkt.
                                cmdID);
    printf("Data Length         %02X\r\n",ackPkt
                                .dataLen);
    printf("Command Recvd       %02X\r\n",
                                ackPkt.recvCmd);
    printf("Status              %02X\r\n",
                                ackPkt.status);
    printf("CRC                 %02X\r\n",
                                ackPkt.crc);
}
void showInfoRsp(void)
{
   BYTE i;
   printf("\r\n");
   printf("Preamble      %02X %02X\r\n",infoRsp.
                         preamble[0],infoRsp.
                         preamble[1]);
   printf("Sequence Number     %02X\r\n",
                          infoRsp.seqNum);
   printf("Command ID    %02X\r\n",
                          infoRsp.cmdID);
   printf("Data Length   %02X\r\n",
                          infoRsp.dataLen);
   printf("Version       %02X %02X\r\n",
                          infoRsp.version[1],
                          infoRsp.version[0]);
   printf("EUI          ");
   for(i=0;i<8;++i)
   {
   printf("%02X ",infoRsp.EUI[i]);
   }
   printf("\r\n");
   printf("Type          %02X\r\n",
                         infoRsp.type);
   printf("HeartBeat    %02X%02X\r\n",
                         infoRsp.heartBeat[1],
                         infoRsp.heartBeat[0]);
   printf("Network Flag     %02X\r\n",infoRsp
                            .networkFlag);
   printf("CRC          %02X\r\n",infoRsp.crc);
}
```

We're ready to test our ACK and INFO functions. We'll use a state machine to:

- Invoke the *sendInfo* command.
- Wait for the ZigBee module to respond.
- Retrieve the ACK packet and determine the status of the INFO command.
- If the command status is OK, retrieve the INFO command response and display it.

Here's the state machine code:

```
//***********************************************
// MAIN
//***********************************************
int main (void)
{
   BYTE i,rc;
   init();

     msgTimer = 0xFFFF;
     do{
     switch(appState)
     {
          case INFO:
             sendInfo();
             nextState = GETACK;
             appState = WAIT;
          break;

          case GETACK:
              if(CharInQueue())
              {
```

```
 if(rc=ackHandler() == 0x00)
                          {
                        showAckPkt();
                             appState = GETRSP;
                          }
                          else
                          {
                             showAckPkt();
                             appState = IDLE;
                          }
               }
          break;

          case GETRSP:
               if(CharInQueue())
               {
                  infoRspHandler();
                  showInfoRsp();
                  appState = IDLE;
          }
          break;

          case WAIT:
               if(—msgTimer == 0)
               {
                    msgTimer = 0xFFFF;
                    appState = nextState;
               }
          break;

          case IDLE:
               ++i;
               break;
    }
    }while(1);
}
```

If everything goes as planned, the ACK packet and INFO response packet will be displayed in the Tera Term Pro terminal emulator window on my laptop. A successful run will also force the state machine into IDLE mode. Here's a look at the ACK packet we captured:

```
UART1 UP
UART2 UP

Preamble         6A 95
Sequence Number  00
Command ID       FF
Data Length      02
Command Recvd    00
Status           00
CRC              00
```

I placed a couple of *printf* statements in the initialization function to tell us when the UART configurations were complete. The list of data following the UART messages is the incoming ACK packet. The Lemos module preamble bytes are always 0x6A and 0x95. In that the INFO command was the first command, the sequence number reflects the sequence number initialization performed in the *init* function.

There are two bytes of data in an ACK packet. The first byte of the payload data is the command that was received. We sent an INFO command (0x00) and that is reflected in the Command Recvd field of the structure. My trusty HP16C programmer's calculator tallies 0x0200 for the seven ACK packet bytes. The most significant byte of the computed CRC is discarded, leaving a posted CRC of 0x00.

An ACK status of 0x00 tells us that everything is cool with the INFO command and to expect an INFO response:

```
Preamble         6A 95
Sequence Number  00
Command ID       80
Data Length      0E
Version          01 08
EUI              84 9D 83 01 00 6F 0D 00
Type             01
HeartBeat        0000
Network Flag     00
CRC              B8
```

Our INFO response contains the expected preamble bytes followed by the sequence number of the INFO command we issued. Command response IDs are always identified as the original command with the most significant bit set. Thus, INFO command ID = 0x00 and INFO response ID = 0x80. If you care to count the bytes between the Data Length byte and the CRC byte, you will come up with 14 (0x0E). The tally includes:

- Two bytes of firmware version data.
- Eight bytes of EUI.
- One Type byte (our module is a router).
- Two bytes of sleep data (our router does not sleep).
- One byte of Network status (no network attachment).

The INFO response packet bytes total up to 0x03B8. Dumping the most significant byte leaves us with a CRC value of 0xB8.

We Can't Drive Off

Not just yet, anyway. We have a few more commands to code up:

