Are You Puzzled by Microcontrollers? Then Step Into The ARDUINO CLASSROOM
Serial-to-Ethernet Solutions with SSL & SSH Security

- SB70 LC
  2-Port 3.3V TTL serial device support

- SB700 EX
  2-Port RS-232 & RS-422/485 serial device support

- CB34 EX
  2-Port RS-232 & RS-422/485 serial device support

- PK70 EX 232/485/MMS
  4-Port RS-232 and/or RS-485/422 serial device support

Access serial ports over SSL & SSH protected Ethernet!

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

**The method:**
Connect serial port to a serial to Ethernet device server

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

NetBurners Secure Serial to Ethernet Servers

Network-enable serial devices out of the box - no programming or development is required. The hardware is pre-programmed to convert your serial data to Ethernet, enabling secure communication with the serial device over a network or the Internet.

<table>
<thead>
<tr>
<th>Part No.</th>
<th>Description</th>
<th>Price (qty. 100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB70LC-100R</td>
<td></td>
<td>$49</td>
</tr>
<tr>
<td>SB700EX-100R</td>
<td></td>
<td>$149</td>
</tr>
<tr>
<td>CB34-EX-100IR</td>
<td></td>
<td>$179</td>
</tr>
<tr>
<td>PK70EX-232CR</td>
<td></td>
<td>$285</td>
</tr>
<tr>
<td>PK70EX-485CR</td>
<td></td>
<td>$288</td>
</tr>
<tr>
<td>PK70EX-MMS</td>
<td></td>
<td>$346</td>
</tr>
</tbody>
</table>

Need a custom solution?

**NetBurner Development Kits** are available to customize any aspect of operation including web pages, data filtering, or custom network applications.

For additional information please visit http://www.netburner.com/kits
Fun starts behind a 5” screen

mikromedia 5 for Tiva™

price: $249.00

ARM Cortex™-M4F on-board
Powerful Workhorse

Mighty TM4C129xNCZAD
rocks: 120 MHz operation, 1 MB of Flash, 256 KB of SRAM, μDMA controller, on-chip LCD controller and a lot more.

Size does matter
5” Touch Screen

Huge touchscreen in 800x480px resolution brings awesome graphics and vivid colors. You’ve never seen such a big display driven just by a microcontroller.

Communicate wirelessly
CC3000 WiFi module

Internet of things - check! Insanely popular CC3000 WiFi module is right here for you. It features easy to use SimpleLink IPv4 TCP/IP stack.

Beyond expectations
Debugger on board

For the first time, we’re embedding programmer and debugger on mikromedia board. Just plug in the USB cable and there you go!

GET IT NOW
http://www.mikroe.com/store/
22 Improved Efficiency 13.8V Power Supply
Many hobbyists use or repair equipment intended for automotive use, so require a power source. The linear design presented here is modernized with MOSFET pass elements that improve efficiency with no switching noise.
By Robert Atkinson

30 An Easy Two-Wire LCD
Inexpensive LCDs show up all the time on the surplus market, but they usually require either four or eight data lines in addition to several control lines. Of course, you could always dip into your wallet and spring for more advanced displays. However, with just a pittance of external parts, it is easy to reconfigure even the humblest LCD to run well on only two port lines of a microcontroller. This article describes just such a technique.
By Thomas Henry

38 An Arduino Controlled Digital FM Radio
While you can always listen to the radio from some newfangled device, why not do it the old-fashioned way — over the air from a desktop unit. (Plus, you get to control it with a remote!) By Craig A. Linderley

08 PICAXE Primer
Sharpening Your Tools of Creativity
PICAXE-Pi Communications — Part 1. Get answers to questions about the behavior of the cyon10.py program from last time, and begin an exploration of PICAXE-Pi communications.

14 Q & A
Reader Questions Answered Here
Inquiries about trickle chargers, ham mobile radio setups, and electronic player pianos are asked and answered this time.

60 Open Communication
The Latest in Networking and Wireless Technologies
DSL and Cable TV Broadband Internet Access: How Do They Work?
Chances are you’re already using at least one of these technologies, but recent upgrades have made them faster. Read about the technical details of each system and the most recent new features and specifications.

52 MakerPlot — The DIY Software Kit Part 5
An important function of MakerPlot is its bi-directional capabilities. What this means is that your microcontroller can control or be controlled with this unique software. Find out how this is accomplished.
By John Gavlik and Martin Hebel

68 Smiley’s Workshop
Programming • Hardware • Projects
The Arduino Classroom.
Arduino 101/Chapter 2: Digital Output - LEDs. As we continue our curriculum on computing and electronics basics, learn how to design circuits using light emitting diodes, and then how to use these LEDs with Arduino software to indicate events to people using your system.
The only USB Oscilloscope built for advanced users, priced for students.

$99.00
(U.S. student price)

Turn any PC into a powerful electrical engineering workstation! The USB-powered Analog Discovery lets you measure, visualize, analyze, record and control mixed signal circuits of all kinds. It’s small enough to fit in your pocket, but powerful enough to replace a stack of lab equipment. Driven by the free WaveForms™ software, the Analog Discovery lets you build and test analog and digital circuits in virtually any environment, in or out of the lab.

- 2-Channel Oscilloscope
- 2-Channel Waveform Generator
- 16-Channel Logic Analyzer
- 16-Channel Digital Pattern Generator
- ±5VDC Power Supplies
- Spectrum Analyzer
- Network Analyzer
- Voltmeter
- Digital I/O

Also Available:

Analog Parts Kit (APK)
- 20+ ICs from Analog Devices
- 200+ discrete components
- Solderless breadboard
- Perfect for Circuits 1&2 classes

$69.99

Discovery BNC Adapter
- Allows the use of standard BNC-terminated probes
- Selectable AC & DC coupling

$19.99

Now supported by MATLAB / MATLAB student edition

www.digilentinc.com/AnalogDiscovery
Night Meeting

When Google announced that it had purchased Boston Dynamics, I couldn't help but think of Ray Bradbury's tale of Night Meeting, from his Martian Chronicles. The story begins when a man from Earth and a Martian encounter each other on a desolate road one night on Mars. The man is driving an old pickup truck, while the Martian is driving a multi-legged vehicle. They look out on the landscape and realize that they come from different times, but they can't determine which of them is from the future and which is from the past.

There's a lot more to the story, of course, but the metaphor of legged and wheeled vehicles passing in the night seems relevant to the Google-Boston Dynamics deal. Of course, Google is the company behind the driverless car that promises to make the steering wheel as useful as your appendix. Then, there's Boston Dynamics, the creator of the Army Mule, Big Dog, Cheetah, and other four-legged robots that can manage rough terrain that would stop a wheeled vehicle in its tracks. If you check out the Army Mule on YouTube, you'll hear that the gas-powered engine needs a bit of muffling before it can be used in a stealth operation, but otherwise, it seems up to the task of hauling gear.

I don't see multi-legged vehicles replacing the four-wheeled car any time soon, but cars aren't the only vehicles in use today. More and more "personal" vehicles are making their way onto sidewalks, in stores, and in the malls. These motorized carts and wheel chairs often require the user to detour onto ramps because they can't navigate steps or escalators. Perhaps there's something in a multi-legged vehicle that would provide value over and above the transportation provided by an ordinary motorized buggy.

For military purposes, there's the obvious advantage of a pack mule that can carry heavy loads and, eventually, serve as a vehicle for soldiers. For the soldiers who lose one or both legs in battle, riding a weaponized robotic mule into battle might be one way to contribute to the fight. For civilian purposes, imagine the spinoffs of the legged technology — from chairs that gently raise or lower an elderly or injured person, to walking assistants that either carry or guide the person to their destination.

One thing's for certain — we're bound to see spinoffs of the technology appear at our favorite online suppliers. I can't wait to get my hands on what I can only imagine is the sensor technology used by the Mule to maintain balance. Then, there's the camera system used to track the terrain. I don't know what sort of gasoline-powered generator is used in the Mule, but I'm sure that I can think of ways to repurpose the technology for other projects.

For now, I have no desire to be transformed into a bionic Centaur, but in another 30 years or so when my joints are arthritic from all those marathons, I may have a different opinion. It's good to have options, and that's certain to come from the Google-Boston Dynamics venture. 

---

Feedback Motion Control

The Old Way
1) Build robot
2) Guess PID coefficients
3) Test
3a) Express disappointment
3b) Search Internet, modify PID values
3c) Read book, modify PID coefficients again
3d) Decide performance is good enough
3e) Realize it isn't
3f) See if anyone just sells a giant servo
3g) Express disappointment
3h) Regress PID coefficients
3i) Switch processor
3j) Dust off old Differential Equations book
3k) Remember why the book was so dusty
3l) Calculate new, wildly different PID coefficients
3m) Learn new, wildly different swear words
3n) Research fuzzy logic
3o) Have it certainly not working in uncertain ways
3p) Pull hair
3q) Switch processors
3r) Regress PID coefficients
3s) Switch programming language
3t) Start a new project that doesn't need feedback control
3u) Set the new project to use fuzzy logic
3v) Start testing every possible combination of PID coefficients
3w) Stop asking questions and go back to old project
3x) Start testing every possible combination of PID coefficients
3y) Stop asking questions and go back to old project
3z) Wait, it's working!
3aa) Decide not to do any more projects that require control systems.
3ab) Wonder why someone doesn't just make a thing that does it itself.

The Kangaroo x2 Way
1) Build robot
2) Press Autotune
3) Get a snack

Kangaroo x2 adds self-tuning feedback to SyRen and Sabertooth motor drivers.

$24.99
www.dimensionengineering.com/kangaroo

Did you know that if you're a paid subscriber to Nuts & Volts, you can get the online version for FREE? Get details at www.nutsvolts.com
Pleased With Project

Referring to the “Quick and Easy USB Keyboard Input” project by William Pippin in the November 2013 issue: This is a very useful and absolutely GREAT project idea! I put it together and it operates exactly as author-advertised. There’s immediate response for use when it’s plugged in and it works without a flaw on any pre-installed or user program that accepts a keyboard or keypad input. I use it to accept ‘touch tone’ decoded digital outputs using my own phone line project that was published in N&V in 2010 to activate a monitoring program for my home phone — even from cell phone use. Great project! Send in more useful projects like this.

John Mastromoro

“Build a PIC-Based Remote Temperature Sensor” there is a sidebar discussing saturation of a bipolar transistor. The definition in the sidebar is not correct. The correct definition of saturation is the state where an increase in base current does not cause an increase in collector current. The formula at the bottom of the sidebar yields this for the value of Rb: Rb = (Vin - VBEsat)/Ib. All this formula will yield is a value for Rb based on the desired Ib and Vin. It has nothing to do with saturation.

In order to calculate the minimum Ib required for saturation, you need to know the load voltage and the load resistance. This allows you to calculate the maximum current (ICmax = Vcc/Rload) that is

(UN)Saturated
In the January 2014 article,
At the end of the previous Primer installment, I posed three questions about the behavior of the cylion10.py program which we’re going to address this month:

- Why does one of the LEDs remain lit when you press ctrl-c to terminate the program?
- After terminating the program, why does the lit LED turn off when you type GPIO.cleanup() at IDLE’s interactive prompt?
- Finally, why didn’t we see something similar in our earlier experiments last time?

We’re also going to begin our exploration of PICAXE-Pi communications. However, before we tackle either of those topics, there are a couple of items that I want to discuss. So, let’s get started!

Unzipping Archived Files on the Pi

If you have read earlier PICAXE-Pi articles here, you know that I have been posting the individual program files on my website because I hadn’t been able to find a way to unzip the downloaded N&V zipped file. However, I recently received an email from a helpful reader (Margaret L.). The solution that she suggested is printed in the Reader Feedback section of last month’s issue of Nuts & Volts – you may want to check it out.

Now that we’re beginning our exploration of PICAXE-Pi communications, we’re going to be working with both PICAXE BASIC programs and Python programs at the same time. As a result, we’re going to need to download files to our PCs anyway, so I’ve decided to stop posting the individual program files on my website.

I think the simplest approach is to download the zipped file to a PC, unzip it, and transfer the Python programs to the Pi via a USB Flash drive, or by setting up the Pi as a server on the local network. (On the Pi, I also save all my Python programs directly on the Flash drive as a backup, in case my Pi crashes at some point.)

PICAXE Editor Beta is Now Available

The second topic that I want to mention is not in any way related to the Pi, but may be of interest to many Primer readers. Revolution Education recently announced the availability of a Beta version of the new Programming Editor software which has been renamed to “PICAXE Editor, version 6,” or just PE6. RevEd emphasizes the fact that this is a Beta release so there may be minor bugs involved, and there will certainly be many changes before the final version of PE6 is released.

PE6 is a major re-write of the PICAXE IDE (Integrated Development Environment); it includes many powerful changes – some of which may be confusing at first. Conveniently, PE6 can be installed alongside version 5 of the Programming Editor, so we can easily experiment with it but still use our older (more familiar) software for our current projects.

When the final version of PE6 is released, we’ll definitely discuss it in some detail. Until then, I just want to whet your appetite by mentioning one new feature that’s been on my wish list for a long time: include files. This new capability in PE6 means that we will be able to simplify our PICAXE software in the same manner that we are now doing with our Python programs.

For example, we’ll be able to write a PICAXE “module” that implements a fairly complex task (e.g., interfacing with an LCD display) and then simply write something like #include LCD.basic in any program that requires an LCD display (“basic” is the extension that needs to be used for an include file in PE6). If you’re interested in experimenting with the Beta version of PE6, you can download it at www.picapec.com/Software/PICAXE/PICAXE-Editor-6/.

On the same page, there’s a link for downloading the “PE6 beta release notes” – a 24 page pdf file that describes the major features of the new software.

Now, let’s turn our attention to the three questions from last time. The first two are relatively easy to answer. As you may remember, we used an infinite while loop to implement our scanning Cylon Eye. (If you need to refresh your memory, take another look at the cylion10.py program.)

Also, in the blink(LED) function that we used, we paused briefly to light each LED but we didn’t pause at...
all after turning off each LED.

As a result — for all practical purposes — one of the LEDs is always lit whenever the program is running. Whenever we press ctrl-c, the program immediately exits which means that the GPIO.cleanup() statement isn’t executed and one of the LEDs remains lit. Of course, when we manually execute the GPIO.cleanup() command at IDLE’s interactive prompt, the GPIO pins are reset to their default input state, and the lit LED is turned off.

The third question (Why didn’t we see something similar in our earlier experiments last time?) is a little trickier. In each of the earlier programs last time, we pressed a pushbutton to light an LED; when we released the button, the LED immediately turned off. So, unless you held the pushbutton down for a while, you never saw the result. (Of course, that was the purpose of the while loop.)

As a result, it’s easy to forget that the GPIO pins were using in the program remain configured as outputs.

At this point, you’re probably wondering why I’m spending so much time on what seems to be a trivial point. The reason is that I want to use our cylon10.py program to illustrate an important feature in Python. It’s called “exception handling,” and it’s so important that most Python textbooks devote an entire chapter to it.

We certainly don’t have enough space in the Primer to thoroughly discuss the techniques of exception handling, but I do want to cover the basics so that we can use this to our advantage in the Python programs we will be writing.

In Python, an exception refers to any set of circumstances that can result in a program being automatically terminated. Many exceptions are what we would call “errors.” For example, if a Python program is in the process of writing a file to a disk and the disk becomes full, an error occurs and the program will automatically terminate (a.k.a., “crash”).

On the other hand, our ctrl-c Keyboard Interrupt is also an exception; when the interrupt occurs, the program immediately terminates. However, the Keyboard Interrupt is definitely not an error; it’s a Python feature that allows us to exit an infinite while loop.

In either case (an error or a feature), the point is that we want to be able to handle the exception gracefully. For example, we could allow the user to switch to another disk to save the file; or — in the case of the Keyboard Interrupt — we could terminate the program “gracefully” by executing a GPIO.cleanup() statement before exiting the program.

Handling Exceptions in Python

The two main statements that Python uses to handle exceptions are try and except. Let’s examine a simple code snippet that uses these two statements to execute a GPIO.cleanup() statement before exiting our cylon10.py program when ctrl-c is pressed:

```python
while 1:
    try:
        for LED in LEDs[0:9]:
            blink(LED)
        for LED in LEDs[9:0:-1]:
            blink(LED)
    except:
        GPIO.cleanup()
        print("GPIO pins reset.")
        print("Program terminated.")
        break
```

The code in the try block above is the same as it was in our original infinite while loop. As long as we don’t press ctrl-c, that’s the only code that gets executed. In other words, the infinite while loop executes exactly as it did last time. However, as soon as we press ctrl-c, an exception occurs and the code in the except block is executed.

As a result, the GPIO pins are reset, we tell the user what happened, and the break statement is executed. I don’t think we’ve used a break statement before, but its function is simple: It “breaks out” of the infinite while loop. Since there is no other code in our program, it terminates.

Because an exception occurred, you might think that the break statement isn’t necessary, but it is. When we include try and except blocks in a program, Python no longer automatically exits the program when an exception occurs; it “assumes” we will handle the exception, including exiting the program if that’s what we want to do.

As a result, if we accidentally omit the break statement, we create another error! The program does not exit, and the while loop executes again. However, we just reset the GPIO pins, so when Python tries to blink another LED, the program automatically exits with an error message stating that the LED has not been defined as an output.

Finally, there’s a potential problem with the way I wrote the above code snippet. The except block will be executed whenever any exception occurs — not just the Keyboard Interrupt exception. Python includes dozens of different types of built-in exceptions, and many external modules include additional exceptions, as well.

In addition, we can even write our own exceptions if we want to. If any one of those exceptions occurs, the program will terminate unexpectedly, providing no clue as to what happened. The solution to this problem is to write the except statement so that it only applies to the exception that we want to “catch.” (In Python jargon, the except statement “catches” the exception.)

In the above snippet, if we replace except: with except KeyboardInterrupt: we will only catch the interrupt; all other possible exceptions will cause the program to terminate normally with a (hopefully) helpful error message.
Before reading further, you may want to experiment with adding `try` and `except` blocks to the `cyon10.py` program until you feel comfortable with the technique. When you’re ready, we’ll turn our attention to serial communications where we will encounter additional uses for `try` and `except` blocks.

**Communication Options With the Raspberry Pi**

The Pi’s internal hardware implements three major communication protocols: I2C, SPI, and serial. The question is: Which of these protocols do we want to use to communicate with our PICAXE processors? The answer, of course, is it depends!

Primarily, it depends on which PICAXE processor we’re using in any given project. For example, if a fairly complex project requires the advanced capabilities of the PICAXE-20X2 processor, the I2C protocol is the obvious choice for implementing PICAXE-Pi communication. The reason is when using I2C communications, the Pi must be the master processor, and the X1 and X2 processors are the only PICAXE chips that can be configured as an I2C slave. Therefore, I2C communication with a Pi is not an option with PICAXE M2-class processors.

In my experience, however, the majority of PICAXE projects do not require the power (or expense) of a 20X2 processor. As a result, our Primer articles are going to focus on communication strategies that can be implemented with any PICAXE processor (M2, X2, or X1). If you’re interested in I2C communications between the Pi and a PICAXE X2- or X1-class processor, you may want to take a look at www.instructables.com/id/PICAXE-Raspberry-Pi-ADC/.

When using its built-in hardware for SPI communications, the Pi is also limited to functioning as a master processor; again, only the PICAXE X2- and X1-class processors are capable of implementing hardware-based SPI communications.

In addition, those PICAXE processors are also limited to master mode only for hardware-based SPI communications. (See the PICAXE documentation for `hspi_setup`, `hspin`, and `hspiout` in Section 2 of the manual.)

PICAXE BASIC also includes the `shiftin` (a.k.a., `spin`) and `shiftout` (a.k.a., `spion`) commands which implement software “bit-bang” versions of the hardware SPI commands mentioned above, but the software-based commands are also limited to the X2- and X1-class processors.

However, on M2-class processors, it’s also possible to write our own bit-bang code to implement SPI functions. The documentation for the PICAXE `shiftin` and `shiftout` commands include sample code for implementing a SPI interface on any M2 processor, but the sample code also configures the PICAXE as the master processor so we still can’t use that code to communicate with the Pi because it also insists on being the master processor!

One possible solution to this dilemma would be to write our own bit-bang routines that configure an M2-class processor as an SPI slave so we could communicate with the Pi as the master processor. I do intend to try that approach in the future, but for our first experiments I think that a serial communications link between the PICAXE and Pi will be easier to establish, so let’s get started!

**Configuring the Pi Serial Port**

By default, the Pi’s serial port is configured so that it can be used for debugging purposes. In other words, when you first boot the Pi, the long list of output that you see on the monitor is transmitted via the serial port. Since we want to configure the serial port so that we can use GPIO pins 14 and 15 to communicate with our PICAXE projects, the first thing we need to do is to disable the serial login connection. In order to do so, there are two files on the Pi that we need to edit: `/etc/init.d/serial` and `sudo nano /boot/cmdline.txt` — we’ll use the `nano` editor for this purpose.

