■ PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

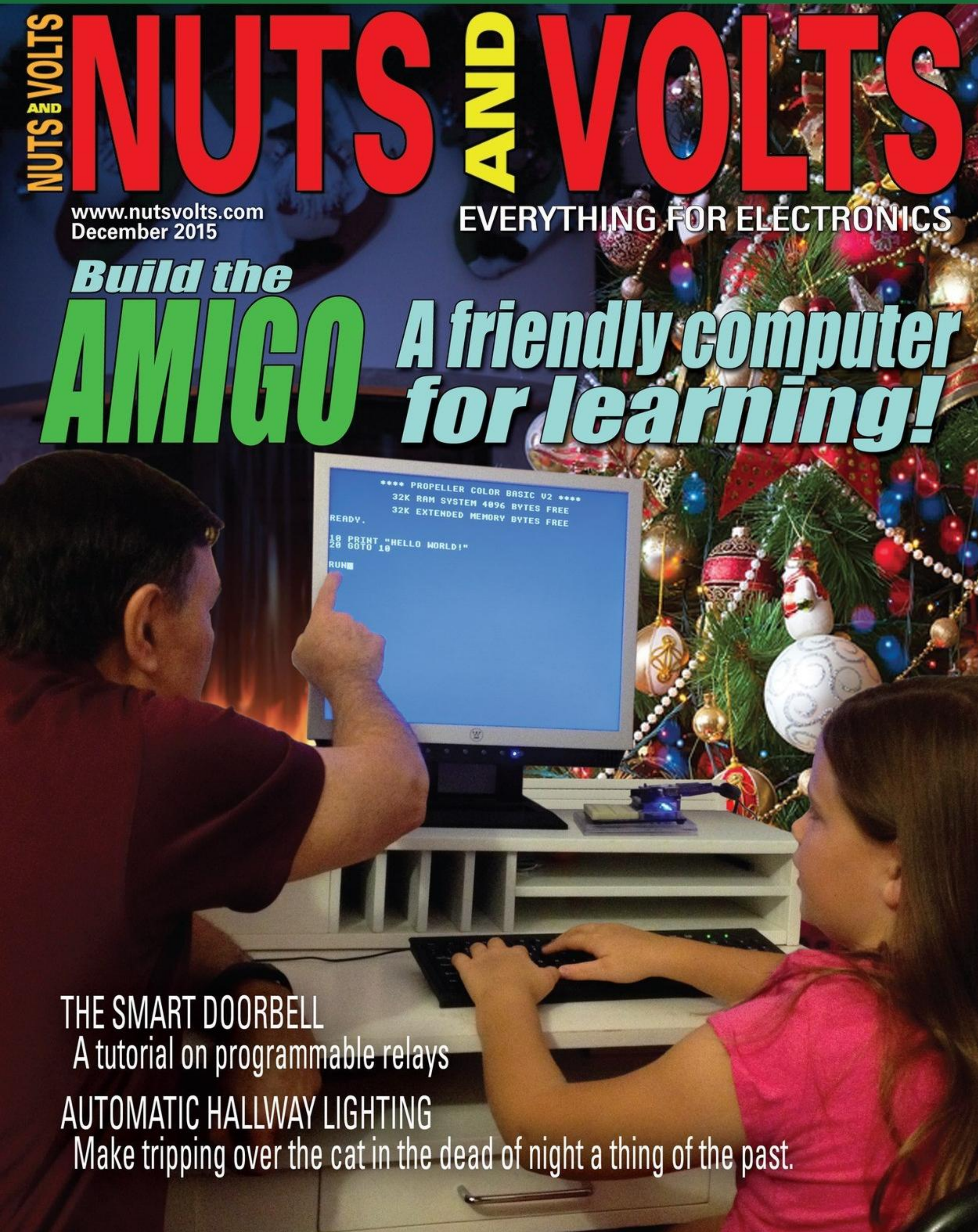
NUTS AND VOLTS

EVERYTHING FOR ELECTRONICS

www.nutsvolts.com
December 2015

Build the AMIGO

A friendly computer for learning!



THE SMART DOORBELL

A tutorial on programmable relays

AUTOMATIC HALLWAY LIGHTING

Make tripping over the cat in the dead of night a thing of the past.