```
//*********************************************
//* Lemos LMZ ZigBee Module COMMANDS
//*********************************************
#define cmdInfo                     0x00
#define cmdInfoRsp                  0x80
#define cmdSetBaud                  0x02
#define cmdSetBaudRsp               0x82
#define cmdSetPower                 0x03
#define cmdSetPowerRsp              0x83
#define cmdBuildNetwork             0x06
#define cmdBuildNetworkRsp          0x86
#define cmdPermitJoining            0x07
#define cmdPermitJoiningRsp         0x87
#define cmdAutoJoinNetwork          0x0A
#define cmdAutoJoinNetworkRsp       0x8A
#define cmdLeaveNetwork             0x0B
#define cmdLeaveNetworkRsp          0x8B
#define cmdSendData                 0x0F
#define cmdSendDataRsp              0x8F
#define cmdReceiveData              0x10
#define cmdACK                      0xFF
```

When we meet again, we will take a look at the code that supports the UARTs. It would also be a good idea to write an application to build a ZigBee network and transfer data between its nodes. **NV**

Connected Cars Communicate

Telematics is hot. *Do you have it yet?*

Telematics is that wireless technology that connects your car to the outside world. The idea has been around for a long time, but its incorporation into new vehicles is exploding. Most new cars — especially the more expensive ones, but some lower cost models too — have some type of telematics system. Here is a summary of this emerging technology.

ONSTAR IS THE STAR

General Motor's OnStar system is a good example of telematics, and may be the oldest case of it. Beginning in 1996, GM started offering a unit that had an embedded and dedicated cell phone as the communications link combined with sensors in the car. An accompanying service anchored the system. OnStar was designed to provide automatic accident notification to the OnStar office so they could send help. It was also used to provide roadside service if needed. Over the years, OnStar has infiltrated virtually all GM cars today. It offers expanded services such as remote door unlocking, mapping services, and stolen car locating. Over six million GM cars now have it, and more systems are being sold as standard every day. That's not all, however. As of the current model year, most other car manufacturers now offer their own version of OnStar – with and without the monthly billed service.

Telematics is usually tied into the vehicle's infotainment system. This is the dashboard combination of information-producing equipment along with entertainment features. Information systems like GPS navigation with its big screen maps and backup video are common on more and more cars. The entertainment systems today are the usual AM/FM/HD/Satellite radios, a CD player, and a port for MP3 players and iPods. Some systems also include a voice response enabled Bluetooth hands-free calling feature that lets you answer and make calls on your cell phone. More is yet to come.

HOW IT WORKS

As each car gets more electronics, the internal systems get continuously more complex. Each subsystem usually has its own embedded microcontroller, meaning that most cars have dozens of the little computers buried all around the vehicle. With more complex systems, the trend is to minimize the number of individual processors and adopt a larger, faster, multicore processor to run everything. That, of course, naturally leads to the need for an operating system (OS) to organize and manage everything. Examples of real time OSs in cars today are BlackBerry's QNX and Microsoft's Windows Embedded. A Linux-based system called GENIVI is also in contention.

There are two basic forms of telematics being built into vehicles: one that uses a dedicated embedded cell phone with its own number and account; and the non-embedded type that uses your own smartphone. OnStar, for example, uses the built-in cell phone; Ford's Sync system uses the driver's smartphone. The cell phone provides the primary outside link to the Internet. Once you are connected to the Internet, all things are possible in the car. What a concept.

Most telematics systems today are either embedded or not. A forthcoming alternative is aftermarket systems you can add to any vehicle. Aftermarket systems will most likely use your own phone for connectivity.

That connection to the Internet gives you email, browsing, shopping, social media, music via the likes of Pandora Internet radio, video from YouTube, Hulu, and Netflix, as well as other multiple sources. In addition, most smartphones can also serve as hot spots using Wi-Fi. This allows

Most drivers will appreciate and use the new telematics systems to the max. They will wonder how they ever got along without them.

passengers to connect to external sources with their smartphones, tablets, or laptops.

The big problem is what do you do with all that information? As the driver, your job is to drive safely — not consume massive amounts of data and multimedia. That's a problem.

UNIQUE FORMS OF TELEMATICS

Trucks have already been using a form of telematics for years. Long haul tractor trailers are installed with asset tracking radios that keep the home office informed of location, speed average, gas mileage, and other monitored conditions. Some use Wi-Fi, while others use cellular connections. Even satellite systems are used. This is a form of machine-to-machine (M2M) communications that is expanding widely to monitor and control almost any asset, truck, or cargo.

One interesting new use of telematics is a system conceived by insurance companies to monitor your driving. Companies like Allstate, Liberty Mutual, and Progressive have a monitoring device that plugs into the onboard diagnostic connector (OBD-II) under the dashboard near

■ **FIGURE 1.**
This is the European Telecommunications Standards Institute (ETSI) vision of an Intelligent Transportation System (ITS). Its main focus is cars and trucks, but all modes of transportation are to be factored in eventually. Wireless systems are at the core of all ITS. The proposed US system is similar.

the steering wheel. That connector is used mainly by service dealers for diagnosing engine and other problems. However, the data provided is useful for determining things like engine speed, gas consumption, and other driving conditions. The idea is to reward good drivers with lower rates. The downside is that poor drivers can get dinged with higher rates. For now, the devices are used voluntarily. The collected data is downloaded to a computer and sent to the insurance company. A commercial version of this device lets you download and access the data via Bluetooth to your smartphone using an app.

INTELLIGENT TELECOMMUNICATIONS SYSTEMS

With the world at your fingertips using the Internet and the cellular system, you have more than enough incoming information to overload you. Yet more is on the way. The government is now working on the Intelligent Telecommunications System (ITS). This is the US Department of Transportation's concept for a communications system that will connect cars to the road and other information sources, and provide car-to-car wireless links. Current research is ongoing and field trials of various types of vehicle connections are being conducted.

Even the Europeans are developing similar systems. **Figure 1** shows the concept. It involves not only cars, but also trucks, trains, boats, and aircraft. Other ground-based systems like broadcast stations, information signs, and toll booths are also involved. The overall goal of ITS is to provide safety information and to keep the traffic flowing.

The main features of the ITS are the vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) systems. Cars, trucks, and other vehicles will be equipped with dedicated short-range communications (DSRC) radios that can correspond with other vehicles, as well as road-side information sources. These radios use a form of Wi-Fi similar to the IEEE's 802.11a standard. It is called 802.11p and operates in the 5.9 GHz unlicensed band. It uses a protocol called WAVE that lets vehicles rapidly form ad hoc wireless connections on-the-fly to send or receive information of all types.

Some of the possibilities are cars trading traffic information or cars receiving weather and road condition data from road-side units (RSU). Digital signs for accident notification and other conditions are a part of the system as are toll collection stations that use RFID for automatic billing. V2V and V2I are not present yet, but are a forthcoming addition to the growing telematics revolution.

With the world at your fingertips using the Internet and the cellular system, you have more than enough incoming information to overload you. *Yet more is on the way.*

DRIVEN TO DISTRACTION

Most drivers will appreciate and use the new telematics systems to the max. They will wonder how they ever got along without them. The upshot of all this, unfortunately, is a massive driver distraction problem. Already, talking on the cell phone is a huge issue and some states have outlawed it. One recent study indicated that the main cause of teen vehicle deaths was not reckless driving or alcohol, but texting while driving. This is a major problem, and it doesn't help things that teens think they can do it safely. Look for more rules and regulations forbidding this action.

Drivers are already distracted while driving by using the radio or CD player, eating, smoking, disciplining kids, and other things. Add cell phones to that and you have a recipe for disaster. Telematics will make the problem worse unless some solutions are found. Hands-free voice response systems like Apple's Siri might be a possibility, but some studies show that it is no safer. However, voice response will no doubt be standard on many future telematics systems.

Improved displays and controls with a simple human machine interface (HMI) are also part of the solution. Research for ITS includes studies on how to package and present V2V and V2I messages to the driver without overloading their already maxed out visual, audible, and mental systems. The National Highway Traffic Safety Administration (NHTSA) recently issued guidelines to auto makers for minimizing distractions as a step in the right direction.

If you think fighter jet pilots are overwhelmed during battle, wait until you encounter your new vehicle with its tricked-out telematics systems. No more boredom during long commutes or extended highway trips. Just don't expect to see dashboard TV. **NV**

One interesting new use of telematics is a system conceived by insurance companies to monitor your driving.

ELECTRONET

CanaKit™

Electronic Kits • Modules • Parts

Save 10% using Coupon Code NV2012* when ordering online at www.canakit.com

Motor Controllers • Arduino • Signal Generators • FM Transmitters • Timers • Audio Amplifiers

200 KHz Function Generator

$49.95
ASSEMBLED

UK307

4-Port USB Relay Controller with 6-Channel Analog/Digital Interface

$59.95
ASSEMBLED

UK1104

5A Motor Speed Controller (PWM)

$22.95
ASSEMBLED

UK1102

Stereo Audio Amplifier (10W)

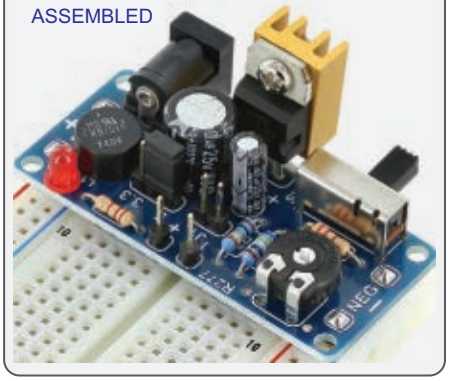
$35.95
ASSEMBLED

UK560

Plug-In Breadboard Power Supply

$19.95
ASSEMBLED

UK277



7A Bi-Directional Motor Speed Controller (PWM)

$39.95
ASSEMBLED

UK1125

USB PIC Programmer (MPLAB Compatible)

$69.95
ASSEMBLED

UK1300

Digital LED Timer (Up/Down)

UK258

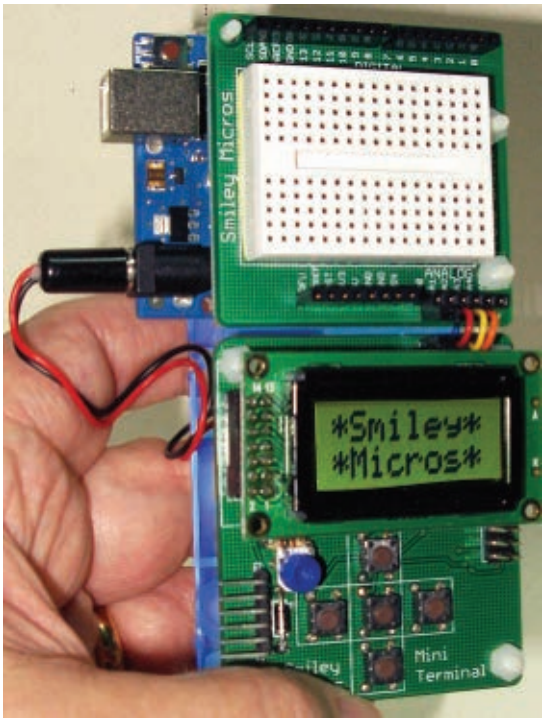
$46.95
ASSEMBLED

* Limit one per customer. Can not be combined with other coupons or volume discounts. Does not apply to products already on sale. Prices subject to change without notice.

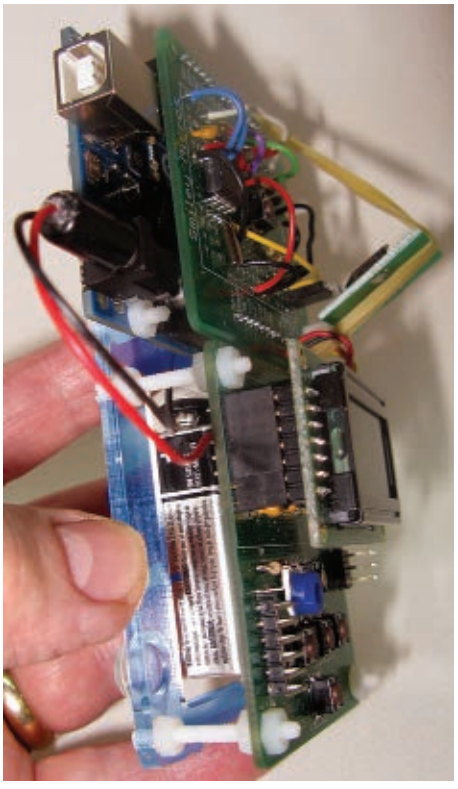
■ BY JOE PARDUE

Arduino Handheld Prototyper Part 2

Last month, we started a three part series on an Arduino-based handheld prototyper shown in Figures 1 and 2 (with the base artificially tinted blue so that it can be seen better). The Arduino handheld prototyper— as the name implies — lets us develop prototypes that are portable (in our very own hand), and provides a bonus that is lacking in the stand-alone Arduino: It can accept user keypad input via pushbuttons, and provides the user with visual output via an LCD. This device consists of three main parts: an Arduino proto shield, an I²C mini terminal, and a base kit that holds them together. We introduced the I²C mini terminal last month. Now, let's look in more detail at the software for that device.



■ FIGURE 1. Arduino handheld prototyper front view.



■ FIGURE 2. Side view.

Using The I²C Mini Terminal Software

The IMT (I²C mini terminal) software provides some simple and easy to use Arduino library functions that we saw last month in our *mini_terminal_master* demonstration program. We learned that by simply putting the MiniTerminal directory in the Arduino installation Libraries directory, we could use the Arduino IDE menu item 'Sketch,' followed by 'Import Library,' and then click the MiniTerminal entry that puts the header in our edit window:

```
#include <MiniTerminal.h>
```

We also add:

```
#include <Wire.h>
```

Go to **www.nutsvolts.com/index.php?/magazine/article/august2013_SmileysWorkshop** for any additional files and/or downloads associated with this article. You can also discuss this topic at **http://forum.nutsvolts.com**.

We are ready to start using the MiniTerminal functions from our Arduino. Doing this gives us the following functions:

```
// Print string to LCD at position p line l
void mt_print(char* x, byte p, byte l);

// Print string from Program Memory to LCD at
position p line l
void mt_printPM(prog_char* x, byte p, byte l);

// Print an 8-bit number at position p line l
void mt_print8BitNumber(uint8_t number, byte p,
byte l);

// Tell the slave where to put the cursor
void mt_setCursor(uint8_t p, uint8_t l);

// Turn the cursor on
void mt_cursor();

// Turn the cursor off
void mt_noCursor();

// Turn the cursor blink on
void mt_blink();

// Turn the cursor blink off
void mt_noBlink();

// Turn the display on
void mt_display();

// Turn the display off
void mt_noDisplay();