To edit the first file, open the terminal and type the following:

```
sudo nano /etc/init.d/serial
```

Press return and move to the end of the file. You should see something similar to:

```
serial:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Comment out that line by adding a `#` character in front of it; then save the edited file.

To edit the second file, type this in the terminal:

```
sudo nano /boot/cmdline.txt
```

Press return — the file will contain the following (all on one line):

```
dwc_otg.lpm_enable=0
console=ttyAMA0,115200
console=ttyS0,115200
```

Edit the file by removing both references to the serial port (ttyAMA0). When you have finished, the file should contain the following (all on one line):

```
dwc_otg.lpm_enable=0
console=tty1
```

Save the edited file, and reboot the Pi by typing this in the terminal (and then pressing return):

```
sudo nano shutdown -r now
```
When your Pi has finished rebooting, the serial port (ttyAMA0) will now be available on GPIO pins 14 (TxD) and 15 (RxD), but we still need to install the pySerial package before we can begin exploring the serial communication between the PICAXE and Pi. Before installing any new packages on the Pi, it’s a good idea to first execute these two commands in the terminal:

```
sudo apt-get update
sudo apt-get upgrade
```

The upgrade process may take quite a while to complete. When it’s finished, you need to execute one of the following commands in the terminal:

For **Python3**:
```
sudo apt-get install python3-serial
```

For **Python2**:
```
sudo apt-get install python-serial
```

At this point, we’re finally ready to begin our exploration of PICAXE-Pi serial communication.

### Experiment 1: Serial Output from PICAXE to Pi

In our first experiment, the Pi will receive serial data from a PICAXE-08M2 processor. (Actually, you can use any X2-, X1-, or M2-class processor — I just wanted to be sure the 08M2 is able to do the job.) The hardware setup is simple: Pin C.2 of the 08M2 is connected to GPIO 14 (TxD) on the Pi, and pin C.1 of the 08M2 is connected to GPIO 15 (RxD) on the Pi. My hardware setup is shown in **Figure 1**.

![FIGURE 1. Breadboard setup for Experiments 1 and 2.](image)

As you can see, I’m using the stripboard interface circuit that we constructed in an earlier Primer. However, you can use any hardware setup you prefer — just make sure that the 08M2 is powered at 3.3V to match the levels on the Pi’s GPIO pins, and that there is a series resistor in the two I/O connections that we are using. (The stripboard interface circuit includes 470 Ω resistors on each GPIO line.)

Of course, we need two programs for this experiment: one for the Pi and one for the 08M2. As usual, go to the article link and download the zip file that contains all the necessary programs for this month’s Primer.

One approach is to unzip the file directly on your Pi if you have been able to get that working. If not, you can download the file to your PC, unzip it there, and transfer the Python files to your Pi either by using a USB thumb drive or by setting up the Pi as a server on your network. One way or another, you need to have the Python files on the Pi, and the PICAXE BASIC files on your PC.

For this experiment, the 08M2 program is **seroutToPi.bas** and the Pi program is **serinFromAx.py**. Let’s start with the 08M2 program which simply sends the ASCII values for characters A through Z to the Pi in an infinite do/loop. The serial transmission is “True” (i.e., the line idles high) because that’s the format required by the Pi. We don’t need to discuss the details of the **seroutToPi.bas** program; when you read through the listing, you will see how simple it is.

The Pi program (**serinFromAx.py**) is even shorter than the 08M2 program, but we do need to discuss a few details. In order to facilitate our discussion, the complete program follows:

```
A) # serialInFromAx.py
   (for Python3)
B) import serial
C) # define & open
   serial port
   @ 9600 baud
D) ser = serial.Serial
   ('/dev/ttyAMA0', 9600)
E) while True:
F)   try:
G)     print(ser.read())
H)   except KeyboardInterrupt:
I)     print('Program
          Terminated.')
J)   ser.close()
K)   break
```

In the above code, the letters along the left edge are not part of the program; they are just there to facilitate our discussion. The following comments refer to the corresponding program lines above:

**A** As the comment in this line indicates, the program is written for **Python3** which makes it a little more complicated than it would be if it were written for Python2. We will discuss the main differences shortly.

**B** This statement imports the **pySerial** package that we installed earlier. If you want to read more about this package, just search for "pySerial documentation" (without...
the quotes). Also, note that we did not import RPLGPIO; all we need is the pySerial package.

D) This is one way of declaring and opening a serial port in the pySerial package. Any name can be assigned to the port; I chose to use ser for brevity. In Python, all I/O devices (including serial ports) are treated the same as files, so the /dev/ttyAMA0 parameter indicates that the port we are opening (ttyAMA0) is located in the dev folder, which is where all device files are located in Linux. The second parameter (9600) indicates that we’re opening the serial port at 9600 baud.

Two additional points are worth mentioning:

- Whenever a serial port is defined, it’s also automatically opened; a separate statement isn’t needed to do that.
- All serial communication in Python is “true” (i.e., the line idles high), and it defaults to eight data bits, no parity, and one stop bit. These defaults work fine for communications with PICAXE processors, but if you’re interested in changing any of them refer to the pySerial documentation.

F) The basic structure of the try and except blocks follows the same pattern we discussed earlier.

G) The read() function in the pySerial package reads a single byte. Also, by default, the read() function has no timeout. In other words, it’s a blocking function — the program will wait at this point until a byte is received. This behavior can be modified by declaring a specific timeout when we initially configure and open the serial port. At some point, we will probably need to do that.

H) In the except block, we need to close the serial port before terminating the program. Leaving a serial port open when we terminate a program can cause problems for another program that attempts to open the same port.

When you’ve completed your hardware setup for this experiment (Figure 1), run the O8M2 program first and then run the Pi program. Figure 2 is a screenshot of both programs running on my Mac. I’m using SSH to connect to my Pi, so the contents of its GUI appear on my Mac’s desktop. If you have a monitor and keyboard attached to your Pi, you should see essentially the same results on two different monitors.

As you can see, the Pi’s output is a little strange. For example, instead of displaying A, ‘A’, or 65, the Pi displays b’A’! That’s Python3’s way of saying that it’s displaying a byte, which is one of several data types supported in Python3.

It took me a while to figure out how to convert the output to something more “normal.” If you want to display the received bytes as their ASCII values, try replacing line G in the program with
print(ord(ser.read(1)))
and run the Pi program again. (You can just leave the O8M2 program running as you experiment with this.) You should see the appropriate sequence of numbers (65 through 90) on the screen.

On the other hand, if you would prefer to see the actual ASCII characters on your screen, replace line G with
print(chr(ord(ser.read(1))))
and run the program again. There may well be a way to accomplish the same thing with fewer parentheses, but I haven’t found it yet. If you find one, please let me know!

Experiment 2: Serial Output from Pi to PICAXE

For our second experiment, we’re going to reverse the direction of the serial link. This time, the Pi will send serial strings to the O8M2 processor using the same hardware setup we did in Experiment 1 (again refer to Figure 1).

As you probably know, PICAXE processors are not able to receive serial strings directly; it’s necessary to store each received byte of the string in a separate variable. As a result, the PICAXE file for this experiment (serinFromPi) has is more complicated than the Pi’s Python file. However, we aren’t going to discuss it in detail because it’s almost identical to the approach we took back in the
October 2011 Primer. (See the discussion of the SerxldFast14M2.bas program in that article.)

Before we move on to discussing the Pi’s Python program for this experiment, I do want to point out a couple of differences between the 2011 PICAXE program and the one that we’re using in this experiment.

First, back in 2011 there was a bug in the PICAXE compiler that has since been resolved, so we no longer need to include the workaround that we used back then. Also, in the earlier article, we were using the srrxld command; this time, we’re using the serin command but the technique involved is essentially the same. In addition, back then we used a 14M2 processor, and this time we’re using an 08M2 processor. As I mentioned earlier, any current PICAXE processor can be used for our Pi experiments, so that difference isn’t significant either.

When you have downloaded the serinFromPi.bas program, read through it to make sure you understand how it functions. If anything isn’t clear, you may want to re-read the 2011 article I just mentioned.

When you’re ready, let’s move on to the discussion of the Pi program for this experiment (serialOutToAx.py). In order to facilitate our discussion, the complete program follows:

```python
sleep(1)
serv.write(b’Where are you?’)
sleep(2)
except KeyboardInterrupt:
serv.write(b’Keyboard Interrupt’)
serv.close()
break

There are two points in the above program that I want to clarify. First, in previous programs, we have written import time which imports all the functions included in the time library. The alternate version that we’re using in this program has two advantages: only the sleep function is imported (which is all we currently need), so this version of the import command reduces the size of our program; and we can use a sleep statement without needing to write time.sleep().

In this program, we’re sending three different strings to the 08M2. In Python2, that was a simple matter (e.g., serv.write(‘Hello PICAXE!’)). However, in Python3 there’s a complication similar to what we discussed in our first experiment. We need to send the string as a series of bytes. Including a lowercase b in front of each string accomplishes that goal.

There are more complicated ways of doing the same thing, but the above approach seems to work well. Figure 3 is a screenshot of both programs running on my Mac; your results should be similar.

Once again, we’re out of space this month! Next time, we’ll continue our serial communication experiments and use an 08M2 processor to help the Pi accomplish something that it can’t do on its own. As you know, the Pi doesn’t have any analog inputs but PICAXE processors have plenty of them (three on the 08M2, seven on the 14M2, and 11 on the 20M2).

As a demonstration of how PICAXE can help the Pi, we’ll interface an MCP9700A analog temperature sensor with an 08M2 processor and serially send the temperature reading on to the Pi for display. As you may remember, we have already accomplished the PICAXE portion of this task (the December 2012 Primer), so you may want to re-read that article to get a jump on things. In fact, you might even try to implement your own working system before the next installment.

Whatever you do, have fun! NV

![FIGURE 3. Screenshot of output for Experiment 2.](image)
Trickle Charger

Could you provide a circuit to make a 12 VDC trickle charger for my home generator? The battery is gel type. I want to build it using a 12.5 VAC x 4.5 amp transformer. Thank you for your past help.

— Ken Bartone

A 12 amp-hr battery should tolerate a constant trickle current of 100 mA, but battery manufacturers recommend float charge for long term storage. The difference is that a float charger limits the voltage to 13.8 volts for lead-acid batteries of six cells.

A bridge rectifier will give a peak voltage of about 17.5 volts; with a full wave bridge rectifier, the DC voltage will be about 16 volts peak. Referring to Figure 1, if the battery is to be float charged at 13.8 volts, the voltage at the Q1 collector will be 14.3V which means the zener (D2) must be 13.8V. R1 shunts any zener leakage to ground to prevent Q1 from turning on prematurely. R2 limits the peak current to less than one amp: (16-14.3)/4 = 0.425 amp and the average current — according to my handbook — will be 0.135 amps.

When the battery is fully charged, all the current flows through Q1 and its power dissipation is: \(14.3\times0.135 = 1.9\) watts. Q1 is rated at two watts with no heatsink, but for reliability I would use a heatsink of 30 deg C/watt at least.

The power in R2 is: \(P = \frac{1}{2}IR^2 = 0.07\) watts, so a 1/4 watt resistor will be fine. A bridge rectifier rated at one amp is available in a four-pin DIP package to complete the design.

D3 is necessary to protect Q1 in case the generator charging circuit tries to charge the battery to more than 13.8 volts.

**FIGURE 1.**

---

Ham Mobile Radio Setup

I had to move and disconnect all my radios and antennas from my car. I am now in the process of rebuilding the VHF and HF power system. I want to make sure that when the car is shut off, the radios will shut off within 10 minutes to save my car battery. Do you know of a schematic that will do the job?

— Jim Houser WA8JIM

A I know the problem, but 10 minutes is a long time for a 555 timer. A PIC12F675 will do the job with fewer parts. In Figure 2, the 12 volts from the ignition switch going down starts the time. Five volts for the PIC will be supplied by a programmable zener diode because its current requirement is low. I don't know the current drain of the radios, but 30 amps should be adequate.

What is needed is a 30 amp 60 volt switch that can be controlled with a five volt signal. That in itself is pretty simple, but the design is complicated by the need to protect the circuit from the 60+ transient volts that are expected in an automotive system.
In Figure 2, the components R2, C1, R4, C2, R10, R12, and D1 are for transient protection. D1 is necessary because pin 4 does not have a diode to VDD; only one to VSS. R4 and R5 provide a safe +5V to pin 2 (GPIO.5) when the ignition is on.

When the ignition is turned off, IC1 is reset immediately through C5 and R12, and the program starts looking for a low in pin 2 — which happens after a short delay caused by C2. Then, the 10 minute delay starts, at the end of which pin 7 (GPIO.0) goes high permanently turning Q2 off. When the ignition is turned on, IC1 is reset through R10, C4, and Q3 which starts the program running again, looking for the ignition to be turned off.

Figure 3 shows the power-off delay program. A file is available at the article link. (If you want to build this, I can supply a programmed PIC12F675 for $5.)

**Electronic Player Piano**

I have an old player piano mechanism that I want to convert to electronic. When a note is to be played, a hole in a paper roll passes over the corresponding “hole” in a metal bar. I have thought about light detectors or even pressure detectors to detect the paper hole but I need something simple, small, and reasonably cheap. I was also thinking about using either a hacked electronic keyboard or a microprocessor to generate the music. Any ideas to get me going?

— Phil Fitzjarrell

I have a player piano, so I know something about how it works. Your project is very interesting to me. One solution is a small rotating wheel (88 of them) that falls through the hole in the paper and completes a circuit. This is electrically simple but mechanically complex. Not being mechanically inclined, I will opt for a photosensor solution. The piano paper roll is normally white but some of mine have yellowed with age. The paper is slightly more than 11 inches wide and has up to 88 slots (0.50 inches wide) which correspond to the 88 keys, plus another slot which I think is for automatic pedal operation. I measured the holes in the metal bar (see Figure 4) which turned out to be .075
MAILBAG

Re: Bandpass Filter, page 20, December 2013:

#1 I was looking over your calculations for the filter in the December issue, and it struck me that there was a resistor missing from the design we use in class. If you add R3, the calculations go as indicated in Figure A. I ran the bode plotter in Multisim, and got results very close to your design goals. I haven’t built the circuit in the lab yet, but maybe soon. I really enjoy reading your solutions and explanations. Please keep up the good work.

Ron Tincham, Professor
Biomedical Engineering Technology
Santa Fe College

Thanks for your interest; this circuit (Figure A) is the one I usually use for an RC bandpass. I was intrigued by the circuit sent by Bob Woycz because I had not seen it before. The calculations got so complicated that I made C1 = C2 to simplify, but forgot there were now only two variables. So, gain, frequency, and bandwidth could not all be specified. That is why my results were off. I leave the calculations to interested parties!

#2 I enjoyed working through your equations regarding the bandpass filter problem. The circuit is called an Infinite-Gain Multiple Feedback Bandpass Filter; Figure A is actually taken from Rapid Practical Designs of Active Filters, D. Johnson & J. Hilburn, Wiley 1975, page 139. The equations from the book are:

\[ \text{Gain} = -2\times R_1/R_2 \]

\[ B = 2\times R_2/C \]

\[ \omega_0^2 = (1/R_3C^2)(1/R_1 + 1/R_3) \]

Referring to Figure A, B is the bandwidth about the center frequency \( f_0 \), being equal to \( f_2 - f_1 \).

\[ Q = \omega_0/B \]

\( C = C_1 = C_2 \). R3 is missing from the problem circuit, so the term 1/R3 disappears from the equation for \( \omega_0^2 \).

Using your numbers:

\( R_1 = 1.589 \text{ ohms} \)

\( R_2 = 317,844 \text{ ohms} \)

\( C = 10^8 \text{ farads} \).

Then

\[ G = 100.01 \text{ (40 dB voltage gain)} \]

\[ B = 1.001 \text{ kHz} \]

\[ \omega_0 = 44,500 \text{ radians/second} = 7.082 \text{ kHz} \]

\[ Q = \omega_0/B = 7.07 \]

Peter A. Goodwin

Thanks for the feedback. The equations that you supplied made me realize that by making C1 = C2, I had lost control of one of the variables and that is why my results were not as expected. It is necessary to keep the ratio C1/C2, but that is too complicated to bother with.

The three-resistor circuit is tunable via R3 without affecting gain or bandwidth. The two-resistor circuit is not tunable.

Re: More on the Jacob’s Ladder, page 20, November 2013:

Looking at the article on Jacob’s ladder, I’m confused on coils/HV being directly connected to the Q1 drain. Shouldn’t it go to output? I want to use this as a fence zapper, but it looks like max smoke to me. So, does the July issue seem right?

Fred

Figure 1 in the November issue was intended to be like the figures in the July issue; the HV at the top of the ignition coil near terminal 4 is where the arc forms. I want to caution you that if used as a fence controller, it could be lethal depending on the frequency. Fence controllers usually produce a pulse every second or so. If you reduce the frequency of the oscillator, it could be a real attention getter but not lethal. Ideally, the secondary should be isolated from the primary, but ignition coils are not built that way. The spark current coming back through ground could overstress C3. To avoid that, connect Q1 drain to earth ground and operate the circuit from a battery.

Re: NiCad Battery Chargers, page 22, December 2013:

Just received the December 2013 issue of Nuts & Volts. Had to comment on Don’s question on page 22, Figure 4. Don specified a battery charger rated 2.9/8.7 VDC at 105/12 mA with humming. No burnt resistors and LEDs light up. I redrew half of the circuit; the 24 ohm resistor is the current limiter.

The LED turns on with sufficient current across the 24 ohm resistor and the (120/260 or 120/350 ohm) resistor and diode divider network; 110 volts applied across the 24 ohm resistor at 50 percent duty cycle from D5 or D6 would require about 126 watts.

Battery connections are not shown. As drawn in the diagram, D1 through D4 are shorted by jumpers (brown, black, green, gold, and silver). If silver is replaced by two cells in series or green is replaced by a 7.2 volt battery, numbers work out.

Source is 8.7 VAC for two-cell and one diode with a 24 ohm resistor gives (8.7-2.4)/24 = 220 mA => at 50% duty cycle = 110 mA.

The NiCad nine volt battery gives 7.2 volts. (8.7-7.2)/24 = 20.8 => 20.8/2 = 10.4 mA which is close to the original specification.

The only element that could hum in this circuit is the transformer at the bottom of the diagram. D1 and D3 appear to be reverse-charge protection for cells.

My conclusion: The batteries were inserted in reverse and will not charge.

Edward Wade

Thanks, Ed, for your thoughtful analysis of the battery charger circuit.
inches; I calculated the hole-to-hole spacing to be .075. That is plenty of room for an 0805 IR emitter and sensor.

Mouser has an emitter and sensor in one package that is .067 wide (part number 852-GP2S608; 42 cents each per hundred). This is a reflective system; the IR bounces off the paper to a phototransistor. The optimum spacing is .020, so a sheet of .020 plastic that is transparent to IR will be optimum, and will protect the paper from the sensors. The phototransistor will be turned on when the paper is present and off when there is a hole. That is probably the opposite of what is wanted, but an inverter will solve that problem.

Another solution is an IR emitter on one side of the paper and a phototransistor on the other. Mouser has these in the 0805 package also (part number 638-PT1721BL41TR8 for the phototransistor and 638-IR1721CTR8 for the matching IR emitter). The paper is somewhat translucent, so some intensity adjustment may be needed. A variable speed stepper motor to turn the rolls would be the easiest, but an analog speed servo would also work.

The metal bar has holes at the edges of the paper to operate a servo that keeps the paper in the center; you will want to implement that. I would do an analog servo; I don’t know how to do a digital servo. I suppose a bang-bang servo (where the paper is moved .025 or less) would work. If the left sensor is covered, the paper moves right until the right sensor is covered, then it moves back. I think the servo in my piano is two speed because there are two holes on each side of the paper, slightly offset (see Figure 5). I don’t see the paper moving back and forth; it stays in the middle.

Your idea of hacking a keyboard is great; it opens up all kinds of innovative possibilities.}
NEW PRODUCTS

AUTORANGING FAULT LOCATOR

EDS — maker of the CapAnalyzer 88A — announces the new LeakSeeker 89 Autoranging Fault Locator model EDS-89, which replaces the original LeakSeeker 82B short and leaky fault locator manufactured from 1995 to 2012.

The American-made LeakSeeker 89 locates the exact spot on a printed circuit board (to within a fraction of an inch) where a shorted or leaky component is bringing a power supply bus or data line to ground. It is able to locate defects from zero to 300 ohms with no loss of resolution. It can even find active shorts that a DVM won’t show. The high GAIN mode can locate shorted components on multi-layer boards with ground planes and a power layer. Three fully automatic range settings allow easy finding of faults along thin, normal, or wide/ground plane copper PCB runs. A video at www.youtube.com/watch?feature=player_embedded&v=BUyEe8G50D4 is available. A 60 day satisfaction or money-back guarantee is included. A copy of the operation manual is available.

SOLAR PANEL CONTROLLER

The SCLED-12V5A-T Solar Panel Controller from J2 LED Lighting uses microcontroller circuitry with user selectable nighttime on functions for LED lighting. The dusk to dawn function works by sensing the voltage of the solar panel. The controller is in dawn mode when the ambient light load on the solar panel produces a voltage over 4.5 volts (ref.). In this mode, the controller’s output is off.

The dusk mode is activated when the panel voltage drops to below 4.5 volts (ref.) due to a lack of ambient light. In this mode, the controller turns the output on to the LED lighting load. A switching delay reduces false triggering with fluctuations in cloud cover. An internal user selectable DIP switch provides the option to limit the amount of time the light is on to two, four, six, or eight hours. The controller also provides standard dusk to dawn control with no time limiting. The time limiting function is useful for various applications in which battery size and/or panel size may need to be reduced by limiting operating time of the LED lighting load at night. The controller also functions to protect the battery (SLA, Sealed Lead Acid) from over-charge and under-charge.

The controller has internal voltage sensing that monitors the battery voltage. The controller will disconnect the LED output load if the battery voltage falls below 11.6
volts (ref.). The controller will allow operation again when the battery voltage recovers to 12.5 volts (ref.). The charge power to the battery is a PWM (Pulse Width Modulation).

An internal TVS (Transient Voltage Suppressor) protects the unit from voltage transients that may occur from nearby lightning strikes to the solar panel. An internal auto resetting polymer fuse on the battery connection provides protection from current and thermal overload.

The solar panel and LED lighting load connections of the controller are recommended to be fused at five amps maximum. Any additional fusing guidelines from the solar panel and battery manufacturer should be followed. A solar panel for a 12 volt system should be used. A 12 volt solar panel rated at 60 watts is the maximum size for the SCLED-12V5A-T controller. The largest size of battery recommended is 40 amp-hours. Smaller amp-hour batteries may be used if sized accordingly to the solar panel and load.

The SCLED12V5A-T is minimally “weather resistant” and needs to be mounted so as to be protected from direct exposure to water/moisture. To ensure an extended life of the controller, mount it in a semi-covered place out of the elements. A standard NEMA electrical enclosure rated for the operating environment may be used for protection of the controller.

Electrical characteristics include:
- Module rating is 60 watts max at 12 volts DC, five amps max. Max idle current in off state is 7.0 mA max.

Electrical specifications are as follows:
- Working Temperature: -20°C-60°C.
- Dimension: L72 x W38 x H28 mm.
- Rated voltage: 12V DC.
- Rated charging current: five amp max.
- Float charge/rapid charge: 13.6-14.6 volts DC (PWM).
- Over-discharge low voltage disconnect: 11.6 volts.
- Over-discharge recover voltage: 12.5 volts.
- Ambient Light Sense (ALS) dusk to dawn switching at 4.0-4.5 panel volts (ref. 10 lux panel light), >10 seconds switching delay.
- Intended to control LED lighting loads only and for use with SLA battery only. Maximum recommended battery size is 40 amp-hours; maximum 12 volt solar panel size is 60 watts.
- Wiring polarity is defined by case marking; use no smaller than 18 AWG wire UL1007 or equivalent; fuse with no larger rating than five amps.

Pricing is as follows (in USD): 1-4 units $22.50 ea; 5-99 units $19.99 ea; and 100-250 units $16.99 ea

For more information, contact:
J2 LED Lighting
http://j2ledlighting.com

**CLASS D BLUETOOTH AMPLIFIER**

Parts Express announces their new, highly capable Lepai audio product. The compact and efficient Lepai LP7498E stereo amplifier delivers a pristine 100 watts into eight ohms per channel, thanks to its Class D output circuit design.

Features include:
- Ample, clean Class D amp output for music and multimedia.
- Advanced STMicroelectronics TDA7498E audio amplifier chip.
- Wireless music streaming via Bluetooth connection.
- Extreme efficiency (90%) delivers high headroom, reduced current draw.
- Compact size: 4-1/2" W x 1-1/4" H x 7" D.
- Power supply included.

Because of its useful power-to-size ratio, this stereo amplifier is perfect for desktop systems, bookshelf speakers, and most home (or office) audio applications. The black chassis and laser engraved silver faceplate provide a modern cosmetic touch.

The LP7498E employs the highly efficient STMicroelectronics TDA7498E audio amplifier chip, which is capable of 160 + 160W output into four ohm loads, or 1 x 220 watts bridged mono. Internal components are sourced from brands such as NEC and Alps. Integrated thermal protection ensures long life and trouble-free operation.

High quality binding posts (5/8" on center) accept bare wire, 3/8" spade terminals, or standard banana plugs for problem-free speaker connections. RCA L/R inputs are incorporated for analog connection (cable not included). Each LP7498E is shipped with its own 36 VDC 4.5A power supply.

The LP7498E also enables Bluetooth connectivity to wirelessly stream audio from an iPhone, Android smartphone, tablet, or laptop. The Lepai LP7498E amplifier's low price and accurate neutral sound character make it an ideal partner for workbenches and test stations.

For a limited time, special pricing...
of US$109 is available.

For more information, contact:
Parts Express
www.parts-express.com

UNIQUE MODULAR PROJECT SYSTEM

Take what you need, leave out what you don’t! This is the slogan for the new Tibbo Project System (TPS) featuring Tibbit™ I/O modules available new from Tibbo Technology. Tibbits (as in “Tibbo Bits”) are blocks of prepackaged I/O functionality housed in color-coded plastic shells. Want an ADC? There is a Tibbit for this. A 5V power supply? Got that. RS232/422/485 port? PoE? PWM? These, and many other Tibbits are available, as well.

Tibbits are divided into Tibbit modules and Tibbit connectors.

Each TPP (Tibbo Project PCB) can accommodate multiple Tibbit modules and connectors. Only bare essentials are provided on each board. There is a CPU, an Ethernet port, and a very simple power supply.

The rest of the board’s functionality is defined by what Tibbits are plugged in. There are several TPP “sizes” that differ in the number of Tibbits they can accommodate.

As with most Tibbo products, TPPs are programmable in a language called Tibbo BASIC. This easy to learn programming language is particularly suited for control, automation, and networking applications. Tibbo BASIC is complemented by a rich set of programming objects. There are objects for socket (TCP, UDP, HTTP) and serial communications, Wi-Fi, GPRS, file data storage, LCD and keypad control, and many other functions.

Tibbo BASIC applications are created using the free Tibbo IDE (Integrated Development Environment) software (TIDE). This software features a built-in debugger allowing users to upload their Tibbo BASIC application onto the TPP board, and cross-debug it through the Ethernet LAN without the aid of any special debugging hardware (such as a JTAG board or an ICE machine).

More than just an enclosure for TPP and Tibbits, a Tibbo Project Box adds an aesthetic touch to automation projects.

The top and bottom walls of an assembled project box are formed by two rows of connector Tibbits installed on a TPP.

The front cover of the box is made of translucent plastic that allows users to see the status LEDs of Tibbit modules installed inside. The front panel also accommodates paper inserts similar to those found on office telephones.

The inserts are meant for marking wires and ports of a TPP-based automation device.

To aid users in the evaluation and creation of TPP-based automation products, Tibbo has designed an online configurator. Name the project, select the TPP board, place required Tibbits, and the configurator will notify builders of potential problems and even calculate the projected power consumption.

Pricing for barebone systems range from US$89–$106; Tibbits range from US$2–$100. All parts can be purchased separately. Check out the website for specific costs.

For more information, contact:
Tibbo Technology
http://tibbo.com

LOW POWER RADIO TRANSCIEVER

Saelig Company, Inc., has introduced the STD-502-R 2.4 GHz transceiver which uses direct sequence spread spectrum (DSSS) modulation and a true diversity circuit, enabling reliable communications even in the congested 2.4 GHz band.


Low power consumption and battery operation give the STD-502-R the performance demanded for in applications where long range and reliability are required.

The transceiver uses a transparent data interface to enable users to communicate using their own protocols, with 77 available channels at a line-of-site range of up to 300 m.

Designed to be embedded in equipment, the STD-502-R was
developed as a radio module for industrial applications that require reliable operation. Besides using highly noise-resistant DSSS modulation, the module has a true diversity receiver function for preventing signal dropout due to multipath fading.

The modules can be easily set via the RS-232 UART interface using dedicated commands. Spread-spectrum correlation is done in a custom onboard ASIC instead of relying on conventional RFICs.

The STD-502-R module itself has no built-in communication protocol. Instead, it has a transparent RS-232 data input/output interface, enabling users to employ their own protocols without modification.

In addition, it can continuously transmit a low or high signal without restriction.

An evaluation board (the TBS-STD502) allows testing of the STD-502-R module to perform range, data packet, and switching signal transmission tests.

Onboard LEDs, CH switch, ID switch, and pushbuttons can exercise switching signal tests and packet tests with packet error indication.

The STD-502-R is designed for an operating temperature range of -20 to +65°C with an RF power of 10 mW (3.3V 65 mA), and is intended for the remote control of industrial equipment or for industrial measurement systems.

Retail price is around US$105.

For more information, contact:
Saelig
www.saelig.com
Many radio enthusiasts and other hobbyists use or repair equipment intended for automotive use. This requires a source of power, nominally 13.8V DC. Many designs have been published using the ubiquitous 2N3055 as the linear pass element. These can be inefficient. More recent are switching regulator designs. Unfortunately while efficient, these can generate unacceptable levels of electrical noise. Here is a linear design modernized with MOSFET pass elements that improve efficiency with no switching noise.
The Problem

With older bipolar (BJT) designs, you have the low current gain and large voltage drop of the pass transistor (read the sidebar Overhead in Bipolar). This requires higher unregulated voltage and consequently a larger transformer and heatsink.

In contrast to the BJT, Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) require no steady state drive current and have a low "on" resistance. The downside is that they require about seven volts on the gate to pass high currents. As this is with respect to the output voltage, the gate will be at around 21V. Using the high current unregulated supply to provide this would be worse than the BJT design. An additional transformer and rectifier could be used, but adds to the cost.

This design uses a voltage doubler to provide the supply for the regulator and gate drive. It's rated at 14A continuous or 20A peak duty but can be scaled up or down as required. It's equivalent to a commercial "20 amp" supply costing around $100.

Circuit Description

Refer to Figure 1. The incoming mains is fused and switched before being applied to the primary of T1—a 300VA toroidal transformer with two 15V secondaries connected in parallel. The AC is full-wave rectified by bridge rectifier BR1. The rectified output is smoothed by C1 and C2, which are 10,000 µF electrolytics. The voltage at this point is approximately 19V under load.

The higher voltage required by the regulator is provided by a voltage doubler comprising C3, D2, D3, and C4. This does not look like a voltage doubler unless the main rectifier and filter capacitor are considered. Note that one side of C3 is connected to the AC output of the transformer. On the negative cycle of the AC, C3 charges to the peak of the AC via D1 and the upper right-hand diode of BR1.

On the positive half cycle, the charge on C3 is transferred to C4 via D2. As C1/2 and C4 are each charged to around 20V and are in series, the voltage at the junction of D2 and C4 is 40V. As this exceeds the LM723 rating, a simple shunt regulator — R1 and zener diode D4—provides 36V to LM723. This also improves the line regulation.

The LM723 is an older IC that is simple to use and low cost. Internally, it has a voltage reference (7.15V), error amplifier, and a current-limiter circuit. The reference voltage (pin 6) is connected to the non-inverting input of the error amplifier (pin 5). The output voltage is sensed and reduced to 7.15V by divider R5, R6, and R4. The output of the divider (wiper of RV1) is connected to the inverting input of the error amplifier (pin 4).

When the output voltage is low, the error amplifier output (pin 10) goes more positive. The error amplifier output is connected to the gates of the MOSFETs via resistors R2, R3, and R4. These low value resistors suppress oscillation of the MOSFETs and are located close to them. The output of the LM723 is emitter follower and can't sink current to turn the MOSFETs off. R7 provides a load to discharge the gate capacitance of the MOSFETs and turn them off. C5 provides feedback compensation.

If it is too small, the circuit may become unstable. If it is too large, the output may over— or under-shoot with load changes. The LM723 current-limiter circuit reduces the output when the voltage between pins 2 (current limit) and 3 (current sense) exceeds 0.6V. This is normally sensed by a resistor in the output current path.

The value for a 20A limit is 0.6V/20A = 0.03 ohms.
Voltage Overhead in Bipolar Design

The traditional linear regulator circuit utilizes bipolar transistors. This results in a significant voltage overhead for correct operation. Overhead or dropout voltage is the additional input voltage above the regulated output voltage that is required for correct circuit operation. For higher current designs, this is composed of the regulator overhead and the pass element overhead. The LM723 overhead is a fairly substantial 3V (typical three-terminal 78xx regulators are 2V and some low dropout types are less than 0.2V).

The popular choice for pass transistors is the 2N3055 bipolar. For 20 amps output, at least two in parallel (probably four, but the number does not affect this analysis) with a third as a driver in a Darlington configuration is appropriate. The 2N3055 has a base-emitter voltage of 1.5V at higher currents. Adding 1V for the driver gives 2.5V. Additionally, the bipolar design needs emitter current biasing resistors. These are normally sized to drop 0.5V or more. Therefore, our total overhead is 3 + 2.5 + 0.5 = 6V. So, the minimum voltage at the input to the regulator is six volts higher than the desired output voltage.

This is all wasted energy converted to heat. For our 20 amp reference design, this is 6V x 20A = 120W. Some reduction could be made by using a separate supply for the regulator and driver. This could lower the overhead on the high current supply to 3.5V (3V output transistor collector emitter saturation voltage plus 0.5V biasing resistors). However, the regulator supply would itself have to be rated at two amps to drive the pass transistors.

For our 13.8V reference design, we need a minimum unregulated voltage of 19.8 volts to ensure correct operation. This equates to a 15.2V AC transformer without any allowance for line voltage drops or other losses. Typically, these designs use a 10V AC transformer. With 10V AC, this is our worst case power dissipation. This increases the power dissipation to around 220 watts for 276 watts of output power. This is less than 55% efficiency. The MOSFET design with a 15V transformer achieves 66% with a smaller transformer. Both these figures include the same bridge rectifier loss of 24V.

This is low for readily available resistors. In this design, the inherent resistance of the cable between the MOSFET source connection and the output terminal is used. This gives a poorly defined short-circuit current, but protects the MOSFETs. Any sustained overload will cause the mains fuse to fail. If an ammeter is fitted, it is included in the short-circuit sensing path.

Component Selection

A toroidal mains transformer was chosen as they are compact, cool running, and have low magnetic and audible hum levels. This example was based on a common stock size with a 15V output – 300VA. This will provide 20 amps AC (300VA divided by 15V), however, as we are using a full-wave bridge rectifier and capacitor filter we cannot take 20 amps DC.

The simple explanation is that the DC voltage is equal to the peak of the AC or 1.4 x 15V = 21V. To maintain the same power in the transformer, we must divide the current by the same factor; 20A / 1.4 = 14A. This is slightly optimistic but is usable for most applications where full load is not drawn 24/7.

For 24/7 applications, a factor of 1.75 should be used. This will give an output current of 11.4A for a 300VA transformer. The specified transformer is made by Antek (www.antekinc.com).

The bridge rectifier is a 35A 200V PIV rated unit. While a 25A bridge will work, there are advantages using the 35A. The main advantage is lower voltage drop due to lower internal resistance. The specifications indicate 40% more current for the same 1.1V drop for the 35A bridge over the 25A, or a voltage drop of 200 mV less at 20A. That’s four watts! The cost difference is minimal.

The main filter capacitor is an important component, particularly if there is a small difference between regulated and unregulated voltages. If the capacitor is too small or low quality, the smoothing will be inadequate. This causes a “ripple” at twice the mains frequency on the power supply output. Conversely, if it is excessively large, high peak currents will flow in the transformer and bridge rectifier causing overheating. So, how do we choose?

There are “rules of thumb” such as “1000 µF per amp of load,” but there is a more calculated approach. In 1943, O. H. Schade published a paper “Analysis of Rectifier Operation” in the Proceedings of the IRE. Schade worked for RCA and the paper was about tube rectifiers. However, it contains formulae and charts for calculating filter capacitor values. These are based on the source (transformer and rectifier) and load resistance ratio, and the relationship of capacitor impedance at the ripple frequency to the load resistance. This is expressed as 2πfC (C in farads, R in ohms). For 50 Hz mains (UK, worst case) the ripple is 100 Hz, so for 20,000 µF and a 0.8 ohm load (17A) 2πfC = 5.1 chose 17A as it’s between the continuous and intermittent loads.

The source resistance is not normally quoted for transformers but can be inferred from the change in output voltage from minimum load to full load (regulation). The 300VA toroidal has a measured regulation of 9%; 0.09 x 15V = 1.3V at 20A. Ohms Law gives us an equivalent resistance of 1.3/20 = 0.065 ohms. I’ll round up to 0.07Ω to allow for lead resistance. This gives a source/load resistance ratio of 0.7 / 0.8 = 0.09.

Schade’s graphs give a ripple of 15% or about 3V with a 20,000 µF capacitor. This gives a minimum unregulated voltage of 16V so we have a couple of volts in hand before ripple appears at the output at 17 amps. The voltage rating should be at least the peak off-load.
More Power

A limit to maximum current with toroidal transformers is the thickness of wire that the automated winding machines can handle. This limit is typically 10A per winding; hence, the choice of a 300VA transformer for the basic design. However, Antek makes a standard 500VA, 15V transformer (part No. AN-5415) with four secondary windings. This gives the option to build a 24 amp continuous 30A peak variation of the basic design. The changes are two 35A bridge rectifiers, three 10,000 µF capacitors, and four IRF540 transistors rather than one, two, and three respectively.

The rectifier and filter capacitor arrangement is shown in Figure A. Note that this is two separate parallel winding and capacitor circuits paralleled at the third capacitor. The resistance of the interconnecting wires helps balance the currents.

Using a capacitor with a lower ripple current rating will cause internal heating and early failure. The rating is related to size. Higher rated capacitors are larger. An alternative to a high rated capacitor is to use two or more lower rated units in parallel. The larger "computer grade" capacitors with screw terminals are preferable. This is one area where a surplus or hamfest component can be used to advantage.

If electrolytic capacitors have not been used for some transformer voltage: 15 + 9% x 1.4 = 22.9V. The minimum rating is 25V.

Another important specification is ripple current. This is the maximum AC current the capacitor has to handle. A rule of thumb is twice the load current; 30A in this case. The theoretical maximum is the AC ripple voltage divided by the impedance of the capacitor at 120 Hz (USA worst case) and the source impedance in series. This is \( \frac{3}{(1/2 + f_C + R_s)} = \frac{3}{0.133} = 22.5A \).

<table>
<thead>
<tr>
<th>ITEM</th>
<th>PART # / DESCRIPTION</th>
<th>NEWARK#</th>
<th>TYPICAL COST</th>
<th>QTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR1</td>
<td>35A 200V Bridge Rectifier</td>
<td>GBPC3502</td>
<td>$2.30</td>
<td>1</td>
</tr>
<tr>
<td>C1, C2</td>
<td>10,000 µF 35V</td>
<td>$2.95</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C3, C4</td>
<td>SLPX153M035E7P3</td>
<td>97M5165</td>
<td>$0.1</td>
<td>2</td>
</tr>
<tr>
<td>C5</td>
<td>47 µF 50V</td>
<td>17T5100</td>
<td>$0.1</td>
<td>1</td>
</tr>
<tr>
<td>D1, D2</td>
<td>1N4002 1A 100V Diode</td>
<td>05R5979</td>
<td>$0.04</td>
<td>2</td>
</tr>
<tr>
<td>D3</td>
<td>1N4753A 36V 1W Zener</td>
<td>1N4753A</td>
<td>$0.03</td>
<td>1</td>
</tr>
<tr>
<td>R1</td>
<td>470R 0.5W Resistor MFR4</td>
<td>47R0547</td>
<td>$0.07</td>
<td>1</td>
</tr>
<tr>
<td>R2, R3, R4</td>
<td>100R 0.5W Resistor MFR4</td>
<td>10R0547</td>
<td>$0.07</td>
<td>1</td>
</tr>
<tr>
<td>R5</td>
<td>1K 0.5W Resistor MFR4</td>
<td>1K0547</td>
<td>$0.07</td>
<td>1</td>
</tr>
<tr>
<td>R6</td>
<td>1K 0.5W Resistor MFR4</td>
<td>1K0547</td>
<td>$0.07</td>
<td>1</td>
</tr>
<tr>
<td>R7</td>
<td>2K7 0.5W Resistor MFR4</td>
<td>2K70547</td>
<td>$0.28</td>
<td>1</td>
</tr>
<tr>
<td>TR1, TR2, TR3</td>
<td>IRF540N MOSFET</td>
<td>63J7319</td>
<td>$4.70</td>
<td>3</td>
</tr>
<tr>
<td>U1</td>
<td>LM725CN Regulator</td>
<td>41K6277</td>
<td>$0.87</td>
<td>1</td>
</tr>
<tr>
<td>VR1</td>
<td>338F102 1K Preset</td>
<td>62J2090</td>
<td>$0.64</td>
<td>1</td>
</tr>
<tr>
<td>150AB1500MB or 325AB100MB</td>
<td>Heatsink</td>
<td>07WX4329</td>
<td>$22.87</td>
<td>1</td>
</tr>
<tr>
<td>MK3306</td>
<td>TO220 Insulator Kit</td>
<td>07WX4339</td>
<td>$6.44</td>
<td>3</td>
</tr>
<tr>
<td>Pomona 8883</td>
<td>Dual Binding Post</td>
<td>02J9383</td>
<td>$2.57</td>
<td>1</td>
</tr>
<tr>
<td>Antek AS3215</td>
<td>15V 300VA Toroidal Transformer</td>
<td>02J9383</td>
<td>$2.57</td>
<td>1</td>
</tr>
</tbody>
</table>

February 2014 NUTSIVOLTS 25
time (years), it's a good idea to connect them to a variable supply via a current-limiting resistor (100Ω) and slowly increase the voltage from zero up to the full rated voltage over a few hours. This ensures that the insulating layer on the electrodes is fully formed.

Let's talk about pass transistors. These can be just about any N-channel power MOSFET with a Vds rating of at least 50V and a current rating of at least 15A in a TO220 package. Power rating is important as the MOSFETs are in linear mode and dissipate excess voltage as heat. With 20V unregulated, 13.8V out, and 20A load, the dissipation is 124W (20V x 13.8V x 20A). My preferred device is the IRF540N. This is rated at 100V, 33A, 130W, and 175°C.

At first glance, it would seem that one of these will do the job. Scrutiny of the datasheet reveals that the 130W rating is at a case temperature of 25°C. This is virtually impossible using a practical air cooled heatsink. The MOSFET must be de-rated by 0.87W for each degree rise up to 175°C. So, that's 175-25 x 0.87 = 130W de-rating. We can dissipate 130W at 25°C or 0W at 175°C.

For a "reasonable" 65°C case temperature, one IRF540 can dissipate a maximum of 95W. However, this requires a heatsink of 40°/124° = 0.32°C/W, which is large. A hot 87°C and two MOSFETs requires a smaller 0.5°C/W heatsink, but only allows 76W per MOSFET. This is too close for comfort, so we will use three IRF540s in our design. This gives 99A current rating and good short-circuit protection.

Another advantage of using multiple MOSFETs is that it spreads the heat input to the heatsink over a larger area and more insulating washers, resulting in lower MOSFET temperatures.

**Construction**

The case used is a matter of choice. Aluminium or steel construction is preferred for safety and strength. If a plastic case is used, extra care should be taken to ensure adequate heatsinking, ventilation, component mounting, and grounding. Line switches with metal toggles should not be used with non-metallic panels. I used an old lab instrument case — complete with mains inlet, fuse holder, and switch. The hardware from a failed commercial supply could be used.