// Put the cursor in the upper left position
void mt_home();

// Clear the LCD
void mt_clear();

// Clear LCD line 0
void mt_clear0();

// Clear LCD line 1
void mt_clear1();

// Clear the LCD line
void mt_clearLine();

// Get an 8-bit number
// Name on line 0 in str
// Number at end of line 1
uint8_t mt_get8BitNumber(char * str,uint8_t
start);

// Blocks waiting for a key to return
uint8_t mt_getKeyWait();

// gets the key
uint8_t mt_getKey();
```

These functions are all we need to write data to the I²C mini terminal LCD and to read the state of the five pushbuttons. Last month, we used the program *mini_terminal_master.ino* to test these functions. You can get that program and the MiniTerminal library from the article link.

As usual, the 'Arduino-simple' library functions hide some less simple programming concepts that we will now look at in a bit more depth. You really don't have to look at this if all you want to do is use the library to drive the LCD and pushbuttons on the IMT. However, if you want to get a bit of a peek under the hood and learn some more advanced concepts, then let's have a go at it.

How The IMT Software Works

In the section above, we saw several commands beginning with mt_ including: `mt_get_key();` `mt_home(); mt_clear(); mt_print();`, etc. Each of these functions uses the Arduino "Wire" library to communicate to the IMT via the I²C bus.

[Time for a rant: Naming an I²C library 'Wire' is probably one of the dumbest ideas ever, especially since the base Arduino code uses the 'wiring' library that is totally unrelated to the Wire library. Both are dumb names but they are great libraries, so let's accept them and move on ...]

Let's look at some of these functions in more detail, beginning with how we get the communications started. We see in the *MiniTerminal.h* module that we have defined the constant SLAVE as the number 42 [for the simple reason that it is the meaning of life, the universe, and everything ... at least according to *The Hitchhicker's Guide to the Galaxy*]. It really doesn't matter what number we choose, as long as both the Arduino master and the IMT slave agree as to what that number is.

We get the I²C ball rolling by running *Wire.begin()* in the *setup()* function. How simple is that? [This function does all the complex low level stuff to set up the ATmega328P on the Arduino to use the resident I²C peripheral.]

FIGURE 3. Say 'Hello world!'

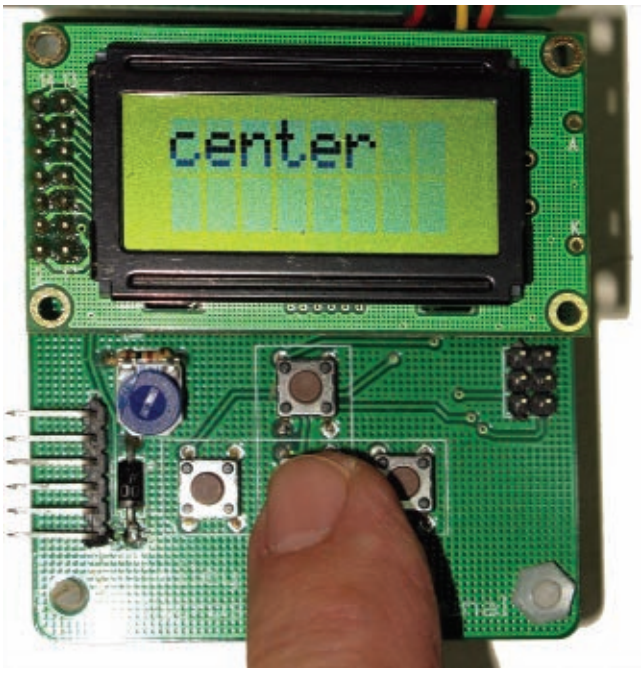


FIGURE 4. Center button pressed.

The Wire library provides us with several functions that let us request information from the slave, and to read from and write to the slave. In the *mini_terminal_master.ino setup()* function, we see:

```
void setup() // one time functions to get
started
{
  Wire.begin(); // join i2c bus (address
optional for master)

  mt_clear(); // Clear the LCD
  mt_print(Hello,0,0); // Print Hello on line 1
  mt_print(world,0,1); // Print world! on line 2
}
```

Calling these functions results in the IMT displaying the text shown in **Figure 3**.

After this, we enter the standard Arduino *loop()* function that asks the IMT slave every 100 milliseconds if any buttons have been pressed; if it returns a button different from the last button it returned, then the master tells the slave to display that button name in the top line of the LCD display as shown when the user presses the center button in **Figure 4**. The 'mt_' functions are all very similar in that they each send an I²C request to the slave in a pre-arranged format such as shown here:

```
// Tell the slave where to put the cursor
void mt_setCursor(uint8_t p, uint8_t l)
{
    Wire.beginTransmission(SLAVE); // start
buffering
    Wire.write("LCD"); // command for the LCD
    Wire.write(LCD_SET_CURSOR); // command name
    Wire.write(p); // cursor position
    Wire.write(l); // cursor line
    Wire.endTransmission(); // send the buffer
    delay(DELAY); // wait a moment to complete
}}
```

The function begins by calling the *Wire.beginTransmission* function that opens up some stuff (you never have to see) to buffer what you are going to send with the following statements. The *write* commands put stuff into that temporary buffer and the *Wire.endTransmission* function sends the data to the slave.

In the case shown above, the slave sees 'LCD' and uses that to select the LCD case statement. The LCD case statement then sees the 'C' which causes it to call the function that handles the cursor setting. That function then sees the 'p' and 'l' parameters and uses those to set the cursor to the indicated position and line.

Another *#!@%&*$! Bug!

As usual, everything is so easy that I had to spend a full day debugging the darn thing! It worked like a champ on the Arduino Diecimila but like a chump on the Arduino Uno. First, I did a bunch of Googling and found a plethora of folks who had problems with I²C code that worked on older Arduinos but not on the Uno. Unfortunately, none of their solutions worked for me. I did the next logical thing and added debugging statements that sent information out the USB serial port between each Wire call such as:

```
Serial.println("Before call to
endTransmission");
Wire.endTransmission();
Serial.println("After call to endTransmission");
```

This code started working ... THAT IS NOT SUPPOSED TO HAPPEN! What is supposed to happen is that I should see a bunch of 'before' and 'after' strings in the serial monitor. Then, the last 'before' I get that isn't followed by an 'after' should indicate the function where the code failed. The code no longer failed.

I started commenting out my *Serial.println* statements and got to the point that I found one that made things work if it had more than a certain number of characters, but not work if it had less than that number of characters. Well, that is just crazy. This is the point in debugging where you find out if you are really cut out for this stuff. If you start screaming and throwing things, maybe you should consider another pursuit. If you get excited because you are cornering a bug, then proceed with my blessings.

I figured that the serial functions must be providing time for something that was happening in the background. So, I started sprinkling around *delay()* functions until I hit on putting a delay after the *endTransmission* function — suddenly the program worked without the serial functions.

This, of course, made sense in retrospect since the *endTransmission* function is where the buffered characters are actually sent out the I²C bus. This means that the I²C peripheral has been told to go do some stuff, and the AVR core gets back to what it does while the peripheral does what it does. Previously, the I²C finished its job before I called it again, but not so with the Uno. On the Uno, there wasn't enough time for the stuff to get transmitted before I was calling another *Wire* function, so it somehow locked things up.

I could go and look into that Wire code since I have the source and probably find where it is locking up, but why bother? I found that adding a delay of five milliseconds got things working on the Uno, so I figured why not just move on and leave the Wire code to the experts?

Storing IMT Strings In Cheap Memory

Last month, we introduced the concept of using program memory to save in the SRAM and we saw how to use the *p_string*. This month, we'll repeat a bit of that and go into a lot more details for those who really want to know. One of the first things you may notice in the *mini_terminal_master.ino* application is the section beginning:

```
// Store strings in program memory
p_string(Hello) = "Hello";
…
```

This uses a macro *p_string* located in the *MiniTerminal.h* header file that lets us store strings of data in program memory. For microcontrollers, SRAM is used to run the program and tends to be precious and small, while the program memory tends to be stored in a cheaper and larger type of memory such as data Flash in the AVR. [For instance, the Arduino using an ATmega328P has 2K SRAM and 32K program memory.] However, the C (and C++) compilers gcc and g++ (used by the Arduino) will put character strings in the expensive limited SRAM unless we take special measures to put it in the cheap abundant program memory.

We want to save the SRAM as much as possible for our program to use when it is running, so we go to some trouble to put those strings in data Flash. The version of these compilers used by the Arduino has some special features that allow us to store and retrieve strings in the cheaper program memory. These features are somewhat arcane but let's take a look at them here — just remember that it isn't necessary for you to understand this to use it. As I said last month, it might seem a bit overkill to go to all that trouble since [in this example] we are only using a few short words, but in a real world application you might want to present a lot of information to the user, so it is good to have an option that won't run you out of memory.

The C compiler that is used by the Arduino has some things added to it that helps it compile code optimized for AVRs. One thing it provides is a set of tools for using the data Flash that calls program memory and uses the identifier: PROGMEM. The features are located in the header: pgmspace.h file.

You can get detailed documentation about this from the avr-libc manual that you may be surprised to learn is already loaded on your computer if you have the Arduino IDE installed. On my computer, I have the Arduino IDE on the C drive, so my copy is located at: C:\Arduino-1.0.4\hardware\tools\avr\doc\avr-libc.

If you want to move beyond the Arduino, this manual is one place to learn about the sorts of tools that underpin the Arduino. Open the *MiniTerminal.h* header file and you will see that it has:

```
#include <avr/pgmspace.h>
```

We need this so that we can define the *p_string* macro:

```
// Use to hide complexity of locating data in
program memory
#define p_string(text) prog_char (text)[]
PROGMEM
```

A macro is a string of characters that follow a *#define* — like the *p_string(text)* above. The compiler substitutes the characters that follow the macro; in the above case,

Header declares: #define p_string(text) prog_char (text)[] PROGMEM

User writes: p_string(Hello) = "Hello";

Compiler sees: prog_char (Hello)[] PROGMEM = "Hello";

FIGURE 5. The *p_string* macro in action.

it's *prog_char (text)[] PROGMEM* for the macro characters that follow the *#define*. It also lets us create a sort of variable for the macro such as the 'text' in the *p_string* macro, and writes the contents of this variable to the location of the 'text' in the macro substitution string. This is less complicated than it sounds. For example:

```
p_string(Hello) = "Hello";
```

As shown in **Figure 5**, the *#define* instructs the compiler to do the following:

For: `p_string(text)`
Substitute: `prog_char (text)[] PROGMEM`
So when the compiler sees: `p_string(Hello)`
It substitutes: `prog_char (Hello)[] PROGMEM`
And it leaves the text that follows alone: `= "Hello";`
Thereby translating: `p_string(Hello) = "Hello";`
To: `prog_char (Hello)[] PROGMEM = "Hello";`

So, what we have done is create an Arduino-simple macro that translates: `p_string(Hello) = "Hello";`

To the less simple: `prog_char (Hello)[] PROGMEM = "Hello";`

The first version that I'm calling Arduino-simple is something that I think we can easily remember to use to store strings in program memory. The compiler-translated version of that line is what is needed by the compiler to actually store "Hello" in program memory. Of course, you could use the second version without having to use the macro, but frankly, I think I'm more likely to remember that I can store a string in cheap memory with the *p_string(mystring) = "mystring";* than the version the compiler wants. We use this concept with the *mt_print* function as follows:

Store strings in program memory:

```
p_string(Hello) = "Hello";
p_string(world) = "world!";
```

Then print them with:

```
mt_printPM(Hello,0,0); // Print Hello on line 1
mt_printPM(world,0,1); // Print world! on line 2
```

The words 'Hello' and 'world!' appear on the LCD as shown in **Figure 3**. To me, that IS Arduino-simple.

Under the hood, some more complex things are going on so that we can use the strings stored in program memory. In the *MiniTerminal.cpp* module, we have the *mt_printPM* function:

```
void mt_printPM(prog_char* x, byte p, byte l)
{
  mt_setCursor(p,l);

  for (int i=0; i < 8; i++) {
    buffer[i] = (char)pgm_read_byte(&x[i]);
  }
  buffer[9] = '\0';