A grounded three-conductor mains cord MUST be used. Any exposed metal parts (e.g., case and heatsinks) must be connected to mains ground. Do not use the top of the toroidal transformer mounting bolt as an earth connection as this will create a shorted turn and damage the transformer. I recommend using an IEC C14 style line inlet. These are the type commonly found on PCs and can be purchased with an integral fuse holder and/or switch.

If used in a noise or RF critical application, a mains filter may be beneficial as may a 0.1 µF capacitor across the output terminals. The transformer has dual primaries and should be connected to suit your supply voltage. The fuse should be a 3.15A (115V, use a 1.6A for 230V mains) anti-surge (slo-blo) ceramic bodied type such as a LitelFuse 215P or Bussmann SS05. Glass bodied fuses should not be used. All mains connections should be
insulated. If you are not sure of any aspect for the line (mains) wiring, seek advice from a professional or experienced builder.

All low voltage power wiring should be as direct and short as possible. This particularly applies to connections between the transformer, bridge rectifier, and filter capacitors. Trim the transformer fly leads to length and solder them directly to the rectifier tabs — two fly leads per tab. Do not use push-on terminals.

The peak current in this circuit is higher than the DC output current. If the leads are enamelled copper, the enamel must be removed to get a good joint. Some enamel will self-fluxing and will vaporize when hot enough to solder. Others will have to be removed by sanding or scraping. Practice on an off cut if you haven't done this before. Don’t nick the copper.

Internal low voltage power wiring is either 14 AWG (2.5 mm²) or 216 AWG (1.5 mm²) wires in parallel. The parallel combination can be easier to handle and has slightly lower resistance. All 0V power connections should all be taken to a single point — preferably the filter capacitor negative terminal. All other components are standard parts available from many suppliers. Newark (www.newark.com) stock numbers are in the parts list. The control circuit is simple enough to be built on a prototyping board. A suitable layout is shown in Figure 2 and Photo 1.

Photo 2 shows the internal construction of my prototype. I used a single 20,000 μF capacitor as it was available.

The tabs (Drain) of the MOSFETs are connected to the unregulated supply they must be insulated from the heatsink. Thin mica washers with heatsink compound have less thermal resistance than the dry elastomeric types. Take care if using computer type heatsink compound as some are electrically conductive.

The heatsink surface under the MOSFET should be clean, flat, and free of burrs — especially around the mounting hole. Use "top-hat" insulating washers on the screws. To achieve the full rating, a total of 0.5°C/W heatsinking is required. Less can be used, but beware of excessive temperatures if fully loaded for long periods.

Three individual heatsinks (one per device) of 1°C/W can also be used. This is another area where surplus or junk box parts can be put into action. Ensure the fins or slots of the heatsink are vertical so the heated air can flow upwards. A smaller heatsink can be used if supplemented with a fan.

The bridge rectifier also requires heatsinking. This can normally be provided by the metal case of the power supply or a small heatsink of about 5°C/W.

Performance Testing

As this is a new circuit topology and optimized design, full testing was carried out. This level of testing is not required for those replicating the design. The test setup can be seen in Photo 3.

It consists of the power supply, an electronic load that switched by a square wave generator, DC current clamp, oscilloscope, and a second DC power supply. The square wave generator and load were set to switch between zero
and 10 amps at about 1 Hz. Channel 1 of the 'scope monitors the output voltage offset by the second power supply to allow higher sensitivity with DC coupling. Channel 2 monitors the current clamp output at 1 mV/amp. The resulting waveform is shown at Figure 3.

This shows a total output voltage step of 80 mV with no discernible overshoot or ringing. This is 0.6% and excellent performance. Most of the regulation error is the first 100 mA or so. A step from 140 mA to 14A gave a 30 mV drop or 0.25%.

I hope you find this power supply as helpful as I do.

NV
ROS-Training

SKIP THE STEEP LEARNING CURVE

Build, develop, create and manipulate the next generation robotics and artificial intelligence with ROS training. Sign up now!

SPECIAL OFFER: 20% OFF YOUR FIRST CLASS!
To reserve your discount sign up at www.ros-training.com/nuts_volts

LEARN BUILD SHARE

www.ros-training.com

CNC Technology in your Shop for Less

Visit us at: www.CNCShark.com
From $2899 - $3999 use code: NV1213A
When getting started with microcontrollers, you’ll quickly discover just how common it is to run out of available port lines. This is especially true when interfacing to parallel devices such as the common LCD (liquid crystal display). Inexpensive LCDs show up all the time on the surplus market, but these usually require either four or eight data lines in addition to several control lines. Of course, you could always dip into your wallet and spring for more advanced displays featuring I2C or SPI serial inputs, reducing the wiring considerably. On the other hand, with just a pittance of external parts it is easy to reconfigure even the humblest LCD to run well on only two port lines of the microcontroller. This article describes just such a technique you can readily incorporate in your own designs.
Why would you really want to roll your own? There are several very practical reasons. First, as introduced above, parallel LCDs are ridiculously cheap nowadays, and with just a handful of inexpensive components you'll get the best of two worlds. Then again, not all microcontrollers support the F2 or SPI protocols that you even be tempted to go that more expensive route. This is especially true with eight-pin chips — the very ones you're likely to need a reduced port line count for anyway.

Even better, we'll get direct software control over all of the neat visual features of the LCD — things like panning, scrolling, various types of cursors, custom characters, blinking, blanking, and lots of other things. Moreover, the methodology used here is important in such diverse applications as communications, electronic music production, data logging, and more, making this an excellent educational experience as well.

One last thing before we dig in: There are lots of designs kicking around that get the serial interface down to three wires. I wanted to go one better and reduce it further. In particular, two wires only are all we need to make the LCD respond reliably. If your curiosity is aroused, carry on and let's see how it works.

## Building Blocks and Pinouts

You've probably already deduced what the miracle ingredient is: a serial-to-parallel converter integrated circuit. I decided on the common and inexpensive 74HC595 which I picked up online through Amazon. Figure 1 shows the pinout and also gives a feel for what to expect.

Let's look at the key features. A single bit (either a zero or a one) is placed on pin 14 — the serial data input. When a low-to-high transition is applied to pin 11, that bit is clocked in to the first stage of the shift register, but only after all other bits are bumped ahead one notch to make room for it. The current contents of the shift register are still "invisible." If, however, a clock signal is next applied to pin 12, then the presently concealed contents are latched into an external storage register and appear on the associated output pins.

So, the steps are (a) put a bit on the serial input; (b) strobe the shift register clock; and (c) strobe the storage register clock to make the bits appear on the outputs. By carrying these steps out eight times in a row, an entire byte coming in serially is moved bit-by-bit down the conveyor belt and finally appears on the output pins, Q<sub>A</sub> through Q<sub>H</sub>. If, for some reason, you'd like to clear everything out and start with all zeros again, a high-to-low pulse on pin 10 will accomplish that.

We won't need most of the other pins (except for power, obviously) and will worry about the details when we get to the schematic.

Figure 2 shows the pinout for a typical LCD. Most of these are set to operate either as eight-bit or four-bit parallel devices, but the latter is more than fast enough for any application I've ever come up against. So, we'll jettison pins 7 through 10 at once simply by grounding them. Anytime we need to send a byte to the device, we'll do so in halves, transmitting one nibble (four bits) at a time.

### 74HC595 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Q&lt;sub&gt;B&lt;/sub&gt;</td>
<td>output B</td>
</tr>
<tr>
<td>2</td>
<td>Q&lt;sub&gt;C&lt;/sub&gt;</td>
<td>output C</td>
</tr>
<tr>
<td>3</td>
<td>Q&lt;sub&gt;D&lt;/sub&gt;</td>
<td>output D</td>
</tr>
<tr>
<td>4</td>
<td>Q&lt;sub&gt;E&lt;/sub&gt;</td>
<td>output E</td>
</tr>
<tr>
<td>5</td>
<td>Q&lt;sub&gt;F&lt;/sub&gt;</td>
<td>output F</td>
</tr>
<tr>
<td>6</td>
<td>Q&lt;sub&gt;G&lt;/sub&gt;</td>
<td>output G</td>
</tr>
<tr>
<td>7</td>
<td>Q&lt;sub&gt;H&lt;/sub&gt;</td>
<td>output H</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>ground</td>
</tr>
<tr>
<td>9</td>
<td>Q&lt;sub&gt;H&lt;/sub&gt;'</td>
<td>output to next stage</td>
</tr>
<tr>
<td>10</td>
<td>SRCLR</td>
<td>shift register clear</td>
</tr>
<tr>
<td>11</td>
<td>SRCLK</td>
<td>shift register clock</td>
</tr>
<tr>
<td>12</td>
<td>RCLK</td>
<td>storage register clock</td>
</tr>
<tr>
<td>13</td>
<td>OE</td>
<td>output enable</td>
</tr>
<tr>
<td>14</td>
<td>SER</td>
<td>serial data input</td>
</tr>
<tr>
<td>15</td>
<td>QA</td>
<td>output A</td>
</tr>
<tr>
<td>16</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>supply voltage</td>
</tr>
</tbody>
</table>

### LCD Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Symbol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>ground</td>
</tr>
<tr>
<td>2</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>supply voltage</td>
</tr>
<tr>
<td>3</td>
<td>VO</td>
<td>contrast</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>register select</td>
</tr>
<tr>
<td>5</td>
<td>R/W</td>
<td>read/write</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>enable</td>
</tr>
<tr>
<td>7</td>
<td>D0</td>
<td>input 0</td>
</tr>
<tr>
<td>8</td>
<td>D1</td>
<td>input 1</td>
</tr>
<tr>
<td>9</td>
<td>D2</td>
<td>input 2</td>
</tr>
<tr>
<td>10</td>
<td>D3</td>
<td>input 3</td>
</tr>
<tr>
<td>11</td>
<td>D4</td>
<td>input 4</td>
</tr>
<tr>
<td>12</td>
<td>D5</td>
<td>input 5</td>
</tr>
<tr>
<td>13</td>
<td>D6</td>
<td>input 6</td>
</tr>
<tr>
<td>14</td>
<td>D7</td>
<td>input 7</td>
</tr>
<tr>
<td>15</td>
<td>A</td>
<td>backlight anode</td>
</tr>
<tr>
<td>16</td>
<td>K</td>
<td>backlight cathode</td>
</tr>
</tbody>
</table>
LCDs do have the ability to talk back, but that feature is almost never required. So, we'll also ground pin 5 which is the Read/Write line. This is a one-way street.

It is possible to send either a command or a character to the LCD. Commands take care of such things as clearing the screen, homing the cursor, and so forth. Characters are indicated by means of the ubiquitous ASCII code set. To let the display know what the incoming byte (or more accurately, pair of nibbles) represents, use pin 4 — the Register Select (RS) line. When this is low, the byte stands for a command; when high, it is a character.

Once either a command or character has been placed on the data lines, a quick low-to-high pulse on the Enable (E) at pin 6 locks it in. A pulse width of a couple microseconds will do the trick.

The few remaining lines are fairly pedestrian in nature and will be explained once we get to the complete circuit. At this point, you have a basic understanding of the pins of both the 74HC595 and the LCD at your fingertips. As you'll see, it's remarkably easy to mate the two parts. So, let's cut at once to the gestalt!

**How They Go Together**

Figure 3 shows the schematic for a complete working circuit, consisting more of wires than anything. Let's chase...
them down. Outputs $Q_A$ through $Q_3$ of the 74HC595 connect to data inputs D4 through D7 of the LCD, respectively. Recall that inputs D0 through D3 are grounded and not needed for four-bit operation. Output $Q_E$ of the chip drives the Register Select input of the LCD. Output $Q_F$, feeds the Enable input of the LCD, but only after being processed by C1 and R4.

You can think of this RC network as being a differentiator if you want, or else as a half-monostable if that term is more comfortable. The purpose is simply to make sure the eventual Enable strobe is shorter than the actual signal appearing on $Q_E$.

IC1 output $Q_{AB}$ is first inverted by transistor Q1 and then fed back to IC1 to provide a reset or clear operation. The inversion is necessary since the shift register clear pin expects negative logic. If you already have an uncommitted inverter kicking around in your design, you can use it instead and then eliminate Q1, R2, and R3.

Output $Q_{CE}$ is not needed and is simply left unconnected. Output $Q_{CE}'$ (note the prime symbol there) is also left floating. This pin is normally employed when ganging several of the chips for more bit outputs.

The shift register clock and storage register clock lines (pins 11 and 12) are tied together and simply become the clock input. When configured in this manner, the storage register (the latch outputs, in other words) will always be one step behind what’s happening internally within the shift register.

The serial data input is found at pin 14 of IC1. Pin 8 is ground, while pin 16 is +5V. Pin 13 is also grounded to enable the tristate outputs of the 74HC595.

Back to the LCD. Pin 3 is used to set the contrast of the display; connect it to a potentiometer or trimmer. Pins 15 and 16 take care of the backlight LED, if any. Some units require a resistor as shown in the schematic (R1), while others need a power rectifier. Use whatever the datasheet for your particular LCD specifies.

Lastly, pin 1 is the LCD ground, while pin 2 is +5V. For reliable operation, it’s very important to straddle these with a 0.1 μF disc capacitor.

The Grand Dance

With the hardware out of the way, it’s now time to choreograph the flow of bits. Obviously, this will be handled by the microcontroller you choose to use. Let’s not worry about the actual code here, but instead focus on what has to happen and when. Figure 4 shows the road map.

Now, don’t panic because the progression displayed

<table>
<thead>
<tr>
<th>FIGURE 4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

Figure 5.

Figure 5 shows the connection between the microcontroller and the LCD.

there is very linear in nature, as you’d expect in serial-to-parallel conversion. The steps are labeled 0 through 10. Furthermore, each row is split in two, showing what the shift register currently holds and what the storage register (called the latch from now on) is presently outputting to the LCD. Don’t forget that the latch is always one clock pulse behind the shift register. Let’s see what happens.

At the outset (Step 0), both the register and the latch are clear, this is the normal condition during quiescence. In Step 1, a one is input. This will eventually first become the enable pulse in Step 8, and then the reset pulse in Step 9.
In Step 2, a zero is fed in. Its purpose is to bring the Enable pulse low again in Step 9.

Step 3 pumps in the RS bit — either zero for a command or one for a character. By the time we get to Step 8, it will obviously be controlling the RS line going to the LCD.

Steps 4 through 7 shift in the four data bits comprising a nibble — a pair of which will eventually make up an entire command or character byte. Keep pulsing the clock away and in time, the bits will arrive at their desired positions.

Now, Step 8 is where the action really occurs. Data lines D0 through D4 are in place, as are the RS and Enable bits. The data is locked in to the LCD here. Then in Step 9, the Enable line goes low once more, followed by a reset a split second after that. The 74HC595 is forced to an all-clear state in Step 10, which is back where we originally started.

Thanks to the software download for this article, you don’t really have to know all of the ins and outs. Nonetheless, it really is instructive to work through the steps with a pencil in your hand, just so you learn how a shift register works.

**The Software**

Just to show what’s all possible with this rig, I put together a complete software package consisting of a demonstration program along with the necessary drivers. This has been implemented for the popular PIC12F683 microcontroller which is, in fact, an eight-pin device just
About the Software

Like many modern electronics projects, this one depends a fair amount on software components as well. Fortunately, all of the programs required here are absolutely free of charge. Let’s start with the firmware for the PIC microcontroller.

Go to the article link and download the package for this article. In it, you will find source code for the drivers and a complete program demonstrating just about everything an LCD can do. The code is written in the excellent Great Cow Basic language. The nice thing here is that this language supports parameter passing and program structures like procedures and functions, so all the routines are just lying there ready for you to pluck for your own work. If you feel the need, you can even port the code over to an existing commercial Basic language like PICBasic PRO, since the syntax is virtually the same. In any event, the drivers contain routines for sending nibbles, and complete commands and characters, along with a set of equates for unlocking all of the other features in an LCD.

You can download an absolutely free copy of the Great Cow Basic compiler from www.gcbasic.sourceforge.net. There are no restrictions and no hidden catches.

The download package for this article also contains everything you need if you’d like to build the piggy-back module I described. In particular, you’ll find .pdf files of the printed circuit board (PCB) artwork and a parts placement guide, along with the complete original file. This has been created with FreePCB — an exceptionally fine PCB design package. As the name suggests, it too is completely gratis. You’ll find it at www.freepcb.com.

The PCB was designed as a double-sided affair which could certainly be sent to a professional etching facility. Since the only top-side traces are straight lines, I actually etched my own single-sided board and simply installed jumper wires on top. It was quick and easy, and works very well.
Passive Components
R1  4.7W (see below)
R2  4.7K
R3, R4 10K resistor
R5  10K trimmer
C1  47 pF disc capacitor
C2, C3 0.1 µF disc capacitor

Semiconductors
Q1  2N3904 NPN transistor
LCD1 2 x 16 LCD display
IC1  74HC395 shift register

Note: The backlight in an LCD typically requires something to drop the voltage a bit. In the unit shown here, R1 was specified. Sometimes, however, a rectifier is used. Check the datasheet for whatever LCD you employ.

is there? So, give it a try. Compile and assemble the code, and then flash it to the microprocessor. Upon power-up, you will be treated to a demonstration of all sorts of things that are possible such as blinking, scrolling, panning, extended character sets, and more.

A Stand-Alone Package

One final thing before I turn you loose to come up with your own application. I thought it might be handy to have an all-in-one arrangement for the two-wire LCD. To this end, the download package for this article also includes the printed circuit board artwork to make a piggyback affair for most any LCD you bump into. It's been designed to permit easy connection to a solderless breadboard during testing stages of circuits incorporating the two-wire LCD.

Figure 6 shows both a typical LCD, along with the piggyback board implementing this circuit. Figure 7 shows what it looks like when the two are snapped together, while Figure 8 shows the entire rig on a breadboard, shining away.

So, let's hear no more of that all-too-common groan, “I've run out of port lines!”

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

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

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

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

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

DIY NEVER HAD A BETTER FRIEND...

Take your DIY project to the next level with PanaVise work holding tools. Perfect for hobbies, soldering, electronics, crafting & repair. PanaVise tools have hundreds of uses only limited by your imagination.

PanaVise, “better than a third hand!”

www.radioshack.com/StoreLocator
Trenton Computer Festival™

Admission: $10.00 (advance)  
$12.00 at Gate  
Free Parking  
www.tcf-nj.org

@ The College of New Jersey, Ewing, NJ  
Saturday, March 15, 2014  
9:00 am — 5:00 pm  
Registration/Flea Market open at 9 am, Talks start at 10:15 am.

Focus on Multimedia in a Post-PC World

KEYNOTE SPEAKER:
Joseph Salvo,  
Manager of the Complex Systems Engineering Laboratory, GE Global Research  
on “The Internet of Things”

INDOOR FLEA MARKET & VENDOR FAIR  
50+ Talks, Workshops, Tutorials, Demos and Special Events!

Object-Oriented Programming University: Introductory Short Course by Mike Redlich

TALKS ON WINDOWS, MAC and LINUX  
2014 IEEE Region 1 Student Conference

IEEE Micromouse Competition

TCF IT PROFESSIONAL CONFERENCE  
Friday March 14, 2014, 8:30 am — 5:00pm**  
Continuing as part of TCF’s Saturday Talks.  
For Conference Info and Fees See http://princetonacm.acm.org/tcfpro/

For additional TCF’14 info, directions and advance tickets: www.tcf-nj.org

DOOR PRIZES

Integrated STEM Education Conference (ISEC)**  
Saturday, March 8 — 9:00 am — 5:00 pm
at Friend Center, Princeton University, NJ  
Designing Pathways to STEM Success.  
Info: http://ewh.ieee.org/conf/stem/

**The IT Pro (Friday) and ISEC Conferences require separate registrations.

The 2014 Trenton Computer Festival™ is sponsored by The College of New Jersey and its School of Engineering.
The working receiver in its prototype packaging.

On the left is the USB cable powering the receiver. On the right, I have my headphones plugged into the receiver. The contrast adjusting trimmer can be seen on the left top of the LCD display.
I'm always on the lookout for cool new (at least new to me) hardware components that I might incorporate into one of my electronic/computer projects.

Recently, while shopping online at SparkFun.com, I came across an evaluation board for a digital FM receiver about the size of a postage stamp that sparked (no pun intended) my interest. Having grown up listening to FM radio, I thought it would be fun to build an FM receiver of my own design. While this may seem to some as a very retro project (what with Internet radios, music players, and smartphones surrounding us), it has turned out to be quite useful. Now, I have a remote controllable FM radio that I listen to with headphones while I am working. It sits on my desk alongside the Desktop Contemplator and the Unique Digital Clock that I have written about previously. My desk is becoming cluttered with all the useful projects I have designed and built.

While I listen to my FM radio/receiver with headphones, it can just as easily be plugged into a stereo amp with speakers to provide sound for a whole room. The fidelity of this receiver is quite good when tuned to strong FM stations. Because of its compact size, you could even incorporate this FM receiver into a boom box of your own design.

This project can be built by anyone with basic electronic assembly and schematic reading experience. The software is somewhat complex, but is made simpler by the use of pre-existing libraries for the major hardware components. The Arduino Uno sketch (program) for this FM receiver project is available with the article downloads. Because you have access to the source code, you can make changes to the design and/or add new features to the receiver. If you come up with a cool new feature, please email me the code so I can incorporate it into my receiver/radio, as well.

Let's begin by discussing the hardware. Software will be discussed a little later.
Hardware

The hardware is built around an Arduino Uno board running at 16 MHz and five volts. I've been finding these boards on eBay for around $13 each, so I bought a few. Other Arduinos could be used, but some of the I/O pin assignments might need to change. Using an Arduino running at a different clock speed will also impact the design, especially in the IR (infrared) detection area. None of these problems are insurmountable but you will be in uncharted territory if you deviate from the design presented here.

I initially planned to have physical controls on the FM receiver — a power switch, a rotary encoder for channel selection, and a couple of pushbuttons for mode selection — but I quickly realized that I would then need physical proximity to the receiver to manipulate it. I then thought adding IR remote control would be interesting but soon realized that if I added that, the other physical controls would be redundant and unnecessary. In the end, I decided to go the “no control” route, so all of the receiver's functionality is controlled through an IR remote. Quite convenient really.

While not technically a control, there is a contrast adjustment for the LCD display which should be set once and left alone. I used a 10-turn screwdriver adjustable trimmer, but any 10K to 20K ohm potentiometer could be used in its place.

A schematic of the hardware is shown in Figure 1. I built my radio using point-to-point wiring (can you say rats nest?), but a prototyping shield could be used for a cleaner build.

The LCD display provides a four-bit parallel data interface to the Arduino, while the Si4703 FM receiver is connected to the Arduino via an I2C interface. Since the Arduino Uno is a five volt part and the FM receiver is a 3.3 volt part, a bi-directional level converter must be used between them. The LCD display also runs on five volts. The backlight for the LCD display is directly controlled by an output pin from the Arduino.

The stereo audio output cable is used as the antenna for the FM receiver, so you must pay attention to the length of the interconnect. The cord on the headphones I use works fine as an antenna since I can pick up all of the FM stations in my area.

When strong stations are tuned in, the audio quality is top notch. When the receiver is receiving a stereo broadcast, the yellow stereo indicator LED lights up.

---

PHOTO 1.

Three components:
LCD, Arduino Uno, and receiver board.
Weaker stations are received in mono and the stereo indicator remains dark. A red LED lights when the radio is on.

An IR receiver (which is what RadioShack called it) is used to detect IR codes from the remote control. It filters out the 38 kHz IR carrier frequency from the received signal, thereby making detection of the key codes more straightforward. The radio is powered via a USB cable and a USB power supply. Alternatively, the radio can be powered by connection to a USB port on your computer. There is no power switch. If you want to power-down the radio, unplug the USB cable and/or power supply.

Wiring and wire routing for this project are non-critical. Be advised that as with all digital circuitry, keeping the wires as short as possible/practical is always a good idea. Using a consistent color scheme for the wires is also recommended. I used red for five volts, white for 3.3 volts, and black for all ground connections. Various other wire colors are used for data and clock signals.

**Packaging**

I packaged my radio using two 4” x 6” pieces of clear 1/8” acrylic plastic in a sandwich-like arrangement, held together by wooden 1-1/2” dowel spacers in the corners. I like the naked electronics look. The LCD display is mounted to the front acrylic piece and the Arduino Uno and receiver board are mounted to the rear piece. The finished radio is free standing with this packaging approach. Check out Photos 1 and 2 for details.

**Software**

All of the radio’s software was developed using the Arduino IDE (Integrated Development Environment) version 1.0.5 for OSX. Windows version of the IDE is available if that is your chosen computing environment. Make sure you have the board type set to Arduino Uno and the serial port set appropriately in the IDE. Of course, you will need to plug the receiver’s Uno into your computer via a USB cable to download the provided firmware via the IDE.

Three libraries are used in this project to ease the software development task: the Liquid Crystal library which comes standard with the Arduino IDE; an IR remote library for IR code detection; and a library for controlling the Si4703 digital FM receiver. The Resources section has pointers/links as to where these libraries can be obtained.

---

**PHOTO 2.** Receiver board close-up. At the top is the Si4703 evaluation board with the black 1/8” stereo output jack. In the middle is the IR receiver facing upward, along with some filter caps. Towards the bottom is the four-channel level converter; the yellow rectangular LED is the stereo indicator and the red LED is the power-on indicator.
The Liquid Crystal library is preinstalled when you download the IDE, so you need not do anything else to use it other than including LiquidCrystal.h in your sketches. To install the IRremote library, you must first unzip the downloaded IRremote library file somewhere on your computer and then copy its contents to your arduino\libraries directory. Finally, rename the ArduinoIRremote-master directory to IRremote and you should be good to go.

Installing the Si4703_Breakout library is similar. First, download the library, unzip it, copy its contents to the arduino\libraries directory, and then rename the directory called Arduino-Si4703-Library-libcode-only to Si4703_Breakout.

The Arduino IDE must be shut down while installing new libraries. Once restarted, the IDE should pick up anything newly installed. You can check for proper installation by clicking Examples from the File menu. There you should see an IRremote entry with a bunch of example sketches, along with a Si4703_Breakout entry with its example sketch.

Whereas the Liquid Crystal and the IRremote libraries can be used as is, the Si4703_Breakout library must be modified before being used in our application. This library was not built with the idea that someone might want to extend it with functionality the library didn’t directly provide.

For our application, I needed the ability to mute/unmute the FM receiver and to poll the stereo reception indicator — functionality the library doesn’t provide. Luckily, the change to the library is trivial and only involves editing the header file (Si4703_Breakout.h). Again, the Arduino IDE must be shut down while editing is being performed. Bring up Si4703_Breakout.h in a text editor of some kind. On my Mac, I use TextEdit; on Windows, you could use Notepad.

The change involves moving the following entries from the private section of the interface definition up to the public section.

Once you have moved these four lines of text, save the file, and we should be ready to go:

```c
void readRegisters();
byte updateRegisters();
int getChannel();
uint16_t si4703_registers[16];
```

//There are 16 registers, each 16 bits large

With this change, we are able to extend the functionality of the Si4703 library as required for our application.

**Software Overview**

You may want to refer to the DigitalFMRadio.ino sketch during this discussion.

The sketch is broken up into numerous sections for organizational purposes. At the top of the sketch are the global definitions, followed by the hardware definitions where all of the I/O pins for the radio are assigned. Following that, there are sections for the major hardware elements of the design in the following order: IR Receiver, the LCD display, the Si4703 FM receiver, EEPROM, and finally, miscellaneous functions. Following that are the setup() and loop() Arduino functions.

In the IR receiver section, an instance of IIRrecv called irReceiver is declared and assigned the IR_RECEIVER_PIN defined earlier. This assignment connects the IR receiver hardware to the software that will manage it. Next, the IR remote control key codes for the AdaFruit remote (see
Photo 3) are declared. I listed all of the key codes available from the remote control—not just the ones used in this sketch.

Finally, two convenience functions are defined which wrap functions in the IR receiver library for ease of use.

In the LCD section, an instance of LiquidCrystal calledlcdis defined and is passed to the hardware I/O pins used in this design. Again, this connects the hardware to the managing software. A single function is then defined that will clear a specified row of the LCD display when called.

The Si4703 section is a little more complicated in that we have to extend the functionality of the Si4703_Breakout library. First, though, an instance of the library is created calledradio that is passed to the I/O pins used in this design. The Si4703 FM receiver is controlled using a series of 16-bit registers. In a typical operation, the registers are read from the chip, values are changed, and the updated register values are sent back causing the receiver chip to react.

The functionupdateStereoIndicatoris a little different in that we are only polling the status of the receiver to determine if a stereo signal is being received to determine if the stereo indicator LED should be lit or not. In themuteRadiofunction, we read the registers, set or clear the DMUTE bit (depending on whether or not we are muting), and then send the modified registers back to the chip.

The EEPROM section has functions for reading and writing eight- and 16-bit unsigned integers from the EEPROM contained within the Arduino’s processor. Values written to the EEPROM survive loss of power and will be available indefinitely until changed.

The Miscellaneous section has functions for displaying the FM station frequency on row 0 of the LCD display; for displaying row 1 messages on the LCD display; for retrieving and storing FM station presets; and for processing presets. TheprocessSetPresetfunction waits for an IR key to be pressed on the remote control and then stores the channel/station that is currently being listened to in a corresponding preset in EEPROM.

The Arduino setup() function prepares the hardware for operation. Here, the backlight, the power-on, and the stereo LED control pins are configured as outputs. The IR receiver is enabled, the LCD display is configured for 16 character by two row operation, some volume and channel defaults are installed, and the EEPROM is prepared for use.

To prevent bogus preset values from being used, I wanted to do a one time initialization of the EEPROM, setting each preset value storage location to zero. To make sure this is only done once, the code looks for a signature in the first two bytes of the EEPROM. If the

<table>
<thead>
<tr>
<th>Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vol-</td>
<td>Mutes the audio</td>
</tr>
<tr>
<td>Play Pause</td>
<td>Turns the radio off and on</td>
</tr>
<tr>
<td>Vol+</td>
<td>Un-mutes the audio</td>
</tr>
<tr>
<td>Up Arrow</td>
<td>Volume up</td>
</tr>
<tr>
<td>Down Arrow</td>
<td>Volume down</td>
</tr>
<tr>
<td>Left Arrow</td>
<td>Scan down for a station/channel</td>
</tr>
<tr>
<td>Right Arrow</td>
<td>Scan up for a station/channel</td>
</tr>
<tr>
<td>Enter Save</td>
<td>Sets up for saving a preset. Tune in the desired station, press Enter Save, and then a key (1 .. 9) to save the station as a preset.</td>
</tr>
<tr>
<td>Keys 1, 2 .. 9</td>
<td>Tunes station if preset is set; otherwise, does nothing.</td>
</tr>
</tbody>
</table>

### TABLE 1.

<table>
<thead>
<tr>
<th>ITEM</th>
<th>PART #/DESCRIPTION</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino Uno</td>
<td>16 MHz Five volt part</td>
<td>SparkFun, AdaFruit, RadioShack, eBay</td>
</tr>
<tr>
<td>Bi-directional Level Converter Evaluation Board for Si4703 FM Tuner</td>
<td>BOB-12009 WRL-10663</td>
<td>SparkFun</td>
</tr>
<tr>
<td>16x2 Line LCD Display</td>
<td>Any LCD noted to be Arduino compatible should work. #2750640</td>
<td>SparkFun, AdaFruit, eBay, eBay</td>
</tr>
<tr>
<td>IR Receiver</td>
<td></td>
<td>RadioShack</td>
</tr>
<tr>
<td>Red LEDs</td>
<td></td>
<td>AdaFruit</td>
</tr>
<tr>
<td>Yellow LEDs</td>
<td></td>
<td>SparkFun, AdaFruit, RadioShack, eBay</td>
</tr>
<tr>
<td>2 x 300 ohm 1/4W resistors</td>
<td></td>
<td>eBay</td>
</tr>
<tr>
<td>10K ohm 10-turn Trimmers</td>
<td></td>
<td>SparkFun, AdaFruit, RadioShack, eBay</td>
</tr>
<tr>
<td>2 x 10 µF @ 25V Capacitors</td>
<td></td>
<td>eBay</td>
</tr>
<tr>
<td>Mini Remote Control</td>
<td>ID: 389</td>
<td>SparkFun</td>
</tr>
<tr>
<td>USB Cable for Arduino to USB Power Supply Connection</td>
<td></td>
<td>SparkFun, AdaFruit, RadioShack, eBay</td>
</tr>
<tr>
<td>USB Power Supply 500 mA or greater</td>
<td></td>
<td>SparkFun, AdaFruit, eBay</td>
</tr>
<tr>
<td>.1” Male Breakaway Header Pins for Arduino connectors Wire, Solder, Packaging, etc.</td>
<td></td>
<td>SparkFun, AdaFruit, eBay</td>
</tr>
</tbody>
</table>

February 2014 NUTSIVOLTS 43
Resources

The following sites may be of interest to those seeking more information on the topics described in this article.

Information about the Si4703 digital FM receiver can be found at the manufacturers website at www.silabs.com. AN230, AN231, and AN243 are application notes concerning the Si470x series parts. AN332 provides example code useful for programmers.


The IRremote library is available at github.com/shirriff/Arduino-IRremote. There is a Download ZIP button in the lower right side of this page. Click it to get the library.

The Si4703_Breakout library is available at github.com/infomaniac50/Arduino-Si4703-Library. Again, click the Download ZIP button to get the library.

An AmForth version of the firmware is also available. Contact the author for details.

signature is not found, the two-byte signature 0xAA, 0x55 is written and all 10 preset locations are set to zero.

If the signature is found — which is usually the case — EEPROM initialization is skipped.

The Arduino loop() function is where the radio is controlled. It consists of a large switch statement that is driven by the key codes received from the remote control. Table 1 details which keys do what.

It is important to note that when power is applied to the radio, it appears to be off even though it really isn’t. Instead, the firmware is constantly looking for the Play Pause key code to be received to virtually turn the radio on. A variable called radioOn tracks whether the radio is off or on. The processing of all key codes is conditional on this variable since I didn’t want the radio responding to commands while it was supposed to be off. Most cases have similar structure.

First, the radioOn variable is checked and if the radio is on, a function is performed and a message is written to the LCD display. If radioOn is false, the received key code is ignored. Processing of the Play Pause key is the most complex as numerous steps are necessary to virtually turn the radio off or on. The comments in the code should make it clear what is happening.

The cases used for retrieving a preset should be mentioned. There are nine presets available, numbered 1 through 9. When a preset key on the remote is clicked, the corresponding preset is retrieved from the EEPROM and its value is examined. If a value of zero is returned, the preset has never been set so the radio does not respond. If, however, a non-zero value is returned, that channel is set.

Preset 10 is a special case. When the radio is virtually turned off, the channel that was being listened to is written to preset 10. Then, when the radio
is virtually turned back on, preset 10 is read and that station/channel is reinstated.

**Conclusion**

Building your own FM receiver is cool and easy — even if listening to FM radio the traditional way over the air seems retro. **NV**

---

**File System Solutions**

**ALFAT**

- **System on Chip**
  - **$9.68**
  - **@ 480**

**ALFAT SD Board**

- **$37.95**
  - **@ 100**

**ALFAT USB Board**

- **$37.75**
  - **@ 100**

**ALFAT OEM Board**

- **$49.95**
  - **@ 100**

---

Access files on SD/MMC cards and USB drives.

Use UART, SPI or I2C serial interface to access files.

Use Long File Name (LFN) from Microsoft without further licensing.

High speed 4-bit SD card interface.

USB High-Speed 480 Mbps through external ULPI PHY.

---

**expresspcb.com**

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

---

**GHIElectronics.com**

February 2014  NUTSIVOLTS  45
THIS IS RADIO DISASTER CALLING

When the Cell Towers Fail, Ham Radio Operators Keep on Talking, and Talking, and Talking ...

Debris lines the streets of Tacloban, Leyte Island. [Image courtesy Trocaire — a charity and development agency from Ireland.]
Okay, the unthinkable just happened and you’re totally cut off from all communications: no cell phones, no landlines, and no way of reaching out to the rest of the world. Who you gonna call?

Well, if you’re prepared for disaster, then you just dust off your amateur radio transceiver, hook up your automobile battery for powering your equipment, and begin transmitting and receiving communications from anywhere in the world.
Affectionately known as "hams," these radio operators (NOTE: A ham’s equipment is also known as ham radios) are able to continue the flow of information to and from a disaster relief area within moments of any kind of emergency. Especially those types of emergencies where the normal lines of communication (i.e., cell phones) are either overloaded (e.g., Boston Marathon bombing) or gone (e.g., Hurricane Katrina).

Fueled by automobile batteries or banks of solar cells and bristling with VHF/UHF satellite and HF antennas, amateur radio operators are able to both get the word out of a stricken area and — more importantly — get the word into a disaster area. It is the flow of information that helps to minimize panic and assist in recovery.

**CQ, Anybody?**

In this day of mobile telecommunications, it might be unknown to most people that amateur radio operators are a major communications backbone throughout the world. This involvement is no clearer demonstrated than in times of natural disasters. Looking strictly at the United States, ham radio operators are divided into several vital roles for providing communications during relief efforts.

The two more common disaster relief roles are through the American Radio Relay League (ARRL) Amateur Radio Emergency Service (ARES) and the Radio Amateur Civil Emergency Service (RACES). While the former is a private organization that lends a helping hand to the American Red Cross, Salvation Army (e.g., Salvation Army Team Emergency Radio Network), volunteer fire departments, and National Weather Service (for example, as storm spotters), the latter is a drop-in replacement communications service that is sanctioned by the FCC.

Regardless of which hat a ham is wearing, however, they are all linked together through a requisite membership within the scope of the national Amateur Radio Service.

Similarly, there is a strong support contingent of amateur radio operators worldwide, including those in the United Kingdom, Australia, and many other countries.

According to the US Navy Joint Typhoon Warning Center, Typhoon Haiyan had maximum sustained winds of 195 mph (314 kilometers per hour), with wind gusts up to 235 mph (379 kilometers per hour) shortly before hitting landfall in the central Philippines. Weather officials in the Philippines reported the storm — known locally as Typhoon Yolanda — hit with maximum sustained winds of 147 mph (235 kilometers per hour) and gusts of up to 170 mph (275 kilometers per hour).

This Category 5+ storm that barreled through the Central Philippines in November 2013 left survivors without food, water, shelter, electrical power, and cell phone service. Springing into action, the Philippine Amateur Radio Association (PARA) immediately issued pleas for emergency communication equipment for supporting PARA’s Ham Emergency Radio Operations (HERO) network. Designed to establish a presence in less populated and harder to reach areas, PARA conducted a painstaking solution to a logistical nightmare. Two weeks after the typhoon’s landfall, PARA had successfully established a complex network of ham radio stations throughout the affected region.

What began with only one local PARA station — Lester, DVSPO, in the capital town of Borongan, East of Samar — quickly mushroomed into a network of makeshift stations. Beginning with a relief operation shepherded by Darwin Torres 4F1FZE (a veteran ham operator), PARA
operators that provides operations as varied as involvement with the Federal Emergency Management Agency (FEMA) and the Department of Defense (DoD). Unlike the natural disaster response, the DoD support is during a national crisis with the Military Auxiliary Radio System (MARS) when ham radio operators are enlisted to provide emergency communications on restricted military radio frequencies (pursuant to invocation of the President's War Emergency Powers under the provisions of section 706 of the Communications Act of 1934, as amended, 47 U.S.C. 606).


In grossly simplified terms, there are three prime definitions that constitute "emergency communications:" (1) safety of life and protection of property; (2) station in distress; and (3) radio amateur civil emergency service (or RACES). Okay, that's a lot of government jargon, but what are the "real" actions required of a ham radio operator when these emergency communications are needed? First and foremost, each operator is required to get his/her house and property in order. In other words, make sure everyone and everything you care about is safe and secure. Then — and only then — an amateur radio operator is required to report to work as an emergency communications operator.

Based on the training for each operator, there will be a set of primary frequencies that each ham is required to monitor. Other emergency personnel on these frequencies will provide the ham with instructions for either relay or implementation. At that point, the operator might have to relay some vital information "up the line."

With the radio set to its maximum power level, the amateur radio operator will say "Break, Emergency" and wait for a repeater station to respond. When he/she has acquired a repeater, an emergency autotach system is activated for making 911 telephone calls.

Once the 911 operator is connected, the ham will indicate that this is an emergency call that is being implemented through amateur radio. The remainder of this emergency call will proceed like a typical 911 cell phone call. At the conclusion of the transmission, however, the ham must release the autotach.

**QSL**

"MAYDAY, MAYDAY, MAYDAY, this is ...," is the standard beginning for a station in distress call.

---

based operations in Tacloban. PARA then implemented a plan to improve VHF coverage over the area, with HF remaining a critical component.

Using battery- and solar-powered repeater stations in Tacloban, teams were deployed to Samar, Guian, and further west linking Samar to Tacloban and vastly improving coverage over a significant piece of the area surrounding Samar.
The National Aeronautics and Space Administration (NASA) has long been a strong and avid supporter of amateur radio operators. From maintaining the Ames Amateur Radio Club station NA6ME, to successful collaborations between NASA and volunteer amateur ham radio operators from around the world, NASA is a world leader in enabling researchers and hams to work together as citizen scientists.

Case in point: A recent NASA mission enlisted the combined efforts of 100s of ham radio operators from around the world, on every continent, except Antarctica. This NASA "nanosatellite" mission — called PhoneSat — which blasted into space on Sunday, April 21, 2013 aboard an Antares rocket from NASA's Wallops Island Flight Facility in Virginia, contained three miniature satellites built around three commercial off-the-shelf (COTS) smartphones (e.g., Nexus One made by HTC Corp., running Google's Android operating system). The goal of PhoneSat was to determine whether COTS devices — like a smartphone — could be used as the main flight avionics system for a satellite in space.

The three satellites — named Alexander, Graham, and Bell — used their smartphone cameras to take pictures of Earth. These photographs were then transmitted as "image-data packets" to multiple ground stations. Each packet held a small piece of graphic data that contributed to "the big picture." As the data became available, the NASA PhoneSat Team and the volunteer amateur radio operators would piece together the data packets into a high-resolution photograph of Earth.

This collaborative effort was declared a success by Bruce Yost, program manager for NASA's Small Satellite Technology Program. "The PhoneSat project also provided an opportunity for NASA to collaborate with its space enthusiasts. Amateur radio operators from every continent (but Antarctica) contributed in capturing the data packets we needed to piece together the smartphones' image of Earth from space." The mission successfully ended Saturday, April 27, 2013 when PhoneSat reentered Earth's atmosphere and burned up. You can see the results from this mission at www.nasa.gov/topics/technology/features/PhoneSat_PHOTO_feature.html.

Accompanying this prefix will be the latitude and longitude for the station, a description of the emergency, and a statement about what type of service is needed by the station. Depending upon the severity of the disaster, this call is repeated until either a reply is received or the station is out of danger.

In certain situations, the station might attempt to relocate its distress call to another frequency. Prior to changing frequencies, the station should indicate the new frequency that is going to be used before the change is made.

Adhering to this frequency change procedure will ensure that other stations who are listening to the distress call but can't be heard by the station in distress will be able to continue monitoring the crisis.

Conversely, operators who hear a distress signal must immediately begin recording the event. A timestamp and frequency notation should be appended to this recording. The proper response to a distress call is to indicate the station's call sign that you received, along with your call sign, followed by asking the station's disposition. After this acknowledgement, the receiving station will be a relay for emergency action from 911, civil authorities, or DoD controllers, and will remain on the air until the emergency is resolved.