  Wire.beginTransmission(SLAVE);
  Wire.write("LCD:");
  Wire.write(buffer);
  Wire.endTransmission();
  delay(DELAY);
}
```

In this function, we are using some things from avr-libc (a set of open source tools that the Arduino uses when it compiles programs). We see that we are using the datatype from avr-libc: *prog_char* x* — a pointer to a string named 'x' in program memory — as a parameter to the function. In the function body, we see that we are using that string as an array and reading it into a buffer (temporarily using SRAM) one byte at a time with *pgm_read_byte* also from avr-libc. We read eight bytes since that is how many characters the LCD can hold. We then terminate the buffer with *'\0'* — the NULL character that tells C that a string has come to an end.

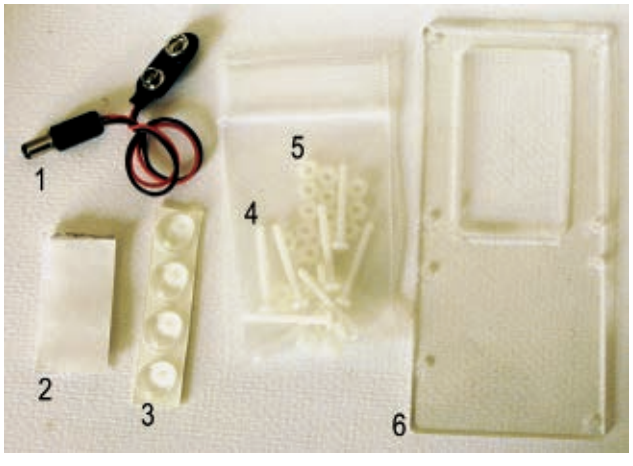


FIGURE 6. Arduino handheld base parts.

PN	#/kit	Part Name
1	1	Nine volt battery snap with 5.5 x 2.5 mm barrel
2	1	Velcro strip
3	4	Bumpers
4	7	1" nylon bolt #6
5	21	Nylon nuts #6
6	1	Acrylic Base

Reference Guide for Figure 6.

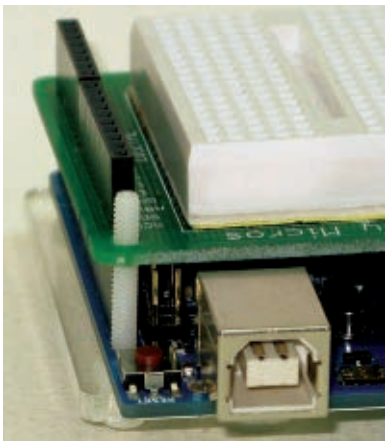


FIGURE 9. R3 bolt.

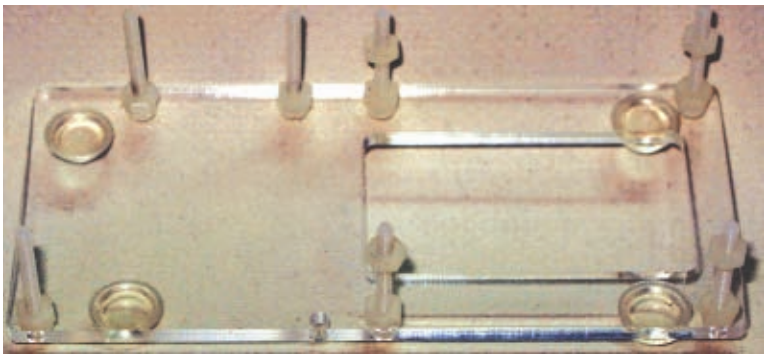


FIGURE 7. Base with bolts.

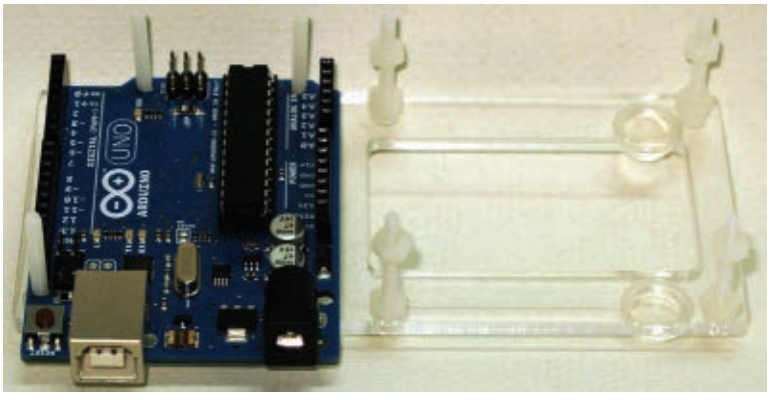


FIGURE 8. Arduino on base.

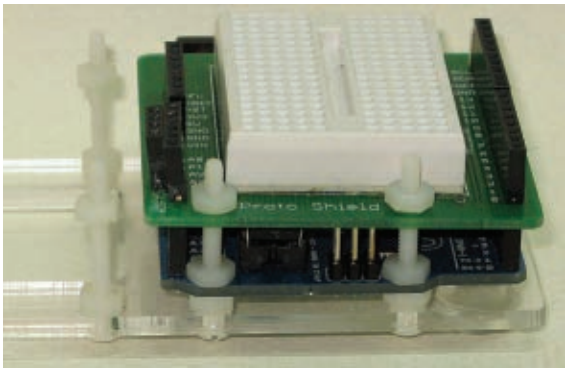


FIGURE 10. Arduino proto shield on base.

Now, we can use that temporary buffer to send the bytes to the MiniTerminal and have them displayed on the LCD. All the IMT sees is the stream of bytes; it knows nothing about how they were stored by the sender. If you think this is complicated, you should take a look at the PROGMEM stuff in avr-libc to see what is even deeper under the hood there.

The *p_string* macro we are using provides two layers of simplification above the truly difficult stuff in avr-libc. If we just want to use the LCD and don't really care how it works, then using these layers makes sense. If we can use the computer to do the hard stuff to simplify our lives, why not do so?

So, this takes us from last month's Arduino-simple high level view of how to use the IMT to the intermediate level libraries, and hints at the further complexities in the next layer down in avr-libc and the Wire library. You should have some further insight about the software abstraction process that puts progressively simpler layers on top of the complex underlying code, then lets the computer deal with the hard stuff while allowing the user to get the job done (wasting as few brain cells as possible).

Some guru types are going to try to make you feel guilty about doing this the easy way, claiming you ought to use real C (or assembler or whatever) or go home. Just smile at them, while you actually make things work using the best (read simplest) tools available.

Building The Arduino Handheld Prototyper

The Arduino handheld prototyper consists of three main parts, two of which we have already seen. First, there is the Arduino proto shield that we looked at in great detail beginning in the December 2012 *Nuts & Volts*. The second part is the IMT that we began looking at last month. The third part is the base kit (**Figure 6**) that ties together the first two parts, giving us an Arduino, a proto shield with a mini breadboard, and an IMT with an 8x2 character LCD and five pushbuttons that (taken all together) can be used to prototype devices.

The base kit has a plastic base board, lots of nylon nuts and bolts, a nine volt battery connector, some Velcro™ to attach the battery, and some bumper feet. [You can get the Arduino proto shield, the IMT, and the handheld prototyper from the *Nuts & Volts* webstore.]

The components in **Figure 6** are intended to use with both the older style Arduinos and the newer Uno R3 with expanded connections. We have sufficient nylon nuts and bolts to build either style, but here we'll see the Uno R3 being built. **Figure 7** shows the plastic base with the bolts in the holes held in place by nuts (also the rubber bumpers are on the bottom, but show through the clear plastic). Note that there is no bolt in the hole that corresponds to the Arduino hole directly next to the

battery connector. This is because there is no corresponding hole on the Arduino proto shield (which doesn't matter since the shield is mostly held in place by all the male/female pass through connections and the other bolts). We see the Arduino placed on the bolts in **Figure 8**. The two bolts at the top of the photo should have nuts added to hold down the Arduino. The bolt nearest the USB (in **Figure 9**) may have a nut added on the older models, but in the Uno R3 the extra connections are too close to allow it. We can see the Arduino proto shield added in **Figure 10**. Note that the nuts are on top only, since they are not needed underneath the PCB to support the board since it is already supported by the male/female shield connections.

We saw in **Figure 7** that there is an extra set of nuts about three-quarters of the way up the shaft of the bolt. These are used to support the IMT as shown in **Figure 11** (with the LCD removed).

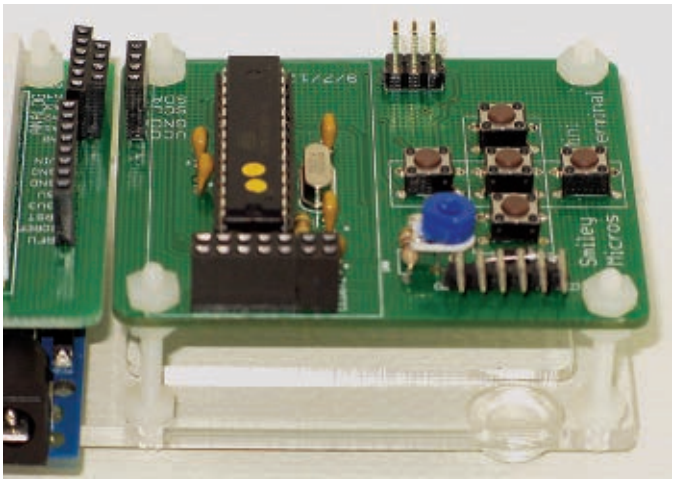


FIGURE 11. I²C mini terminal PCB bolted on base.

I²C Connection For Old Arduinos Versus New Arduino Uno R3

There are two types of I²C connections for the Arduino (at this writing). For the newer Arduino Uno R3, we wire the four-pin female connectors straight onto each other as shown in **Figure 12**. For the older non-R3 Arduinos, we connect analog A4 to SDA, and analog A5 to SCL as shown in **Figure 13**. Notice that the I²C wires cross each other in **Figure 13** but are parallel in **Figure 12**.

Finally, we add one side of the Velcro to the bottom of the IMT PCB and the other to a nine volt battery. Then, we add the snap connector to the battery and the barrel connection to the Arduino as shown in **Figure 14**. Be sure to unplug the battery from the Arduino when you aren't using it since the Arduino may be running (you can't see the 'on' LED under the proto shield) and you will deplete the battery.

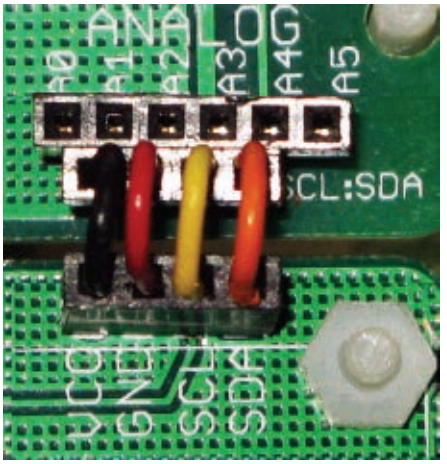


FIGURE 12. I²C R3 connection.

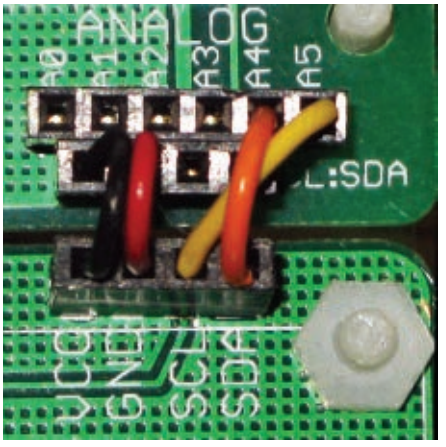


FIGURE 13. I²C non-R3 connection.

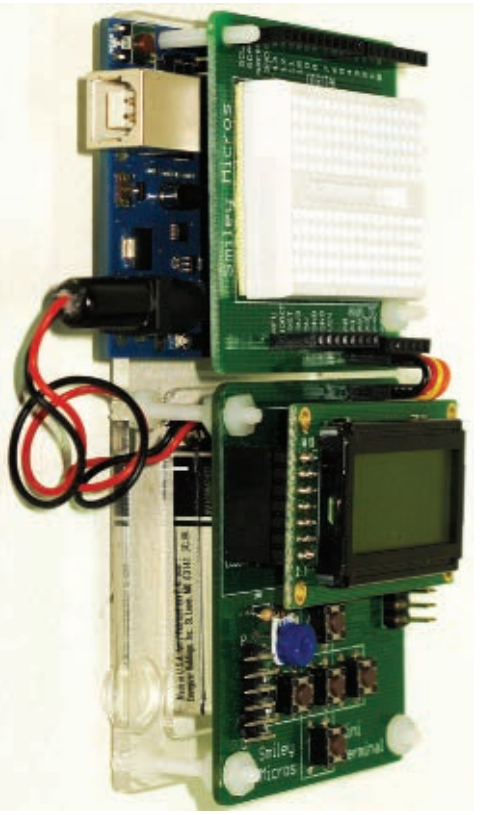


FIGURE 14. Assembled with battery.

Wrap-Up

The main goal for this device is to help you prototype your own designs. I'm always amazed at the stuff folks are doing, and I'm looking forward to hearing from readers who use this device to see what sorts of things are developed.

Last month, I promised to discuss how to create a fresh air controller for a castle, but I'm putting that off till next month when we will apply the handheld prototyper to a practical project. This is a perfect application for our new tool, even if you don't own a castle. **NV**

CLASSIFIEDS

KITS/PLANS

LIGHTING

ROBOTICS

COMPONENTS

SURPLUS

KIT BUILDING

HARDWARE WANTED

ENERGY SOLUTIONS

ENGINEERING/ DESIGN SVCS

OSCILLOSCOPES

GREAT FOR DIYers!

Getting Started with chipKIT
by Chuck Hellebuyck

In this book, Chuck Hellebuyck — who has authored many entry level technical books — shows you how to get started with the chipKIT Uno32 using some very simple example sketches (sketches are a software program in the Arduino world) that demonstrate how to use digital inputs, digital outputs, analog inputs, and analog outputs. These are the fundamental building blocks every electronic project needs. With the examples in this book, you'll have what you need to get your electronic project up and running.

Reg Price $19.95
Sale Price $18.50

How to Diagnose and Fix Everything Electronic
by Michael Jay Geier

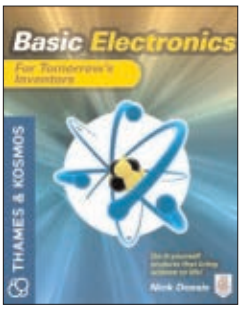
Master the Art of Electronics Repair!

In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear. **About the Author:** Michael Jay Geier began operating a neighborhood electronics repair service at age eight that was profiled in *The Miami News*.

$24.95

Basic Electronics for Tomorrow's Inventors
by Nick Dossis

Learn all about electronics with hands-on experiments and projects.

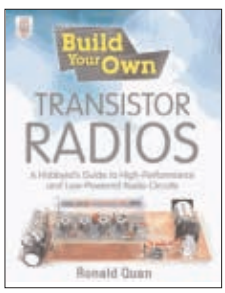


This great book features a variety of simple, solder-free projects, including an LED reading light, electronic security keypad, IR target practice game, a real working telephone, temperature and moisture sensors, spy gadgets, and other neat stuff. Best of all, these experiments require only plug-and-play "breadboards" and other commonly available parts.

$19.95

Build Your Own Transistor Radios
by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits

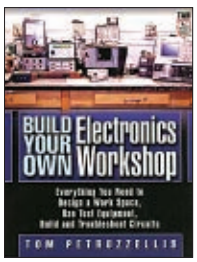


Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.

Paperback, 496 pages
$49.95

Build Your Own Electronics Workshop
by Thomas Petruzzellis

BUILD YOUR OWN DREAM ELECTRONICS LAB!



This value-packed resource provides everything needed to put together a fully functioning home electronics workshop! From finding space to stocking it with components to putting the shop into action — building, testing, and troubleshooting systems. This great book has it all! And the best part is, it shows you how to build many pieces of equipment yourself and save money, big time!

Reg Price $29.95
Sale Price $26.95

Beginner's Guide to ... Programming the PIC24/dsPIC33
by Thomas Kibalo

Kibalo takes you step by step through the fundamentals of programming the PIC24H which can equally be applied to the dsPIC33. His clear explanation of the inner workings make learning the PIC24H/dsPIC33 16-bit architecture easy. His code examples demonstrate how to perform the functions most applications require. The hardware is shown in a simple breadboard setup so even a beginner can build it, along with very few extra components needed.