73

Believe or not, this system of radios, telephones, government regulations, operators, and hams really works during times of natural
disasters. In fact, it works really, really well. Consider that in the aftermath of Hurricane Katrina, Congressional testimony exploring the government's emergency response to this disaster cited the use of amateur radio operators as one of the few bright spots in an otherwise lackluster relief effort. Likewise, despite the rise in social networks being used following the impact of super-storm Sandy in 2012, the overwhelmed cell networks were pushed into the background and ham radio operators were able to keep information flowing throughout New York and New Jersey.

Whether it's knowing about an impending storm threat, the site of a local water distribution center, or simply that someone "out there" cares about you, a large comprehensive communication network is vital when a disaster cuts you off from the rest of the world.

That's exactly what amateur radio operators do. They keep information going when the rest of our "modern" services are out. NV

2010 WPCX SSIB AT THE W7RN CONTEST STATION USING FOUR ELECRAFT K3S.
[Photo courtesy Elecraft.]

NASA ASTRONAUT DOUG WHEELOCK, EXPEDITION 24 FLIGHT ENGINEER, USES A HAM RADIO SYSTEM IN THE ZVEZDA SERVICE MODULE OF THE INTERNATIONAL SPACE STATION. [Photo courtesy NASA.]

How to Become a Ham Radio Operator

Are you interested in becoming an amateur radio operator? If you live in the US, it's just a matter of taking an examination and obtaining a license from the Federal Communications Commission (FCC). A license is valid for 10 years and is available in three classes: Technician, General, and Extra.

Once you've obtained your entry level Technician Class License, you can upgrade to the higher classes. Although the higher classes offer more communication privileges, the Technician Class License is a great way to sample the field.

In order to obtain your Technician Class License you must pass a 35 question Technician Written Exam. This exam does not require that you understand Morse Code, but you will need to demonstrate a mastery of basic FCC regulations, radio operating practices, and electronics theory.

Once you've received your Technician Class License, you will access privileges to all VHF/UHF amateur bands above 30 MHz, including the two meter band—which is capable of supporting handheld radios. Furthermore, Technician Class hams can operate FM voice, digital packet, television, single-sideband voice, and make international radio contacts via satellites. Additional frequencies are open to Technicians on the 80, 40, and 15 meter bands using continuous wave (CW), and on the 10 meter band using CW, voice, and digital modes.

The cost for basic coursework and license application fee is around $40. The American Radio Relay League is a national association that can assist you with your license, as well as provide you with a vast network of education and support for all of your ham needs.

For more information, visit www.arrl.org/what-is-ham-radio.

THE TDXONE TD-O8 HT COMPACT HANDHELD TRANSCEIVER KIT INCLUDING LICENSE STUDY GUIDE AND FCC RULE BOOK FROM RADIO CITY, INC., FOR $69.95.
[Photo courtesy of TDXone.]

ELECRAFT K2/100 (K2 + KPA100): K2 HF TRANSCEIVER KIT, $759.95; AND ELECRAFT KPA100 100W OPTION WITH SERIAL INTERFACE KIT, $399.95.
[Photo courtesy Elecraft.]
An important MakerPlot function is its bi-directional capabilities. What this means is that your micro can control and be controlled by MakerPlot. As we've mentioned before, you should think of MakerPlot as a front panel GUI for your micro because you can create meters, LEDs, and switches that can monitor and control your micro's operation just like a physical front panel can, along with plot areas for your analog and digital data.

In this two-part feature of our series, we're going to introduce you to how MakerPlot accomplishes bi-directional monitoring and control by first adding two physical pushbuttons and an LED, in addition to the 10K pot to our Arduino Uno hardware. Then, we'll go on to show you how to change the setpoint with these two switches, and also how to code the sketch to establish the setpoint and make things happen. After that, we'll show you how MakerPlot can operate with the Arduino for data acquisition and control of the setpoint function.

To get a handle on things, this article begins with uni-directional or one-way control of MakerPlot using an the Uno. So, instead of you clicking buttons and switches on the MakerPlot Interface to change things, we're going to show you how to do it with MakerPlot instructions that are embedded in the Arduino sketch and sent to MakerPlot using the same serial connection that outputs the analog and digital data. (If you haven't already done so, download a free 30 day trial copy of MakerPlot from www.makerplot.com to follow along.) Let's get going.

**Expanded Arduino Uno Setup**

To begin our setpoint example, we've added two physical pushbutton switches and an LED to the Arduino Uno so that we can change the setpoint value for the pot using them ([Figure 1](#)). Switch SW1 controls incrementing the setpoint and SW2 controls decrementing it. The LED is only there to show you...
when the potentiometer analog value goes above or below the setpoint without having to look at the MakerPlot display.

Figure 2 illustrates the data flow from the Arduino into MakerPlot. Right now, it’s only in one direction (micro to MakerPlot), but that’s just to get things started. Now, let’s look at the code.

**Setpoint Code**

Figure 3 is the sketch that defines how to get the added switch inputs into MakerPlot. You can find this at [www.makerplot.com](http://www.makerplot.com) Arduino Sketch Examples → Arduino Setpoint Example. Load it onto your Arduino to follow along.

We’ve added constants to establish where the 10K pot is attached, as well as the new switches and LED on the Uno: SW1 is on pin 2; SW2 is on pin 3; and the LED is on pin 9, with the center tap of the pot attached to analog input A0.

The setup part of the sketch simply enables the internal pull-ups for SW1 and SW2, and defines the LED as an output. The loop part of the sketch is equally straightforward. In effect, if SW1 is depressed (logic 0), the setpoint is increased; if SW2 is depressed, it’s decreased. Also, if our pot level — the raw 0 to 1023 analog-to-digital (A2D) value — is above the setpoint, the LED on the Uno will illuminate; below it, it will extinguish.

Also, the last part of the loop code sends the setpoint and pot information to MakerPlot where the digital setpoint, SW1, and SW2 digital values are plotted on the top (Figure 4), along with the analog potentiometer (black) and setpoint (red) levels on the bottom.

**Setting up MakerPlot for Remote Control**

We’re using the same Digital Monitoring Interface as in the previous article as it has LEDs on the right for the digital setpoint SW1 and SW2 switches, as well as two analog bar graphs for both the potentiometer (Analog 0) and setpoint (Analog 1) values.

To make things clearer (Figure 5), we’ve changed the top three digital LED designations from Bit 0, Bit 1, and Bit 2 to Setpoint, SW2 - Decrease, and SW1 - Increase to better reflect the digital setpoint, SW1, and SW2 pushbutton
switches. To do this, just key the new names into the text boxes and click the Update Traces button below. We also blanked out the Bit 3 to Bit 7 designations since they’re not used in this example. Figure 5 also shows how the two analog bar graphs were renamed to represent the Pot value (top bar graph) and Setpoint (bottom bar graph). This is easily done in MakerPlot by simply putting your cursor in the text box and keying in the new name for the control.

**MakerPlot as a Front Panel**

In this simple setpoint example, we want to demonstrate how MakerPlot can become a GUI front panel for your micro’s data and control functions like the two physical switches and LED on the Uno. According to the sketch code, we’ve established the setpoint at an arbitrary value of 100, so that’s what we’ll adjust first. This is the line of code that does it:

```c
int setPoint = 100;
// Initial value of setPoint
```

For example, in Figure 6 when SW1 is pushed, you’ll see the digital trace go from high to low (negative logic) and the SW1 LED extinguish. You can also see the analog setpoint (the red line) move up. Going the other way, in Figure 7 when SW2 is pushed to decrease the setpoint, the digital trace goes from high to low and the SW2 LED extinguishes again, negative logic — plus the red setpoint trace decreases along with the Setpoint bar graph.

When the setpoint is stationary, Figure 8 illustrates how adjusting the potentiometer value (black trace) back and forth through the setpoint (red trace) causes the digital setpoint line to change from high to low and back again, along with the setpoint LED on the Uno, flashing as the setpoint is crossed. Even though you can’t see this, you can imagine what happens. To see a “live” demonstration of this setpoint example, go to [www-makerplot-com](http://www-makerplot-com) and click on the Learn Video → Arduino Setpoint Example. In this short video, everything we’ve just covered is explained in much greater detail, and you can actually see what takes place.

**Controlling MakerPlot from the Micro**

We’re going to expand the setpoint example a bit. We can keep the same physical Arduino hardware setup with the 10K potentiometer, pushbutton switches, and LED, but we’ll need to add a few more lines to our sketch in order to make it add remote control to MakerPlot. For further reference, Figure 9 illustrates the data flow for our modified sketch.
In order to establish “hands off” control of the MakerPlot Interface, the micro will send commands (instructions) as well as data to MakerPlot via the same serial link. Up until now, we’ve shown you how to send only analog and digital data; now, let’s make the leap to controlling MakerPlot directly with your micro by also sending instructions to it. Figure 10 is our modified sketch that does this. You can find it at www.makerplot.com Arduino Sketch Examples → MakerPlot Arduino Bi-Directional Control – Part 1. Load it into your Arduino if you’re following along.

A typical MakerPlot command or instruction starts with an exclamation point (!). Look at the excerpted code in Figure 11; the first instruction is a MakerPlot reset instruction (!RSET) which clears the Interface of all plots and sets it to the default condition. Like serial data, the reset command is proceeded by the Arduino’s Serial.println instruction. Then, the instruction is encased in quotes (“ “) to make it ASCII, which is then followed by a carriage return. The Serial.println instruction handles sending the carriage return.

Another common MakerPlot designator is the capital letter ‘O.’ This is an abbreviation of the POBJ — or Plot Object — MakerPlot command. The next four instructions use the ‘!O’ prefix to tell MakerPlot to switch from seconds to minutes on the X-axis (time), and to set the time range to two minutes. The final instruction in this group sends the text “Controlled from Arduino” to the Interface.

When the sketch is loaded and run, the result looks like Figure 12. We’ve switched to the Interactive Interface, since we’re going to use the Slider control on
the right side for the rest of this example.

What Just Happened?

It’s important to understand what just happened in these few lines of code. Look at the bottom menu buttons in Figure 12 for reference. Rather than manually clicking on the Reset Plot button to clear the screen, then clicking on the MIN button to switch from seconds to minutes on the time scale — followed by setting the time duration to two minutes with clicks to the DBL or HLV buttons — we did all this using these first five serial commands to MakerPlot from the Arduino (Figure 11).

Okay ... now let’s look at what else our remote control sketch does. In the first part of the loop (Figure 13), it reads the potentiometer value then goes on to test it against the current setpoint. If the potentiometer value exceeds the setpoint, the code instructs MakerPlot to “snap an image” of the Interface screen. This is handled with these three lines of code:

```c
Serial.println("10 butSnap=1"); // Turn on Snapshot button
Serial.println("10 butSnapRun=1"); // Run snapshot event code
Serial.println("10 butSnap=0"); // Turn off snapshot button
```

Notice the “10” prefixes that precede the butSnap instructions. This three-line sequence is equivalent to manually clicking on the Snapshot button in the menu section. The result is an image of when the potentiometer value crossed the setpoint line, and Figure 14 shows it. To view it, click the View Snap button in the Logging menu. The jpg image looks just like the “live” Interface, so it’s easy to confuse it from the real thing. You can use these jpg images as a visual record of your experiments or process control activities. If you’re a student (or teacher), these same images can be incorporated into lab reports.

**Figure 11. MakerPlot Instruction Examples.**

Winding Up for Now

The reset of the loop code (Figure 10, again) simply outputs the setpoint, SW1, SW2, and potentiometer data values to MakerPlot. What we
want to do next is add more code to the sketch so that we can use the Slider control on the Interactive Interface to adjust the analog setpoint value. This will be the beginning of real bi-directional control (MakerPlot → micro). We’ll do that in the next article. In the meantime, you can view the video of this article at www.makerplot.com Learn → Arduino Setpoint Example with GUI Control. The video will give you a better idea of how everything works.

**Conclusion**

In this first of a two-part feature on bi-directional data acquisition and control, we showed you how your code can affect one-way control of MakerPlot by outputting commands from the microcontroller instead of clicking on buttons and switches on the Interface screen.

We also introduced you to some basic MakerPlot instructions. These are the same instructions that are used to create the controls (buttons, switches, meters, etc.) on the Interface screen. What’s interesting to note is that these same instructions can come directly from your micro, so
Figure 13. MakerPlot snapshot instructions.

It’s entirely possible to completely build an Interface directly from your micro. We’ll get into that in a subsequent article.

In the meantime, remember that MakerPlot is available as a free download. If you like what you see and what it does, you can order it from the NV Webstore at a discounted price.

That’s all for now, so just remember: Got Data – MakerPlot It! NV

Figure 14. Snapshot jpg image of setpoint crossing.
available to the transistor. Using that current and the beta of the transistor, you can then calculate the resistance required: \( R_b = (V_{in} - V_B)\left(\frac{I_C}{I_{C_{max}} + \beta}\right) \). Lower values of \( R_b \) will also work but cause a higher \( I_b \) than what is required.

Larry Cicchinelli K3PTO

You are, of course, right on the money. Not sure how this one passed through the cracks, but it did.
Thanks for the eagle eyes.

Bryan Bergeron
Editor

To Zip or Unzip

On page 81 of the January 2014 issue, Margaret Lyell talks about ZIP and .GZ files, and all of the steps to move them to a Raspberry Pi. Has everyone forgotten that these many steps are not necessary?

Why not Unzip the files first on your PC or MAC, and then use WinSCP to move them over to the RPi? Simple!

Ted Mieske

Thanks, Ted. Great feedback on the feedback!

Bryan Bergeron

A Closer Look

Regarding the January 2014 Developing Perspectives editorial, electronic magnifiers have their uses, but the Luxor magnifiers are wonderful when you’re drilling PCBs. I have a stand-mounted Dremel tool with carbide bits chucked into a collet. My chair is low enough that the drill press table is at eye level. The Luxor lamp is swung into position between the work and my face, providing three important functions: excellent lighting, good magnification, and — most importantly — eye protection from debris and breaking drill bits. The setup makes PCB drilling most comfortable, even over long time periods. In addition, I have a small muffin fan at the side of the work to blow most of the debris off to the side.

Dean Huster

Thanks for taking the time to write. I have a Luxor on my desk, as well. The halogen bulb makes a great heat shrink tubing heat source when I just have one piece to work with and don’t want to pull out the big heat gun. I’ve been saved from solder splatter on several occasions by the large lens. Glad you’ve made use of your investment, as well.

Bryan Bergeron

Passing Compliment

Regarding the question in the December 2013 Q&A: Nice job with the KCL exercise with the band pass filter. Brought back ancient memories!

Figure 1 is similar info from the 2013 ARRL Handbook, page 11.22, Figure 11.46.

James Lynes

PLASTIC PARTS CAN BE MADE ECONOMICALLY FOR LIMITED QUANTITY APPLICATIONS

Clickfold Plastics utilizes a process that takes flat 1/8th inch plastic sheet and fabricates it into useful parts and enclosures. Since there is no hard tooling involved, the unique thermo-forming operation allows for fast, highly repeatable distortion-free part forming using 3D driven high speed CNC routers.

Franklin Huggins needed a limited supply of plastic parts as accessories for radio controlled model racing boats. An airfoil was developed to increase the speed and stability for the model boats. Injection molding the part required expensive tooling, was not timely, and uneconomical for the low volume of units initially required. So, Franklin came to Clickfold Plastics in Charlotte, NC for samples to test.

Within a week, prototypes were produced for testing on the boats. Boat speeds increased by 10% with improved stability. Two phases of prototypes were made to obtain improved performance, and within one month the new parts were on the market to model racing boat parts distributors.

Franklin Huggins credits Clickfold Plastics for its fast delivery of prototypes, quickly developing several designs of the product without expensive tooling. The CNC programming and first part production cost less than $1,000.

Check out www.clickfoldplastics.com for details on ordering custom plastic parts for your special projects.
Chances are you use one of these technologies.

Most individuals get their high speed broadband Internet access via a digital subscriber line (DSL) or a cable TV company modem.

While some direct fiber and wireless broadband systems are deployed, the DSL and cable TV connections dominate in over 80% of homes and businesses. Both of these systems have been around for over a decade, but recent upgrades have made them faster. This article shows the technical details of each system while highlighting the most recent new features and specifications.

**Digital Subscriber Line**

DSL broadband systems use the existing public switched telephone network (PSTN) wiring. Also known as the plain old telephone system (POTS), this vast network throughout the world uses unshielded twisted pair (UTP) cabling as the transmission medium. This network was designed to carry voice signals in the 300 to 3,400 Hz range, and few ever imagined that it would eventually carry high speed digital data. Yet today, it is possible to transmit digital data at rates up to about 50 Mbps under the right conditions. New standards push for up to 1 Gbps data rates.

UTP is a great medium for voice. It is simple, low cost, rugged, and very flexible. The typical length of a connection from the telephone central office to a home or office phone is in the 9,000 to 18,000 foot range. Therefore, this connection suffers from very high attenuation and it is vulnerable to noise pickup from many sources. Despite this limitation, electronic technology has overcome these problems. Attenuation factors of up to 90 dB on an 18,000 foot line really reduce signal strength, but special amplifiers and frequency compensation methods successfully mitigate this problem. This has made DSL the most widely used high speed Internet connection worldwide.

Early developments in dial-up modems demonstrated that this medium could carry digital data. Using FSK and PSK techniques, modems were developed to carry data at rates from 1200 to 9600 b/s. Using quadrature amplitude modulation (QAM) – a mix of phase and amplitude modulation with a center frequency in the 1,700 Hz range – modems were created that were capable of delivering data at a rate about 53 kbps. Such modems were very popular for a while until DSL was created.

Several variants of DSL were developed over the years, but the most popular and widely implemented is called asymmetrical DSL or ADSL. The asymmetrical designation refers to the fact that the download and upload speeds are different. Download from source to consumer is much faster than the upload speed from consumer to destination. There are multiple versions of ADSL and an even faster version called very high speed DSL (VDSL) that is used for carrying video.

The basic structure of ADSL uses a form of orthogonal frequency division multiplexing (OFDM) called discrete multitone (DMT). Using QAM, DMT transmits data divided amongst many 4.3125 kHz wide subcarriers. These subcarriers called bins or channels are distributed along the cable bandwidth as shown in Figure 1. (Note: The diagram is not to scale.) The 0 to 4 kHz spectrum is reserved for analog voice calls. The spectrum from 28.875 kHz to 138.8 kHz is reserved for 25 bins used in upstream transmission. The spectrum from 138.8 kHz to 1.104 MHz holds 256 bins for downstream transmission.

Depending on the type of modulation in each bin, line length, and noise conditions, a data rate up
to 60 kb/s per bin is possible. The
typical maximum realistic data rate
for a 9,000 foot line is 6.144 Mb/s
downstream and 576 kb/s upstream.
For a maximum 18,000 foot line, the
typical maximum rate is 1.536 Mb/s
downstream and 384 kb/s upstream.
Keep in mind that the rates vary
considerably with line length and line
conditions.

DMT works by dividing the high
speed serial data up into many slower
streams that are transmitted in
parallel or concurrently, one stream
per subcarrier. Using QAM, the rate
can be as high as 60 kb/s per bin.
The key to DMT operation and
practicality is the use of DSP to
implement the modulation scheme
using the fast Fourier transform (FFT)
for reception and the inverse FFT for
transmission. These mathematical
operations are implemented in
software on a digital signal processor
(DSP), or in a field programmable
gate array (FPGA), or other system on
a chip (SoC). An FFT takes a digitized
analog signal and breaks it down into
the individual frequency components
present. This is used in the receiver.
An inverse FFT assembles the binary
words into a sequence that will
generate a modulated signal.

While the initial version of ADSL
is capable of a maximum of 8 Mbps
downstream and 1.3 Mbps upstream,
other versions have been
developed to improve on that.
A newer version called ADSL2
extends the bandwidth on the
UTP cable to 2.2 MHz (Figure 1).
This enables a maximum
data rate of 12 Mbps downstream
and 1.3 Mbps upstream. An even
newer version has extended this to
24 Mbps downstream and 3.3 Mbps
upstream. Again, keep in mind that
the cable length is the limiting factor.

To keep cable lengths short, the
telephone carriers have installed an
interface unit called a digital
subscriber line access multiplexer
(DSLAM) in neighborhoods where
the UTP from subscriber lines
terminate. This helps ensure that lines
will not run more than about 5,000
feet. The DSLAMs connect back to
the central office with fiber optic
cable.

As mentioned previously, the
even newer very high speed version
of DSL called VDSL was developed
to carry digital video. It extends the
frequency range on the cable to 8.8
MHz. This allows up to 2,048 bins
producing data rates to 50 Mbps.
Extending the bandwidth to 12, 17,
or 30 MHz and increasing the
number of bins further boosts data
rates to 100 Mbps or more. To get
these speeds, cable lengths are
limited to the 1,000 to 3,000 foot
range. Careful placement of local
DSLAMs permits this.

A newer version called VDSL2
further boosts data rates. Using up to
30 MHz bandwidth on the cable and
doubling the size to 8.625 kHz
for higher level modulation methods,
data rates can be up to 200 Mbps.
The technology that makes this
possible is known as vectoring.
Vectoring is a technique for
overcoming the crosstalk problem on
the cable. Most telephone cable
bundled multiple UTP lines together.
The close capacitive and inductive
coupling causes signals on one cable
introduce interference on adjacent
lines. In other words, each line

![Figure 2](image-url). This is the DSL frequency spectrum on a single UTP cable showing the bandwidth used for the various versions of DSL. (Not to scale.)
interferes with the others. The induced signals are small and are generally not a problem at low data rates as long as the lines are kept short. To achieve the higher data rates, the crosstalk is the limiting factor. Vectoring is an advanced signal processing method that helps cancel crosstalk.

You may have encountered VDSL or VDSL2 if you have used AT&T's U-verse system. It was developed to carry multiple TV channels to compete with local cable TV companies. It also provides Internet connection data rates to 50 Mbps.

That's not all, however. A new and forthcoming version of DSL is known as G.fast. Like the other versions of DSL, it is a standard of the International Telecommunications Union (ITU). It is designed to achieve up to 1 Gbps over copper twisted pair. This is done by extending the bandwidth on the cable to 106 MHz or 212 MHz. Then, using improved methods of vectoring and a technique called channel bonding - the 1 Gbps level can be achieved. Channel bonding is the use of two or more UTP lines in parallel. The main limitation is the range which is restricted to 250 meters.

G.fast has not yet been implemented, but could be popular as it can deliver data speeds normally associated with fiber optic cable but at a fraction of the cost. It could be used to distribute video and Internet access in multi-tenant buildings or as backhaul for various wireless services.

**CABLE TV INTERNET**

Cable TV that is about 60+ years old was originally developed to deliver analog TV. Today, it delivers digital TV, voice over Internet Protocol (VoIP) digital phone service, and Internet access. Over 50% of US households have cable TV Internet service.

Local cable systems vary considerably but all are of the same basic design using a combination of fiber optic cable and coax cable known as a hybrid fiber coax (HFC) system. The basic idea is illustrated in Figure 2. The cable headend - also known as the Cable Modem Termination System (CMTS) - is where the system connects to the various video sources by satellite, fiber, or antenna. It also provides a connection to the Internet core network and the PSTN for voice. It then distributes the video and other services via a fiber optic cable to connection nodes located in the neighborhoods where services are to be distributed. The nodes provide optical to coax and coax to fiber conversions, as well as amplification. The signals are then sent to the homes by way of coax cable, usually RG-6/U. Typically, an optical node can handle up to about 2,000 homes. In the home, the coax connects to a cable modem for Internet access and VoIP service, or a cable TV set top box.

Most HFC systems organize the video and data by channels. These 6 MHz wide channels are distributed from 50 MHz to about 700 MHz. There used to be one analog TV station per 6 MHz channel. Today, thanks to digital compression techniques - two or three digital video signals can occupy one 6 MHz wide channel. The coax cable bandwidth used is commonly at least 750 MHz, or in some systems up to

![Figure 2: This is a simplified block diagram of the typical hybrid fiber coax (HFC) cable systems widely used in the US.](image-url)
860 MHz or 1 GHz. Many channels can be accommodated. Most of the spectrum is for downstream delivery of video or Internet downloads. However, the spectrum from about 5 MHz to 40 MHz or 65 MHz in some systems is used for upstream communications from the users.

HFC cable systems use a set of standards developed by CableLabs. Known as the Data Over Cable Service Interface Specification (DOCSIS), this standard defines the spectrum assignments, modulation types, error correction, security, and other protocol related specifications. The current version is DOCSIS 3.0. Downstream modulation is either 64QAM or 256QAM, providing a data rate up to 38 Mbps. Higher rates can be achieved by channel bonding – the use of two or more channels to transmit data in each channel concurrently. Upstream modulation is typically QPSK, but systems allow faster 16QAM or up to 128QAM.

CableLabs recently announced DOCSIS 3.1 – a major upgrade to provide even higher data rates. The 3.1 version uses orthogonal frequency division multiplexing (OFDM) and an improved error correction method called low density parity check (LDPC). The OFDM format eliminates the 6 MHz channels and replaces them with channels that can be from 24 MHz to 192 MHz wide using OFDM subcarriers that are in the 20 kHz to 50 kHz range. Using modulation from 256QAM to 4096QAM, data speeds can easily reach the peak of 10 Gbps downstream and 1 Gbps upstream. That is certainly fast enough to compete with fiber to the home (FTTH), fiber optical systems from Verizon (FiOS), Google, or others.

DOCSIS 3.1 has not been implemented yet. It will take time for cable companies to make the necessary hardware and software changes. Look for these faster systems in 2015 and beyond. NV
The Steampunk Adventurer’s Guide
by Thomas Willeford

Steampunk stalwart Thomas Willeford cordially invites you on an adventure—one in which you get to build ingenious devices of your own! Lavishly illustrated by award-winning cartoonist Phil Foglio, The Steampunk Adventurer’s Guide: Contraptions, Creations, and Curiosities Anyone Can Make presents 10 intriguing projects ideal for makers of all ages and skill levels, woven into an epic tale of mystery and pursuit.

$25.00

Programming Arduino Next Steps: Going Further with Sketches
by Simon Monk

Arduino guru Simon Monk reveals advanced programming techniques for Arduino! Programming Arduino Next Steps: Going Further with Sketches is the must-have follow-up to Monk’s bestseller, Programming Arduino Getting Started with Sketches. Aimed at experienced programmers and hobbyists who have mastered the basics, this book takes you “under the hood” of the Arduino, revealing professional-level programming secrets.

$20.00

Beginner’s Guide to Programming the PIC32
by Thomas Kibalo

Thomas Kibalo, who has written many articles for Nuts & Volts Magazine delivers the beginner’s book many have been looking for: Beginner’s Guide to Programming the PIC32. Using the low-cost Microchip Microchip’s MX25XX128A microcontroller and the free to download version of the MPLAB X32 compiler, Kibalo takes you step by step through the fundamentals of programming the PIC32.

Reg Price $39.95 Sale Price $31.95

by Simon Monk

Fully updated throughout, this do-it-yourself guide shows you how to program and build fascinating projects with the Arduino Uno and Leonardo boards, and the Arduino 1.0 development environment. 30 Arduino Projects for the Evil Genius, Second Edition, gets you started right away with the simplified C programming you need to know, and demonstrates how to take advantage of the latest Arduino capabilities.

$25.00

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

Raspberry Pi Projects for the Evil Genius
by Donald Norris

This wickedly inventive guide shows you how to create all kinds of entertaining and practical projects with the Raspberry Pi operating system and programming environment. Each fun, inexpensive Evil Genius project includes a detailed list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions for easy assembly. The larger workbook-style layout makes following the step-by-step instructions a breeze.

$25.00

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 gadgets to cherished analog products of yesteryear.

$24.95

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 I 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
### Projects

<table>
<thead>
<tr>
<th>Product</th>
<th>Image</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
</table>
| Super Detector Circuit Set                   | ![Image](image1.png) | Pick a circuit! With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few. | Subscriber's Price $32.95  
Non-Subscriber's Price $35.95 |
| 3D LED Cube Kit                              | ![Image](image2.png) | 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 $69.95 |
| Radiation Monitor Alarm Kit                  | ![Image](image3.png) | This is an inexpensive surface-mount project that is good for beginners to start with. This kit has its own printed circuit board (PCB) which makes mounting the components easy. Plus, it comes in a pocket size shielded aluminum case measuring 3.5" in length and 1" in diameter. The on-off switch is a push button on the bottom. | Subscriber's Price $32.45  
Non-Subscriber's Price $33.95 |
| Geiger Counter Kit                           | ![Image](image4.png) | As seen in the March 2013 issue. 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 µroentgen/hr. The LND712 tube in our kit is capable of measuring alpha, beta, and gamma particles. Partial kits also available. | Subscriber's Price $145.95  
Non-Subscriber's Price $148.95 |
| Seismograph Kit                              | ![Image](image5.png) | 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 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 |
| Battery Marvel                               | ![Image](image6.png) | As seen in the November 2011 issue. Battery Marvel helps protect cars, trucks, motorcycles, boats, and any other 12V vehicles from sudden battery failure. This easy-to-build kit features a single LED that glows green, yellow, or red, indicating battery health at a glance. An extra loud piezo driver alerts you to any problems. Details, please visit our website. | Subscriber's Price $18.95  
Non-Subscriber's Price $19.95 |

### For Beginner Geeks!

- **The Learning Lab 1**
  - **Fundamental Concepts**
  - **Price:** $59.95
  - **Description:** 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.
  - **Additional Info:** For more info and a promotional video, please visit our webstore.

- **The Learning Lab 2**
  - **Basic Digital Concepts and Op-Amps**
  - **Price:** $49.95

- **The Learning Lab 3**
  - **Basic Electronics Oscillators and Amplifiers**
  - **Price:** $39.95
For the ElectroNet online, go to www.nutsvolts.com

click Electro-Net

USB - Add USB to your next project - It's easier than you might think!

USB-FIFO • USB-UART • USB/Microcontroller Boards
RFID Readers • Design/Manufacturing Services Available
Absolutely NO driver/software development required!

www.dlpdesign.com

Serial LCD & LED
www.melabs.com

INVEST in your BOT!

GET THE NUTS!VOLTS DISCOUNT!
Mention or enter coupon code NVNMZ142 and receive 10% off your order!

Hobby Engineering
Kits, Parts and Supplies
www.HobbyEngineering.com

Mobile App Now Available!

Authorized for Mobile

Tablets/Smartphones

iTunes

Download NOW On Your Favorite Mobile Device!

For Robot Builders

iOS • Android • Kindle Fire

NATIONAL RF, INC.
www.NationalRF.com
RF, DF, Wx/Sat/Ham Gear, Dials, Attenuators, Plug-in Cables

ALL ELECTRONICS
Electronic Parts and Supplies
www.allelectronics.com
Free 98 page catalog 1-800-828-0432
The Arduino Classroom

Arduino 101 — Chapter 2: Digital Output - LEDs

This article continues the Arduino 101 series started last month as part of a formal curriculum where you can learn computing and electronics basics much like you would if you took a standard semester-based introductory course. To help in this, I've started www.arduinoclassroom.com where each of these magazine articles will be presented along with laboratories, exercises, quizzes, and a forum. Now instead of being a passive reader, you get a chance to have some interaction and to talk back. You can find the Arduino 101 Projects Kit — a kit of components for the projects in this course — from the Nuts & Volts Webstore.

LEDs are Everywhere

LEDs (Light Emitting Diodes) have been used as indicator lights for decades in things like alarm clocks and entertainment systems, but recently they have started taking over many general lighting tasks since they are durable and very energy efficient. This month, you will learn how to design circuits using LEDs and how to use LEDs with Arduino software to indicate events to people using your system.

Figure 2 shows the four...
LEDs on the Arduino. The power LED indicates that the board has power (this LED is the first and most fundamental debugging indication — if something seems amiss, first make sure the power LED is on). The TX and RX LEDs indicate serial communications traffic with the TX and RX LEDs blinking when transmitting and receiving, respectively. This provides another very useful diagnostic tool which helps you see if the board is communicating and in which direction the communication is going.

The pin 13 LED is connected to the Arduino digital I/O pin 13 and can be used for a variety of software tests without having to add any external hardware to see the software in action — our first program in Chapter 1 blinked this LED to indicate that the program was working.

The LEDs on the Arduino are SMD (Surface-Mount Devices) like the one shown in Figure 3. While they do not look like the LED in Figure 1, the difference is in the size of the packaging. The active circuit — the silicon die — is about the same size in both packages. The limits to how small the LED is made is not the size of the semiconductor die, but the surrounding casing and connections which must be large enough to be handled by people and machinery.

How an LED Works

The name says it all: light emitting diode. A diode is an electronic device that only allows current to flow in one direction. It acts as a sort of valve that prevents the current from backing up or flowing the wrong way. [We will learn more about diodes in a later chapter.] A diode can be made from semiconductors — the same silicon-based material that computer chips are made from. Some of these materials configured as diodes emit light when a current passes through them. The color of the light depends on the materials used. The brightness of an LED is proportional to the electricity flowing through it. However, if you provide too much electricity, the LED heats up and burns out.

Since too much electrical current will cause an LED to burn out, we need a way to control the flow of electricity to provide the maximum light without risking destroying the LED. Electrical flow is known as current. We use a component called a resistor that resists current flow to an acceptable level. Resistors have a given resistance value measured in units called ohms which you may also see expressed with the Greek character omega: Ω. To control the current through our LED in this lab, we will use a resistor with a value of 1,000Ω.

What is a Resistor?

Materials with low resistance are known as conductors, while materials with high resistance are known as insulators. Electrons move easily in some things such as copper wire, but are stopped cold by things such as glass. Copper has very low resistance. The jumper wires in the Projects Kit have near zero ohms resistance (about one ohm per 62 feet), while glass can have millions of ohms of resistance.

Electronic components often have symbols associated with them that are used in schematics which are drawings of an electronic circuit. Figure 5 shows a schematic symbol for resistors next to a Fritzing drawing of a resistor showing the color stripes. [Fritzing is an open source electronic design program for drawing breadboards, schematics, and printed circuit boards (PCBs).]

The ohm value of resistors is shown by the color of the stripes around the resistor; refer to Figure 6. Notice
that the stripes are bunched toward one end; this is the end you start counting from.

By referring to a resistor values card (like the one in Figure 7), we see that the first band is brown and represents a one; the second band is black for zero; the third band is red for two zeros. So, we read this as a 1,000Ω resistor. The fourth band is gold for ±5% tolerance. This means the actual resistance is 1,000 plus or minus 5% of 1,000, which is 50Ω. So, we have a tolerance from 950Ω to 1,050Ω which is good enough for our purposes.

The LED Component

In Figure 8, we see the schematic symbol and Fritzing illustration for an LED.

The schematic symbol has a triangle and a bar that shows the direction that the diode allows current to flow; the two arrows indicate that this particular diode emits light. Because this component is directional with respect to current, there is a positive side in which the current enters and a negative side in which the current exits. The positive side is called the anode and is sometimes represented with a + (plus sign). The negative side is called a cathode and is represented by - (negative sign).

Building the Circuit

What is a Circuit?

We get electricity to do useful work by channeling it from devices that produce electric
force (like generators and batteries) through
devices that do electric work (like lights and
motors), then back to the device that
created the force. That last part is critical.
Circuit is just a fancy way of saying ‘circle.’
Electricity must complete a circle to do
useful work.

Figure 9 shows arrows marking the
direction of conventional current from the
higher voltage side of a nine volt battery
(the positive terminal) through a resistor
and an LED, then back around to the lower
voltage terminal of the battery.

You have probably seen really complex
circuits on PCBs or as schematics, but no
matter how complex it looks it can be
simplified to one part producing the force
as a current, one part using that force to
do work, and the electrical connection
between them.

Wonder why we have to add
‘conventional’ to current? Well, that’s
because Benjamin Franklin himself guessed
wrong when he said that charge is carried
by positive particles that flow from the +
side of a circuit to the - side. It was much
later that folks came to realize that the
actual charge carrier is an electron — which
has a negative charge. Of course by then,
textbooks all had current going from + to -,
so we are stuck with this backwards
concept.

Schematic Symbols for 5V and GND

Current flows from the higher voltage
[anode (+)] to the lower voltage. In
Figure 10, we see the schematic symbols
for the current source and destination. The source shown
[anode (+)] is five volts and the destination is zero volts
[cathode (-)] — also known as ground (GND).

We will build our circuits on a Arduino mini
breadboard shield (included in the Arduino 101 Projects
Kit). First, let’s see how a breadboard works.

How a Breadboard Works

In the good old days, electronics experimenters would
build prototypes by nailing components to an actual
wooden breadboard and then soldering wire between
connection points. Today’s solderless breadboards are
made of plastic blocks with holes on 0.1 inch centers that
allow you to insert jumper wires into hidden clips below
the holes.

Integrated circuits (ICs) are made of tiny blocks of
silicon that are much too small and delicate to be used by
hand. Early on, they were encapsulated in a special
package like the one shown in Figure 11. This DIP (Dual
Inline Package) packaging has the IC silicon encapsulated
in a block of epoxy with the pins directed downward.
These pins are usually lined up 0.1 inches apart on the
side of the package, with the two lines often 0.3 inches
apart.

Breadboards were designed to facilitate prototyping
with these packages so that these DIP ICs could be
plugged into the breadboard with the pins lined up on
either side of the trough in the middle of the breadboard
(shown in Figure 12). The vertical rows of sockets shown
above and below the IC are electrically connected and
allow components to be added to the prototype to make
temporary circuit contacts.

Figure 13 shows the top and bottom of a solderless
breadboard (the bottom has the foam tape stripped off to show the connections). **Figure 14** shows the clips pulled out. **Figure 15** shows how a clip grabs a wire, and **Figure 16** shows a cutaway drawing with an LED, 1K Ω resistor, and a jumper wire all connected such that if you have +5 volts in the upper + channel and GND in the lower – channel, the LED should light up.

**Our Mini Breadboard**

A miniature breadboard shown in **Figure 17**. **Figure 18** shows that this board has 17 columns of sockets with two sets of five connected sockets – one each above and below the central trough. This provides 170 tie-points for connecting circuits together.
The Mini Breadboard Shield Mounted on an Arduino

Our electronics and computing learning platform throughout this series has a mini breadboard shield mounted on an Arduino as shown in Figure 19. This platform provides many useful and convenient features that you will learn about in upcoming lab exercises.

Lab 1: Assemble Your Arduino Mini Breadboard Shield.

**Required tools:**

- 1 Arduino
- 1 Proto shield

**Estimated time for this lab:** Five minutes

**Check off when complete:**

- Place the shield with the long legs above the mating Arduino header sockets as shown in Figure 20.
- Place the legs carefully into the sockets, making sure they align properly as shown in Figure 21. You may have to use your fingernails to carefully align the pins while gently applying pressure.

- Figure 22 shows the legs pushed in all the way. Note there is about 1/16" space between the bottom of the shield and the top of the Arduino sockets.
Lab 2: How to Use an LED – Analog.

**Required tools:**
1 Arduino
2 Proto shield

**Estimated time for this lab:** 20 minutes

**Parts Required:**
1 Red LED
1 1,000Ω resistor
2 Jumper wires

**Check off when complete:**
- CAUTION: MAKE SURE POWER IS OFF BEFORE YOU BUILD A CIRCUIT! Check that the Arduino USB is unplugged and that you do not have a battery plugged into the power connector.
- Plug your mini breadboard shield into your Arduino as described in Lab 1.
- Plug an LED into the breadboard as shown in **Figure 23**, Make sure that the long leg is towards the bottom of the board as shown. Notice that the illustration shows the short leg connected to the column of five sockets above the LED and that the long leg connects to the column of five sockets below the LED.
- Plug a 1,000Ω resistor (brown, black, red bands) as shown in **Figure 24** and note the connection to the five sockets in the two columns — one unconnected and the other connected to the long leg of the LED.
- Attach a wire between the column of five sockets on the LED short leg to GND (ground), and then connect the column of five sockets on the resistor to 5V (five volts) as shown in **Figure 25**.
- Provide power to your system by either plugging the Arduino USB into a PC (**Figure 26**) or plugging a nine volt battery into the power jack (**Figure 27**). When you provide power, the LED should light up.

In **Figure 28**, you see a schematic on the left of the LED circuit on the right. At the top, you see a symbol with 5V above it. That symbol is for the input of five volts into the circuit. At the bottom, you can see three progressively smaller bars that are the schematic symbol for ground, which is zero volts. Since current flows from a higher voltage to a lower voltage, here it will run from 5V to 0V (ground), and the LED will light up.

Notice that the current in the schematic image flows from the top to the bottom, but in the proto shield...
We can also illustrate the current flows from the bottom to the top. These two images are ‘upside down’ relative to each other. This is just an artifact due to the traditional way schematics are drawn — usually with the higher voltage toward the top and the lower voltage or ground toward the bottom. The ‘flipped’ illustration is due to the 5V being at the ‘bottom’ of the image on the proto shield. This may be momentarily confusing, but it is good to keep in mind that current flows from high to low voltage and has nothing to do with the physical orientation of the device.

**What Happens if We Reverse the Current?**

In Figure 28, we have the five volts connected such that the current runs from the five volts through the resistor and the LED to ground. What happens if we reverse the connection and place the five volts at the bottom and the ground at the top of the schematic as shown in Figure 29?

**FIGURE 28: LED + resistor schematic.**

- Reverse the 5V and GND wires as shown in Figure 29. What happens?

- Why didn’t the LED light up? It is because an LED is a diode symbol. Notice that we had to bend the lines in the schematic to put the GND at the top and the 5V at the bottom. This is because (as mentioned previously) it is conventional to put the higher voltage symbol above the ground symbol.

**FIGURE 29: Reversing 5V and GND.**
Estimated time for this lab: 30 minutes

Parts Required:
1 Red LEDs
1 1,000Ω resistor
2 Jumper wires

Check off when complete:
- Make sure the power is off before building the circuit.
- Hook up the wire to pin 12 and ground as shown in Figure 30.
- Plug the USB cable into the Arduino.
- Open the Arduino IDE and File/Examples /Basic/Blink like you did for Lab 2.

- Change the code to add the pin 12 controlled LED as follows:

```cpp
void setup() {
  // Initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);  // Use pin 12 to blink LED
  // on breadboard
}

void loop() {
  digitalWrite(13, HIGH);  // Set the LED on
  digitalWrite(12, HIGH);  // Set the LED on
  delay(1000);             // Wait for a second
  digitalWrite(13, LOW);   // Set the LED off
  digitalWrite(12, LOW);   // Set the LED off
  delay(1000);             // Wait for a second
}
```

- Note that the `delay(1000)` causes a 1,000 millisecond (one second) delay.
- Following the methods you learned last month, verify and upload the program to your Arduino.
- Does the LED on the breadboard blink?
  - Yes: Great! You are ready to move on.
  - No: Looks like you'll need to do some debugging.

Does the pin 13 LED on the Arduino blink?
- No: You've probably wired the circuit wrong on the breadboard. Carefully check your wiring.
- No: You've probably entered something wrong in the code or not uploaded it properly. Carefully check that the code is exactly as shown and try verifying and uploading it again.

Lab 3: How to Use an LED — Digital.

Now that we know how to light up an LED with current supplied by our five volt source, let's apply that to using an Arduino digital output pin to supply five volts or zero volts to turn the LED on and off. We saw in Chapter 1 Lab 4 last month how to write and run a simple program that blinks the pin 13 LED. In this lab, we will put an LED on the breadboard and make it blink using pin 12.