$39.95*

HTML: A Beginner's Guide
by Wendy Willard

Create highly functional, impressive websites in no time. Fully updated and revised, *HTML: A Beginner's Guide*, Fourth Edition explains how to structure a page, place images, format text, create links, add color, work with multimedia, and use forms. You'll also go beyond the basics and learn how to save your own web graphics, use Cascading Style Sheets (CSS), create dynamic web content with basic JavaScript, and upload your site to the web. By the end of the book, you'll be able to build custom websites using the latest HTML techniques.

$29.95

Programming PICs in Basic
by Chuck Hellebuyck

If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

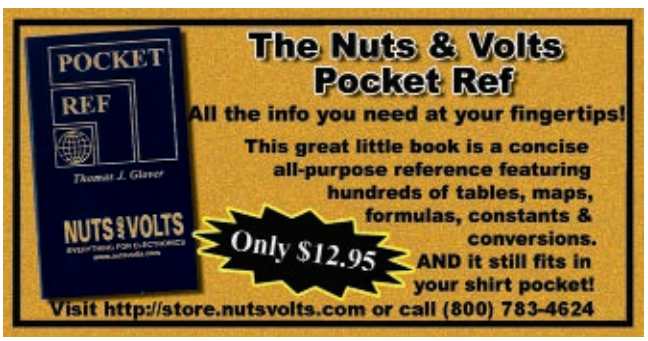
Master and Command C for PIC MCUs
by Fred Eady

Master and Command C for PIC MCU, Volume 1 aims to help readers get the most out of the Custom Computer Services C compiler for PIC microcontrollers. The author describes some basic compiler operations that will help programmers particularly those new to the craft create solid code that lends itself to easy debugging and testing. As Eady notes in his preface, a single built-in CCS compiler call (output_bit) can serve as a basic aid to let programmers know about the "health" of their PIC code.

$14.95

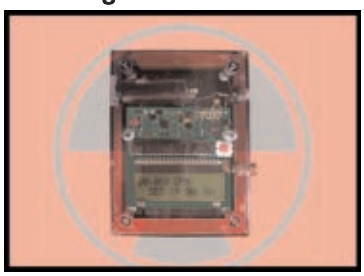
BOOK & KIT COMBOS

CD-ROM SPECIAL



PROJECTS

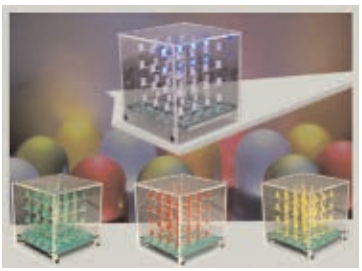
Geiger Counter Kit



This kit is a great project for high school and university students. The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr. The LND712 tube in our kit is capable of measuring alpha, beta, and gamma particles.

Subscriber's Price **$145.95**
Non-Subscriber's Price **$148.95**

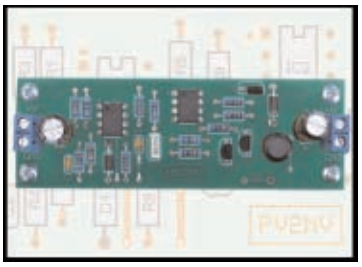
3D LED Cube Kit



This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue

Subscriber's Price **$57.95**
Non-Subscriber's Price **$59.95**

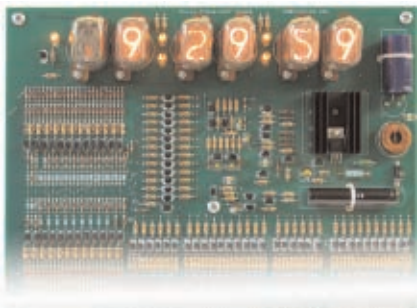
Inverter DC-DC Converter Kit



If you need +12V and -12V, but all that's available is +12V, then this project is for you. The inverted DC-DC converter gives you -V out when you put +V in, and works over a range of voltages without adjustment. It's an easy to build voltage mirror that can supply over 100 mA without a significant drop in voltage.

Subscriber's Price **$29.95**
Non-Subscriber's Price **$32.95**

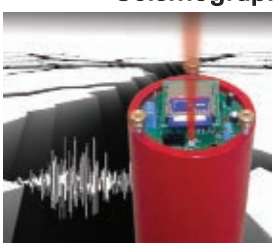
Neon Transistor Clock Kit



Add HIGH VOLTAGE to your clock!
This is a Nixie Tube display version of the Transistor Clock. It uses only discrete components — no integrated circuits. For more info, see the April 2012 issue.

Subscriber's Price **$245.95**
Non-Subscriber's Price **$249.95**

Seismograph Kit

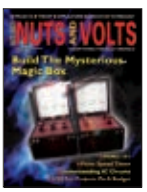


As seen in the May 2012 issue. Now you can record your own shaking, rattling, and rolling.

The Poor Man's Seismograph is a great project /device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

Subscriber's Price **$79.95**
Non-Subscriber's Price **$84.95**

Magic Box Kit

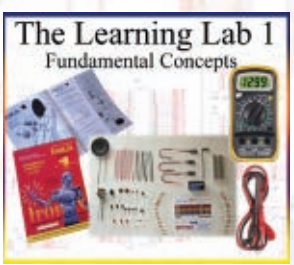


As seen on the April 2007 cover.

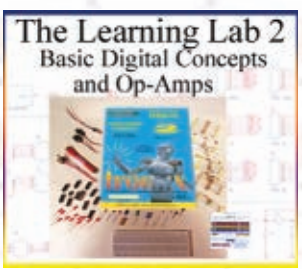
This unique DIY construction project blends electronics technology with carefully planned handcraftsmanship. This clever trick has the observer remove one of six pawns while you are out of the room and upon re-entering you indicate the missing pawn without ever opening the box.

Subscriber's Price **$39.95**
Non-Subscriber's Price **$45.95**

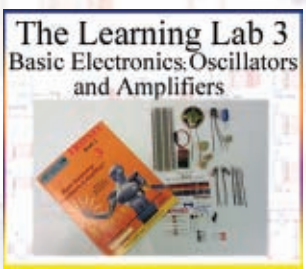
FOR BEGINNER GEEKS!



$59.95



$49.95



$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board.

As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and a promotional video, please visit our webstore.

READER-TO-READER TECHFORUM

All New TECHFORUM Online At www.nutsvolts.com/tech-forum

>>> QUESTIONS

Diode Selection On Multimeter

Why is there a diode selection on multimeters? What does the value mean?

#8131 **George Powelson**
Ogden, UT

Capacitor Types

How do the different types of capacitors (ceramic, electrolytic, tantalum, etc.) differ and why would you use one type over another in a circuit?

#8132 **Jim Stanton**
Ft Meyers, FL

Antenna Length

Can someone explain how adding inductance or capacitance to an antenna changes the length?

#8133 **Henry Stewart**
Spokane, WA

Sine Wave

I would like to know how you can get a positive and a negative part of a sine wave from a circuit that runs on a nine volt battery.

#8134 **George Powelson**
Ogden, UT

Circuit Simulation Program

I'm looking for a circuit simulation program that is inexpensive and easy for a novice to learn. I'm mostly interested in analog circuits, but I do dabble with digital from time to time. I want to be able to change component values and analyze circuit behavior. This is for my own enjoyment and education, not for professional use. Can someone recommend a program they are using or have prior experience with?

#8135 **Justin Lange**
Bowling Green, KY

>>> ANSWERS

[#6136 - June 2013]
Cell Phone Battery Cable

How can I make a cable so that I could use a nine volt battery for my cell phone? Would a large six volt lantern battery work? Are there any electronics needed?

#1 First, determine how long you want the phone to run, and choose a battery with enough milliampere hours for the duration. (A nine volt radio battery may be inadequate).

That said, for efficiency I recommend a switching converter such as the On Semi MC34063. See the datasheet for how to use it. For an inexpensive inductor, consider the toroidal inductor CR-345 from **AllElectronics.com**. This should have more than enough capacity to not run dry. If, in the future, you need a voltage inverter, be aware that I have found other datasheets to be in error with their circuit for this part, but the ON Semi circuit works well. Simply use the formulas in the datasheet. If the source voltage is close to 12 volts, then a surplus car adapter for your phone from Goodwill is the low cost option.

Jim Lacenski
Bellevue, WA

#2 How about a 6V lantern battery? First, a 9V "transistor" battery doesn't have enough current to run a cell phone.

The lantern battery might work. Look at your cell phone battery; it will be a li-ion probably 3.7V, so you will need to lower the voltage of the lantern battery to that.

Next, measure or just read on the battery which pins are + and -. Then, figure out how to connect to the small pins/strips inside the phone. In one of mine, I found that using the little hook-type clips worked fine.

That should do it. If you need to learn how to make a voltage reducing circuit, I believe that both linear types using an LM317 chip — as well as some of the newer switching chips — have already been published in *N&V*. I found the LM2575 at **http://pdf.datasheetcatalog.com/datasheet/SemtechCorp/mXvqxtq.pdf**

Philip Karras KE3FL
via email

[#7131 - July 2013]
Voltage Filter

I have a 40 year old Chevy. The fuel gauge bounces half the gauge reading when it gets under a 3/4 tank. I put in a new sending unit but that did not help. I tried a digital gauge and that helped some, but not enough. It is a 90 ohm sending unit, zero ohms (grounded) = empty and 90 ohms = full. I would like to find a circuit that will filter out the high and low spikes, and put out an average voltage reading based on the sending unit's position.

Send all questions and answers by email to **forum@nutsvolts.com** or via the online form at **www.nutsvolts.com/tech-forum**

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES WHATSOEVER** are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

#1 As you know, a gas sending unit is just a variable resistor acting as a voltage divider. There are several possible reasons for a jumpy reading. One would be if the float is not well placed in the tank and the gas sloshing around just moves it up and down. If that is the case, a filter may help.

Since the gas level should change very slowly, a simple series connected capacitor/resistor combination connected from the output of the sending unit to ground would be a good filter. A large electrolytic with a high enough voltage rating to withstand any spikes in the 12V system would be best.

You need a time constant that is longer than the period of the gauge's jumps; perhaps a couple of seconds. You also need a resistance that is small enough to allow enough current to pass through it when needed to maintain the dial reading. TC (Time Constant) = R * C or C = TC/R. R is the value of the series resistor. I would take it to be a few ohms, perhaps 10 as a first try.

It is a bit hard to calculate the power dissipated in this resistor as it depends on the frequency and severity of the fluctuations, but I would start with at least a one watt size and if that burns up, try five watts. With a one second time constant, we have: C = 1/10 = 0.1 farads. That is 100,000 µF which is probably impractical. I would try a nice large electrolytic and see what happens; 20,000 µF at 25 volts would be a good first choice.

If that helps but is not enough, go larger. The negative (-) terminal on the capacitor would go to ground. It does not matter which side of the capacitor the resistor is on, as long as it is in series with it. This will not affect the accuracy of the readings, as long as the added resistor is not in series with the line to the gauge.