Required tools:
1 Arduino
1 Proto shield

Next Month

In Arduino 101 Chapter 3, we will begin learning how an Arduino program works.
Traffic Detector

I live in the country on a side road that is one of 100 feet of visibility. Most of my trips involve turning left, thus crossing oncoming traffic. A car coming around the curve at 60 MPH gives me a little over one second reaction time when it first becomes visible.

At night, I can see oncoming headlights reflecting off a guardrail which gives me plenty of warning. During the day, I roll down my window and listen. However, this is not the best method with my aging hearing. I'm looking for a clever electronic solution to detect approaching cars and provide an earlier warning.

There is a pole on the other side of the road about 75 feet away that could be used to mount a device to bounce a signal off. I'm new to electronics but if "steered" in a useful direction, I can do the research and make it work. Any ideas?

Curtis Erpelding
Port Orchard, WA

Laser Cutter To 3D Printer

I want to move from a mill to a 3D printer to fabricate parts for my projects. As far as printing materials go, I've heard that regular plastic is expensive and print quality is poor, and that the PLA alternative is brittle and heavy. What's the best printing material out there? Are there better choices?

Alec Litmar
via email

Accelerometer Versus Gyroscope

I'm working on a robotics project that requires navigation and I'm trying to decide between an accelerometer and a gyroscope chip. Accelerometers are cheaper, but I've heard not as accurate. Is this true? When should I use one over the other?

They are two different sensors. A gyro measures rotation about an axis. An accelerometer measures acceleration in a given direction. To create a true inertial navigation system, one needs to accumulate both acceleration and rotation information in all three axes (x, y, and z). An accelerometer can be used to make a rotation measurement. It will not be as accurate as traditional rotation sensors.

Nevertheless, for making "guidance" sensors as opposed to "navigation" sensors, there are many (relatively inexpensive) three-axis monolithic sensors based on MEMS accelerometers. Guidance is used to provide orientation feedback whereas navigation sensor is used to place location after so many seconds, hours, or days.

For further study on the differences, look at the physics behind a ring laser or fiber optic gyro (it only takes a few minutes to figure out what is going on). The difference in what the sensors are measuring becomes apparent (http://en.wikipedia.org/wiki/Ring_laser_gyroscope).

D Ferguson
Austin, TX

Battery Dilemma

I use one of those headlamps a lot and replacing the batteries is getting expensive. Are there affordable rechargeable batteries that provide the same burn time as single use batteries?

Are there ever! Rechargeable 3.2V Lithium Iron Phosphate (LiFePO4) cells are surely the answer. They're powerful but safe, very lightweight, and available in both AA and AAA sizes at ~US$5 each.

Perhaps best of all is that one will replace two 1.5V alkalines (or three NiMHs!).

Simple placeholder dummy cells can put out the battery bay. I've taken to using these in my Canon digital cameras to great effect.

Stan Swan
Wellington, NZ

Explosive Potential

I want to encapsulate a circuit board with a small Li-Ion battery in epoxy for an outdoor weatherproof project. Does anyone have experience encapsulating batteries? Is this a good idea?

Epoxy encapsulation is pretty drastic. Why not simply seal things in a plastic screw top container?! If the explosive nature of Li-Ion cells is the concern, then you may be better off using the far more tolerant 3.2V rechargeable LiFePO4 (Lithium Iron Phosphate) version.

D Ferguson
Austin, TX

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

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!
AA/AAA sized LiFePO4 cells sell for ~US$5 each and are stated as good for 1,000s of charge/discharge cycles too. See www.instructables.com/id/Simple-AAA-LiFePo4-cell-powered-project-in-a-parti/.

Stan Swan
Wellington, NZ

[#1141 - January 2014]
Amplifier Hum

I picked up a "hum blocker" for my guitar amp, only to discover that all it does is disconnect the ground from a three-prong outlet.

It seems to work, but is it safe to use?

#1 Safe is a relative matter of degree. The safety ground wire in a standard three-wire AC cord is there for a reason. However, the NEC does allow for devices that do not have that safety ground connection. Many electric and electronic devices are made without it.

The safety ground connection is there to insure that the fuse does blow or the circuit breaker does trip if a conductive outer case of the device becomes electrified. The current that flows from the hot wire to the safety ground blows the fuse or trips the breaker and then there is no danger of shock.

Devices that do not have that safety ground usually employ a double insulation method to prevent such a short to the outer case or to the user. Thus, if there is internal insulation AND the outer case is an insulating material like plastic, then TWO different insulators would have to fail for the user to get shocked.

Older devices — like early electric lamps — were made with metal outer cases and often the electric wires inside them were only a small fraction of an inch from that conductive case. Older electric cords were insulated with real rubber which deteriorated over time and became "frayed." The cardboard insulators in lamp sockets literally fell apart after some years of service, usually with over-sized bulbs in the lamp (rated at 40W so the user put a 60W or a 100W for more light). Etc., etc., etc.

It is not possible to tell if a particular device (like your amplifier) is safe without examining it. If it was commercially manufactured in the US or imported properly, then chances are it is okay. Things like the UL (Underwriter's Laboratories) tag are supposed to tell the consumer that it has been inspected and found to comply with the code. If it is a kit or was made in someone's kitchen or garage, then it may or may not be okay.

Hum like this is usually a result of what is called a "ground loop." Both the signal cable between the audio devices and the power cables to them are grounded, and they form two different and distinct ground paths between them. If there is any 60 Hz current flowing in either of them, then the circuit/signal ground planes in each of those devices will be at a different AC potential and the hum is introduced into the signal.

One way of eliminating this situation is to cut the safety ground on one or both of the devices, but this eliminates the safety feature of that connection. A better way to eliminate this situation is to use a balanced audio line between the devices and to connect (ground) the shield only at the source end of that line. In this way, there is only one ground connection between them via the AC power. Many times this is used in professional audio installations where long lines must be run. This may not be possible on amateur equipment where single ended audio lines are often run.

Another solution is to use a differential input amplifier at the receiving end of the audio line. This can be accomplished with a transformer or with actual active circuitry. With a transformer, the center lead and the shield of the single ended cable are connected to the primary of the transformer so the shield is not connected to the chassis/signal ground of the amp.

Active circuitry with a balanced input is more complicated. I am designing such an amp at the present time to overcome a noise problem between my audio/visual equipment cabinet behind my chair and the computer on the desk in front of it. Due to the arrangement of the room, I had to run a 50 foot cable and it does pick up hum.

An audio transformer of sufficient low frequency response would cost a bundle. Since the audio is single ended (not balanced), this may be the only solution. I have searched and could not find a reasonably priced differential input audio amp for this application. Professional ones seem to start in the hundreds and go up from there.

Another factor that helps to overcome noise and hum is the impedance of the audio line. A lot of amateur equipment uses a high line impedance which may be okay for a few feet, but longer high impedance (10 kohms) lines will pick up noise and hum from the environment. Professional audio uses lower impedance lines to overcome this.

The old standard was 600 ohms, but modern equipment often has extremely low output impedance (one ohm or less) and a higher input impedance. This works in most situations. When long lines need lower termination impedance on the receiving end, a simple resistor can be connected across the line at that point.

Paul Alciatore
Beaumont, TX

#2 Some comments and concerns regarding your question. First, I am not clear if this is a repair or a retrofit. I would also presume this is a tube-type...
amplifier, since they were often built with point-to-point wiring and hum pickup was highly likely with so many signal wires running around the chassis.

I would presume that either this is a retrofit or the original noise filter cap was removed in the past. The fact that adding the new cap from line to ground reduced the 60 cycle hum (or 120, depending on the rectifier design), it seems that this might actually be a repair.

Or, the filter caps themselves are old and leaky, so you are adding them but not actually curing the problem. If the amp is old, it often needs to have the filter caps replaced and (sometimes) some interstage decoupling caps as well.

With that background, I caution that this involves principles of safety and if you are not experienced with working on mains circuits, this effort would be best left to a professional. Just because something works does not mean it is well-executed or robust.

Since guitar amps get moved, jostled, banged around, and are often not treated carefully, there is a chance your soldered connections of this component could break loose without your knowledge and you could have a real shock hazard on your hands.

Okay, assuming all that is well-appreciated, the type of cap you use is critical. This component is connecting one side of the AC mains to the accessible chassis. You cannot assume that a ground pin on a cord will save your life since you have no idea how well the ground-to-neutral impedance is controlled in the building's branch circuit you just plugged into.

Hence, the cap itself must be suitably rated to be connected from line to ground. Therefore, this must be a Y2 type of cap.

If you have no idea what a Y2 cap is, then read this link first: www.justradios.com/safetytips.html. There are many good manufacturers of Y2 caps (Vishay, KEMET, Panasonic, etc.). Here is a good selection from Mouser: www.mouser.com/new/Kemet-Electronics/KEMET_XY_Film_Caps.

Be sure that you do not rely on solder for mechanical support of live parts. Always wrap the cap leads snugly around the terminals it is connecting to and check to determine that it would tend to stay there if no solder was used. Solder is not meant for mechanical retention of massive parts. Once you feel it is well mounted, then use solder to make the electrical connection to the terminals.

Jon Kalfus
Red Bank, NJ

#1142 - January 2014]
Which Way Should I Go?

I'm new to electronics, recently retired, and in need of some direction. Should I spend my time learning about resistors, capacitors, and transistors, or start with an Arduino or other microcontroller?

#1 In answer to your question — Yes, you should learn about electronics fundamentals along with learning about microprocessors. You need to have an understanding of the basics, however, you do not need to get a degree in electrical engineering.

Your best bet is to check out your local library and see if they have any books dealing with electronics fundamentals. You may want to see if they have a copy of the ARRL Amateur Radio Handbook. They also have a book titled Understanding Basic Electronics which will help you get started.

I don't know whether or not you have your ham license but you may want to see if there is an amateur radio club in your area and get your self an "Elmer," which is ham radio talk for a mentor, as most hams that I know of are willing to pass on the knowledge they have acquired.

There are also several groups on Yahoo! devoted to beginners which are worth joining.

Craig Kielhofer
Des Moines, IA

#2 Starting learning electronics with discrete components (resistors, capacitors, diodes, etc.) or micros is a good question. Fortunately, you can choose to start with either and be successful due to modern kits and the Internet. If you are into "tinkering" programming in C language, microcontrollers would be a good place to start.

To begin in the microcontroller field, I would start with the BASIC Stamp. Parallax (www.parallax.com) has some good starter kits which supply the microcontroller, breadboard, and other components to perform the experiments they have well documented in the enclosed manual. After gaining proficiency with the Stamp, you can move up to the Arduino (http://makerzone.mathworks.com/arduino) or the Propeller (www.parallax.com). Nuts & Volts has projects every month on microcontroller type systems which can also be used to further your self-training.

If you don't feel comfortable with programming, learning the basics of electronics may be the place to start. Googling "electronics tutorials" will show you a number of free websites which offer electronics training for beginners. (Also, the Nuts & Volts Webstore sells books and kits to help you learn electronics.)

Electronics is a neat field to study. It is not easy. It is not for everyone. However, for anyone who is looking for a challenge, electronics is the place to be. Whether you start with basic components or microcontrollers, you will eventually see the need to learn about the other field.

Microcontrollers use basic electronic components to function as a system, and often it is advantageous to replace a "kludge" of electronic components with a microcontroller app.

Tim Brown Ph.D EE, PE
via email

Got electronic issues and need help dealing? Let our readers help! Email your questions to forum@nutsvolts.com
# A D v e r t i s e r I N D E X

Find your favorite advertisers here!

## AMATEUR RADIO AND TV
- Ramsey Electronics ............... 82-83

## BATTERIES/CHARGERS
- Cunard Associates ................. 21

## BUYING ELECTRONIC SURPLUS
- Weirdstuff Warehouse ............ 21

## CCD CAMERAS/VIDEO
- Ramsey Electronics ............... 82-83

## CIRCUIT BOARDS
- AP Circuits ...................... 57
- Cunard Associates ............... 21
- Digilent ......................... 5
- Dimension Engineering .......... 6
- Express PCB ..................... 45
- Front Panel Express LLC ........ 36
- GHI Electronics ................ 45
- PCB Pool ........................ 29

## CNC ROUTER
- Next Wave Automation .......... 29

## COMPONENTS
- Eastern Voltage Research ........ 17

## COMPUTER
### Hardware
- Weirdstuff Warehouse ........... 21

### Microcontrollers / I/O Boards
- Abacom Technologies ............ 57
- C2 Robotics ...................... 44
- GHI Electronics ................ 45

### MISC./SURPLUS
- All Electronics Corp. ............ 63
- Front Panel Express LLC ........ 36
- Weirdstuff Warehouse .......... 21

## DESIGN/ENGINEERING/REPAIR SERVICES
- Express PCB ..................... 45
- Front Panel Express LLC ........ 36
- PCB Pool ........................ 29

## EDUCATION
- Command Productions .......... 7
- Digilent .......................... 5
- Next Wave Automation .......... 29
- NKC Electronics ................. 21
- Poscope .......................... 58
- UCount ........................... 29

## EMBEDDED TOOLS
- NetBumer .......................... 2

## ENCLOSURES
- Front Panel Express LLC ........ 36

## EVENTS
- Trenton Computer Show .......... 37

## KITS & PLANS
- Eastern Voltage Research ......... 17
- NetBurner .......................... 2
- NKC Electronics ................. 21
- QKITS ............................. 21
- Ramsey Electronics .............. 82-83
- UCount ............................ 29

## MOBILE RADIO MOUNTS
- E Tip Inc. ......................... 21

## PROGRAMMERS
- C2 Robotics ...................... 44
- microEngineering Labs ........ 28
- MikroElektronika ................. 3

## RF TRANSMITTERS/RECEIVERS
- Abacom Technologies ............ 57
- E Tip Inc. ......................... 21

## ROBOTICS
- Digilent .......................... 5
- GHI Electronics ................ 45
- Robot Power ...................... 21
- Sphero 2.0 ......................... Back Cover
- UCount ............................ 29

## SECURITY
- E Tip Inc. ......................... 21

## TEST EQUIPMENT
- Dimension Engineering .......... 6
- NKC Electronics ................. 21
- Poscope .......................... 58

## TOOLS
- C2 Robotics ...................... 44
- MikroElektronika ................. 3
- NetBurner .......................... 2
- Next Wave Automation .......... 29
- PanaVise .......................... 36
- Patco .............................. 29
- Poscope .......................... 58

## WIRE PROCESSING/STRIPPING
- Patco .............................. 29

---

Abacom Technologies ............ 57
All Electronics Corp. ............ 63
AP Circuits ........................ 57
C2 Robotics ....................... 44
Command Productions ............ 7
Cunard Associates ............... 21
Digilent ............................ 5
Dimension Engineering ............ 6
Eastern Voltage Research ....... 17
E Tip Inc. ........................... 21
Express PCB ....................... 45
Front Panel Express LLC ......... 36
GHI Electronics .................. 45
GHI Electronics .................. 45
microEngineering Labs ........ 28
MikroElektronika ................. 3
NetBurner .......................... 2
Next Wave Automation ............ 29
NKC Electronics .................. 21
Sphero 2.0 ......................... Back Cover
PanaVise .......................... 36
Patco .............................. 29
PCB Pool ........................... 29
Poscope ............................ 58
QKITS ............................... 21
Ramsey Electronics ............. 82-83
Robot Power ....................... 21
Technologic Systems ............. 35
Trenton Computer Show .......... 37
UCount ............................. 29
Weirdstuff Warehouse ............ 21

February 2014 NUTSIVOLTS 81
Valentine’s Day Goggles!

**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 beeps when an ECG waveform is displayed
- Simple and safe 9V battery operation!

February is the month for Valentine’s Day, and what a great time to think of your heart! Not how many times it’s been broken, but how many times it’s fallen head over heels in love, but how it actually works... and how it’s doing these days! Not only will building an ECG be a great learning project, you’ll get hands-on knowledge of the relationship between electrical activity and the heart rhythm. Each time the human heart beats, the heart muscle causes small electrical changes across your skin. By monitoring and amplifying these changes, the ECG will detect 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 ECG to astound your physician with your knowledge of ECG/ECG systems. Enjoy learning about the inner workings of the human heart, unlike, at the same time, covering the state-by-state 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.

**Electronic Love Tester**
- 10 LED love scale display!
- Audible love level sound!
- Great party fun!
- Heart shaped board!

This uniquely shaped “Love Tester” is the ultimate gag for any couple! Designed to check your love life, each partner holds one end of the tester PCB at the appropriate male and female touches pads. Then they romantically join hands and watch the results on the love meter. 10 green, yellow, and red LEDs act like a speedometer, and just like the love meter, it hits the top they flash, indicating you’re a red hot couple! 280 millisecond delay between flashings is the “love level.” Next time the party isn’t going anywhere, bring this out, it’s a hit! Compatible with all couples! Runs on a standard 9V battery (not included).

**LED Flashing Heart**
- 28 brilliant red LEDs!
- Unique dual heart design!
- Freestanding mount!

What a way to display your feelings to the one you love in your life! Get out your soldering iron and dazzle her with this unique double heart electronic display that you can say you built yourself!

28 brilliant red LEDs are formed into two separate heart designs on the heart shaped PCB board creating a flashing display that’s easy to assemble. Built-in battery holder means you don’t need the sweetheart frozen hand-held and perfect for a desk or table. Runs on a standard 9V battery (not included). Measures 2.4”x2.4”x1.2”.

**SMT LED Heart Display**
- Alternating flashing!
- 6 super bright SMT LEDs!
- Light and all about SMT!
- Definitely gets her attention!

This cute little kit gives you a distinctive red display using 6 Surface Mount (SMT) LEDs. The PCB board is in the shape of a red heart. The small size makes it perfect to be used as a badge or hanging pendant around your neck. Even better as an illuminated attention-getting heart to accompany a Valentine’s Day card! Makes a great SMT learning kit to bring you into the world of SMT technology, design, and hands on soldering and troubleshooting. Don’t worry, extra SMT parts are included just in case you lose in damage runs! Small CR2032/32button cell (not included). Measures 1.9” x 1.7” x 3/4”.

**Tickle-Stick Shockie**
The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And if that can’t resist a blinking light and an unbeatable switch? Great fun for your desk, “Hey, I told you not to touch!” Runs on 3.4V.

**Tickle-Stick Shockie**

**Check out our virtual on-line catalog that includes over 400 neat hobby kit and electronic products, with easy-to-find search, print, and share features, and frequently updated. Visit www.ramseycatalog.com!**

**Follow Us and Save $50!**
Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!
Stereo Audio Gain Controller

- Stereo audio processing while preserving audio dynamics!
- True stereo control keeps virtual source location intact!
- Auto-bypass restores original levels when power is turned off!

The SG1C is one of our latest innovations and provides a great solution to the age-old problem: how can we easily control inconsistent audio levels without negatively affecting the dynamics of the audio signal? The SG1C circuit implements a principle known as the "Platform Gain Principle," which was originally developed by CBS Labs (what we now know as the National, SBE, and AR members). Its full bi-directional communication with the external world using the USB port of your computer allows the SG1C to communicate with both Mac and Windows operating systems, making it ideal for use in a studio environment.

The SG1C is a stereo audio platform gain controller that can be used to adjust the output level in a studio environment. It allows you to adjust the output level in a studio environment and ensure that the audio is balanced and consistent. The SG1C is a great solution for those who want to ensure that their audio is always at the perfect level.

8-Channel Remote Ethernet Controller

Now you can easily control and monitor up to 8 separate circuits via the standard Ethernet network in your home or office. The connection wise it couldn't be simpler. The controller functions as an IP based web server, so it can be controlled by any internet browser that can reach your network. There are no drivers or proprietary software required, just access the controller like any web page from your PC, laptop, or even your smartphone!

Security is assured allowing up to 4 separate user credentials. The controller can be set to a specific static IP within your network subnet or can be set to DHCP (auto negotiate). The controller can even be programmed to send you an email on notify and confirm power up and status changes!

To simplify the connection of your equipment to the controller, 8 separate isolated relay output ports are provided! This gives you internet or network control of up to 8 separate functions. No need to add a relay or bury the interface! The applications are endless! From something as simple as turning on and monitoring lights at your house to controlling power via a remote switch, it can be used to advanced closed access to any advanced electronic of your electronic gadgets, radio equipment, or even your garage door!

Each relay contact is rated at 12A at 30VDC or 16A at 230VAC. Each of the 8 channels has built-in timer and scheduler programs for day, weekend, working days, every day, and every day except Sunday. Relay control functions are programmable for on, off, or pulse (1-999 seconds, 1-99 minutes, or 1-99 hours). In addition to control functions, the web interface also displays and confirms the status of each channel. Each channel can be customized to your specific function name. The controller operates on 12VDC or 120VAC at 500mA or our new AC121 global 12VDC switching power supply below. Factory assembled, tested, and ready to go! Even includes a Cat-5 cable!

VM201 8-Channel Remote Ethernet Controller, Factory Assembled & Tested $169.95

GET THE IN-STORES DISCOUNT!
Mention or enter the coupon code NVRM2142 and receive 10% off your order!
GET YOUR NERD ON

FASTER
Sphero 2.0 rolls at speeds of up to 7 feet per second. That's only slightly slower than a Lamborghini. Trust us, this thing hauls.

BRIGHTER
With new multicolor LEDs, Sphero 2.0 is 3x brighter. Put your sunglasses on so the awesomeness doesn't blind you.

SMARTER
Every aspect of Sphero 2.0 has been reprogrammed. It's like Einstein and C-3PO had a mutant baby.

PROGRAMMABLE
With iOS and Android SDKs, and apps like MacroLab, you can unleash Sphero's inner robot and program like a pro. Time to geek out.

25+ FREE APPS
Drive Sphero from your smartphone or tablet, turn your living room into a video game with augmented reality, and more.

Buy now at gosphero.com/NV