Another possible cause would be a bad sending unit, but you say you have already replaced it so that is not likely. A third cause would be bad wiring between the sending unit and the gauge. Before you try anything else, you might run an extra temporary wire to see if it helps. If it does, repair the harness.

Finally, it could be a bad gauge. After trying the above filter and extra wire fixes, you could temporarily rig a 100 ohm pot at the dashboard to activate the meter. Disconnect the wire from the sending unit and observe the meter as you drive down a bumpy street. If it still jumps, it is the gauge itself and that must be replaced.

#2 On my old Dodge, (and this applies to Fords and Chevys, too) the instrument panel has a thermal voltage regulator. The instruments operate at a voltage of about 5-7 volts. The gauges have built-in damping.

The thermal voltage regulator is nothing more than a bi-medal strip with a heating coil on it that opens and closes a contact, thus maintaining a poor — but semi constant — voltage. I've replaced these devices with simple voltage regulators like an LM317 set to a voltage that approximates six volts. More reading is available at **www.chevytrucks.org/tech/gasgauge.htm**

Mentioned in the above link is to check for a good ground at the sender in the tank.

Ray
Vancouver, BC

[#7132 - July 2013]
Tech Jobs

I'm trying to get my 18 year old interested in a technical career. I need opinions as to what skills are the most marketable and have the best chance of employment in today's reality. Are online or off-campus trade schools worth the money, or do companies favor traditional college degrees?

Technical careers can vary far and wide, and many of them do almost require a degree of some kind. If he/she is interested in engineering, then a degree will be extremely helpful. An easy way to figure out whether it's necessary is to take a look at local technical job listings. Unfortunately, the degree is often very arbitrary. It's required to get hired, but most of what you learned to get the degree is useless.

I recommend trying to test out of as many classes as possible. Look into CLEP testing through the College Board, as well as specific college's Credit by Exam terms. If programming is more up his/her alley, then a degree is likely very necessary to get hired. However, unless he/she can learn very well from professors, I recommend "playing" with as much code outside of college as possible. Some of the top (non-web) languages in demand are: Java, Python, C/C++, C#, and Perl.

One of the few tech jobs that does not require a degree is freelance or semi-freelance web design/development. You can learn all that you need to know through free and low cost resources. You can also get a degree in web design/development, and that might help you get hired by a design house.

The absolutely required technologies to know for web are: HTML, CSS, Javascript+JQuery, and SQL. SQL doesn't get you anywhere without a server-side scripting language such as PHP or Ruby on Rails. I am admittedly biased toward freelancing web design, as that is what I do.

Web design has one of the biggest markets for selling yourself in, and a lot of money can be made. It's certainly not for everyone, though. It requires that you keep up with current technology, are good with selling to people, and are willing to put some hard (but fun!) work in. Have a look at this link for learning (web and non-networked) programming: **antsmagazine.com/web-development/20-amazing-places-to-learn-code-from-scratch**. Ignore W3Schools and anyone who says they are a good resource; **codecademy.com** is where I first learned HTML and CSS (then did it again in college).

Sam
Jersey Shore, PA

■ LOOK FOR ■ SEARCH FOR ■ FIND

Find your favorite advertisers here!

ADvertiser INDEX

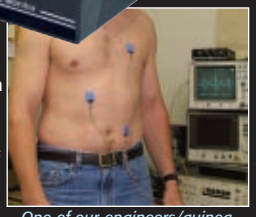
Beat The Heat...Build a Kit!

Ramsey Kits Are Always Neat, Even In The Dog Day Heat!

Electrocardiogram ECG Heart Monitor

- ✔ **Visible and audible display of your heart rhythm!**
- ✔ **Bright LED "Beat" indicator for easy viewing!**
- ✔ **Re-usable hospital grade sensors included!**
- ✔ **Monitor output for professional scope display**
- ✔ **Simple and safe 9V battery operation**

When we think summer, we normally think of vacations, traveling, and all the activities you have been waiting for all winter! And whether that includes hiking that new trail you've heard about or simply riding the new rides (and waiting in line in the scorching heat!) at Wally World, there WILL be physical exertion involved! While we are frequently reminded that February is national Heart Smart month, we think every month should be Heart Smart month. Heart Smart is a way of life, and certainly shouldn't be limited to one month a year. We kept that in mind when we designed the ECG1!

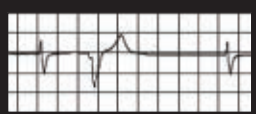


One of our engineers/guinea pigs, checking his heart!

Not only will building an actual ECG be a thrill, but you'll get hands-on knowledge of the relationship between electrical activity and the human body. Each time the human heart beats, the heart muscle causes small electrical changes across your skin. By monitoring and amplifying these changes, the ECG1C detects the heartbeat and allows you to accurately display it, and hear it, giving you a window into the inner workings of the human heart and body!

Use the ECG1C to astound your physician with your knowledge of ECG/EKG systems. Enjoy learning about the inner workings of the heart while, at the same time, covering the stage-by-stage electronic circuit theory used in the kit to monitor it. The three probe wire pick-ups allow for easy application and experimentation without the cumbersome harness normally associated with ECG monitors.

The documentation with the ECG1C covers everything from the circuit description of the kit to the circuit description of the heart! Multiple "beat" indicators include a bright front panel LED that flashes with the actions of the heart along with an adjustable level audio speaker output that supports both mono and stereo hook-ups. In addition, a monitor output is provided to connect to any standard oscilloscope to view the traditional style ECG/EKG waveforms just like you see in a real ER or on one of the medical TV shows!



Look what I found!

On a personal note... See the display to the left? That's me! In between writing this montly ad copy, catalog copy, and plethora of other tasks here, I noticed some skipped beats in my pulse! An immediate cardiac check found I had something called Trigeminy, or PVCs that occur at intervals of 2 normal beats to one PVC! And I saw it with our ECG1 kit!

The fully adjustable gain control on the front panel allows the user to custom tune the differential signal picked up by the probes giving you a perfect reading and display every time! 10 hospital grade re-usable probe patches are included together with the matching custom case set shown. Additional patches are available in 10-packs. Operates on a standard 9VDC battery (not included) for safe and simple operation. Note, while the ECG1C professionally monitors and displays your heart rhythms and functions, it is intended for hobbyist usage only. If you experience any cardiac symptoms, seek proper medical help immediately!

ECG1C	**Electrocardiogram Heart Monitor Kit With Case & Patches**	**$44.95**
ECG1WT	**Electrocardiogram Heart Monitor, Factory Assembled & Tested**	**$89.95**
ECGP10	**Electrocardiogram Re-Usable Probe Patches, 10-Pack**	**$4.95**

Digital Voice Changer

This voice changer kit is a riot! Just like the expensive units you hear the DJ's use, it changes your voice with a multitude of effects! You can sound just like a robot, you can even ad vibrato to your voice! 1.5W speaker output plus a line level output! Runs on a standard 9V battery.

MK171 **Voice Changer Kit** **$14.95**

Digital Audio Recorder

Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.

K8094 **Audio Recorder/Player Kit** **$32.95**

5A PWM Motor Controller

This handy controller uses a pulse width modulated output to control the speed of a motor without sacrificing torque! Handles a continuous current of 5A and includes an LED to indicate speed, as well as an oversized gold heatsink! Also available factory assembled.

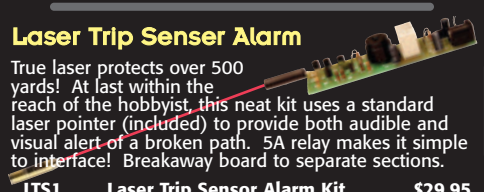
CK1102 **5A PWM Motor Controller Kit** **$14.95**

Sniff-It RF Detector Probe

Measure RF with your standard DMM or VOM! This extremely sensitive RF detector probe connects to any voltmeter and allows you to measure RF from 100kHz to over 1GHz! So sensitive it can be used as a RF field strength meter!

RF1 **Sniff-It RF Detector Probe Kit** **$27.95**

Laser Trip Senser Alarm



True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.

LTS1 **Laser Trip Sensor Alarm Kit** **$29.95**

Electronic Watch Dog

A barking dog on a PC board! And you don't have to feed it! Generates 2 different selectable barking dog sounds. Plus a built-in mic senses noise and can be set to bark when it hears it! Adjustable sensitivity! Unlike the Saint, eats 2-8VAC or 9-12VDC, it's not fussy!

K2655 **Electronic Watch Dog Kit** **$39.95**

The Learning Center!

Beginners To Advanced... *It's Fun!*

- ✔ **Learn and build!**
- ✔ **130, 200, 300, & 500 in one electronic labs!**
- ✔ **Practical through hole and SMT soldering labs!**
- ✔ **Integrated circuit AM/FM radio lab!**
- ✔ **Fuel Cell, Solar Hydrogen, and Bio-Energy labs!**
- ✔ **Beginner's non-soldering kits!**

For over 3 decades we've become famous for making electronics fun, while at the same time making it a great learning experience. As technology has changed over these years, we have continued that goal!

PL130A Gives you 130 different electronic projects together with a comprehensive learning manual describing the theory behind all the projects.

PL200 Includes 200 very creative fun projects and includes a neat interactive front panel with 2 controls, speaker, LED display and a meter.

PL300 Jump up to 300 separate projects that start walking you through the learning phase of digital electronics.

PL500 The ultimate electronics lab that includes 500 separate projects that cover it all, from the basics all the way to digital programming.

SP3B Whether young or old, there's always a need to hone your soldering skills. Either learn from scratch or consider it a refresher, and end up with a neat little project when you're done!

SM200K Move up to Surface Mount Technology (SMT) soldering, and learn exactly how to solder those tiny little components to a board!

AMFM108K We not only take you through AM and FM radio theory but we guide you through IC's. When you're done you've built yourself an IC based AM/FM radio that works great!

KNS10 With a reversible PEM fuel cell that combines electrolysis and power conversion into a single device you end up building your own fuel cell car! Learn tomorrows technology today!

KNS11 Learn alternative fuel technology while you build your own H-Racer car and refueling station!

KNS13 Convert ethanol alcohol to run a PEM fuel cell and watch it all work in front of your eyes!

KNS1 A great beginner's kit for the dinosaur enthusiast in the family, young and old! A wooden hobby kit that teaches motor and gear driven operation that requires no soldering.

PL130A	**130-In-One Lab Kit**	**$39.95**
PL200	**200-In-One Lab Kit**	**$84.95**
PL300	**300-In-One Lab Kit**	**$109.95**
PL500	**500-In-One Lab Kit**	**$249.95**
SP1A	**Through Hole Soldering Lab**	**$9.95**
SM200K	**SMT Practical Soldering Lab**	**$22.95**
AMFM108K	**AM/FM IC Lab Kit & Course**	**$36.95**
KNS10	**Fuel Cell Car Science Kit**	**$82.95**
KNS11	**H-Racer & Refueling Station Kit**	**$144.95**
KNS13	**Bio-Energy Fuel Cell Kit**	**$129.95**
KNS1	**Tyrannomech Motorized Kit**	**$17.95**

ramsey
www.ramseykits.com

Where Electronics... Is Always Fun!

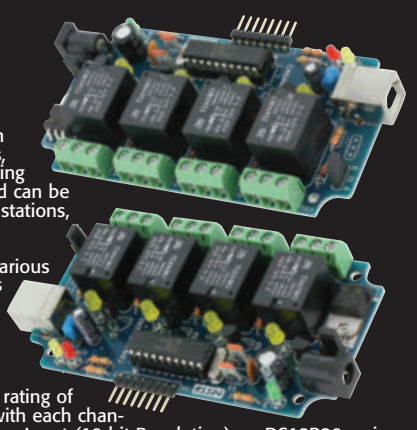
- ✔ *Build It!*
- ✔ *Learn It!*
- ✔ *Achieve It!*
- ✔ *Enjoy It!*

Follow Us and SAVE $$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!

4-Channel USB Relay Controller Board

- ✔ **Individually configurable I/O channels!**
- ✔ **Compatible with DS18B20 temp sensors!**
- ✔ **USB control for your custom applications!**



This professional quality USB relay controller and data acquisition module allows computer controlled switching of external devices, plus full bi-directional communication with the external world using the USB port of your computer. The controller is very flexible and can be used for a wide range of custom applications, including weather stations, temperature monitoring, logging and control, etc.

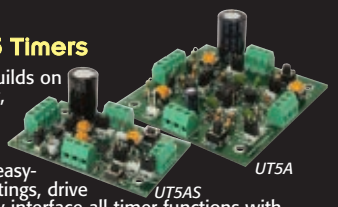
It is compatible with both Windows and Apple OS X, as well as various Linux flavors. When you plug it into your computer, it appears as a USB CDC device that creates a Virtual Serial (COM) port allowing easy communication with the board through any programming language that supports serial communications (VB, VB.NET, C#, C, C++, Perl, Java, etc).

The controller features four onboard relay outputs with a current rating of 10A each. Also onboard is a 6-channel Input/Output interface, with each channel individually configurable as Digital Input, Digital Output, Analog Input (10-bit Resolution), or DS18B20 series Temperature Sensor.

In Digital Input/Output modes, each channel can support a TTL compatible or ST input or a 5V output signal. In Analog Input mode, each channel can convert a voltage of between 0 to 5V into a 10-bit digital representation. Finally, in Temperature Sensor mode, each channel can be connected to a DS18B20 series Digital Temp Sensor.

UK1104 **4-Ch USB Relay Interface Kit** **$59.95**

Ultimate 555 Timers



This new series builds on the classic UT5 kit, but takes it to a whole new level! You can configure it on the fly with easy-to-use jumper settings, drive relays, and directly interface all timer functions with onboard controls or external signals.

All connections are easily made though terminal blocks. Plus, we've replaced the ceramic capacitor of other timer kits with a Mylar capacitor which keeps your timings stable over a much wider range of voltages! Available in through hole or surface mount versions! Visit www.ramseykits.com for version details.

UT5A **Through Hole 555 Timer/Osc Kit** **$21.95**
UT5AS **SMT 555 Timer/Osc Kit** **$29.95**

OBDII *CarChip Pro*

The incredible OBDII plug-in monitor that has everyone talking! Once plugged into your vehicle it monitors up to 300 hours of trip data, from speed, braking, acceleration, RPM and a whole lot more. Reads and resets your check engine light, and more!

8226 **CarChip Pro OBDII Monitor-Asmb** **$79.00**

RF Preamplifier

The famous RF preamp that's been written up in the radio & electronics magazines! This super broadband preamp covers 100 KHz to 1000 MHz! Unconditionally stable gain is greater than 16dB while noise is less than 4dB! 50-75 ohm input. Runs on 12-15 VDC.

SA7 **RF Preamp Kit** **$16.95**

Touch Switch

Touch on, touch off, or momentary touch hold, it's your choice with this little kit! Uses CMOS technology. Actually includes TWO totally separate touch circuits on the board! Drives any low voltage load up to 100mA. Runs on 6-12 VDC.

TS1 **Touch Switch Kit** **$9.95**

Doppler Direction Finder

Track down jammers and hidden transmitters with ease! 22.5 degree bearing indicator with adjustable damping, phase inversion, scan and more. Includes 5 piece antenna kit. Runs on 12VDC vehicle or battery power.

DDF1 **Doppler Direction Finder Kit** **$169.95**

Mad Blaster Warble Alarm

If you need to simply get attention, the "Mad Blaster" is the answer, producing a LOUD ear shattering raucous racket! Super for car and home alarms as well. Drives any speaker. Runs on 9-12VDC.

MB1 **Mad Blaster Warble Alarm Kit** **$9.95**

Laser Light Show

Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1 **Laser Light Show Kit** **$49.95**

USB PIC Programmer

Finally, a compact USB PIC Programmer with a 20 pin ZIF socket for easy programming of most Microchip PIC Flash devices that does not require low voltage programming. Plus it uses USB therefore no more RS232 compatibility blues!

CK1301 **USB PIC Programmer Kit** **$34.95**

Water Sensor Alarm

This little $7 kit can really "bail you out"! Simply mount the alarm where you want to detect water level problems (sump pump)! When the water touches the contacts the alarm goes off! Sensor can even be remotely located. Runs on a standard 9V battery.

MK108 **Water Sensor Alarm Kit** **$6.95**

USB DMX Interface

Control DMX fixtures with your PC via USB! Controls up to 512 DMX channels each with 256 different levels! Uses standard XLR cables. Multiple fixtures can be simply daisy chained. Includes Light Player software for easy control. Runs on USB or 9V power.

K8062 **USB DMX Interface Controller Kit** **$67.95**

HV Plasma Generator

Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

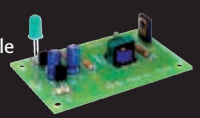
PG13 **HV Plasma Generator Kit** **$64.95**

Air Blasting Ion Generator

Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7 **Ion Generator Kit** **$64.95**

Tickle-Stick Shocker



The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 **Tickle Stick Kit** **$9.95**

Classic Nixie Tube Clocks

Our next-gen Nixie clock kits for 2013! Hand made teak-maple, polilshed aluminum, or clear plexi base with options including GPS timebase! Check our site!

NIXIE **Classic Nixie Clock Kits** ***from*** **$229.95**

Tri-Field Meter Kit

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of ghosts! Req's 4 AAA batteries.

TFM3C **Tri-Field Meter Kit** **$74.95**

Passive Aircraft Monitor

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1 **Passive Aircraft Receiver Kit** **$89.95**

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Assmb.

PR2 **Broadband RF Preamp** **$69.95**

Electret Condenser Mic

This extremely sensitive 3/8" mic has a built-in FET preamplifier! It's a great replacement mic, or a perfect answer to add a mic to your project. Powered by 3-15VDC, and we even include coupling cap and a current limiting resistor! Extremely popular!

MC1 **Mini Electret Condenser Mic Kit** **$3.95**

12VDC Regulated Supply

It gets even better than our AC121 above! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility! Dual ferrite cores!

AC121 **12VDC 1A Regulated Supply** **$9.95**

12VDC Worldwide Supply

It gets even better than our AC121 to the left! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility! Dual ferrite cores!

PS29 **12VDC 1.25A Worldwide Supply** **$19.95**

GET THE NUTS AND VOLTS DISCOUNT!

Mention or enter the coupon code ***NVRMZ12*** *and receive 10% off your order!*

800-446-2295
www.ramseykits.com

RAMSEY ELECTRONICS®
590 Fishers Station Drive
Victor, NY 14564
(800) 446-2295
(585) 924-4560

CAD CONNECTED

Proteus Version 8
Launch Offer - SAVE 10%
Use NVL2013ATJ
When placing your order

PROTEUS DESIGN SUITE VERSION 8

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

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

labcenter Electronics www.labcenter.com

Visit our website or phone 866.499.8184 for more details

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