STAR WARS™

DISNEY

THIS IS THE DROID YOU'RE LOOKING FOR

Meet BB-8™ - The app-enabled Droid whose movements and personality are as authentic as they are advanced. BB-8 responds when you speak, expresses different traits, and rolls under your control or on its own. The magic this Droid creates is unlike anything you've ever experienced. It's more than a toy – it's your companion.

BB-8™ APP-ENABLED DROID™

sphero.com

December 2015

Page 28

Departments

Columns

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

NUTS AND VOLTS

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

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

DEVELOPING PERSPECTIVES

by Bryan Bergeron, Editor

For Learning, Old School Can be the Best School

When I started out in electronics, my "junk box" of rescued parts from TVs, radios, and the like was the source of endless projects and test instruments. Armed with a few key texts — especially the *ARRL Handbook* and *Getting Started in Electronics* by Forrest Mims — just about anything was possible. Sure, my projects didn't win any beauty contests with labels made with a permanent marker and reused chassis with dozens of extra holes, but most worked — eventually. It's the "eventually" part that's key.

I can recall dozens of blown circuit breakers, exploding electrolytic capacitors, and shorted vacuum tubes. However, I also recall the satisfaction of seeing copper, carbon, and steel come to life.

With time and savings, I later could buy just about anything that I wanted — from commercial test gear to top-of-the-line ham radio equipment. It made for a great looking test bench and ham shack, but I lost out on the learning end of things. It didn't matter that I could read the schematic of the hermetically sealed phase locked loop synthesizer in my communications transceiver — I could never really know it. I could replace it if defective, but not really fix it the way I could one of my old creations.

From a practical perspective, having a nice portable o-scope with high bandwidth and Flash memory storage makes debugging a pleasure. Then, there's the safety issue — none of my creations were UL listed or approved.

So, there's nothing wrong with new gear that's compact, safe, and easy to use. It's just that — from an experimenter's perspective — shiny commercial equipment can become a black box. I make a habit of disassembling everything I buy; in part to understand what's in the black box, but it's still an imperfect exercise.

If your goal is to maximize the learning experience — whether for yourself or someone you hope to pass on your knowledge of electronics to — then I'd consider the old school "junk box" approach to learning. Fill your box with parts from tear-downs of whatever you can get your hands on. It's amazing what you can harvest from an old PC, for example. Even a discarded compact florescent bulb can yield a half dozen reusable components.

I'm fortunate to live a few miles from MIT, where there's a regular flea market of used test gear and lab equipment that's sold by the pound. Find out where your local ham or flea market is held and drop by at least once a year. Even if you don't use parts harvested from the gear to build your own, the exercise of a tear-down is educational in itself.

You can't wildly rip things apart, however. Take a methodical approach, trace the connections to see what components are associated with each other and — if you can — create a schematic diagram of the circuit in the device.

Lately, I've been partial to vacuum tube projects. With a few tubes and high voltage power supplies on hand, it doesn't take much effort to build oscillators, tuners, sound effects devices, and so on. So, go ahead. Give the "old school" junk box method of setting up your workbench and your communications, robotics, or other projects a try. Your projects may not look as attractive as the commercial systems, but you'll really understand the inner workings of what you build.

You'll then be well on your way to being a real experimenter. **NV**

BY JEFF ECKERT

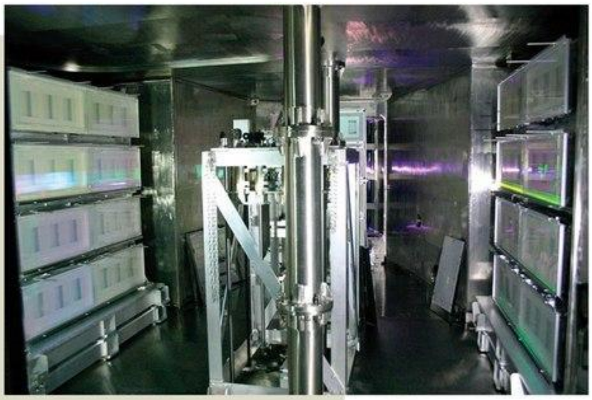
ADVANCED TECHNOLOGY

World's Most Powerful Laser (Sort Of)

The Institute of Laser Engineering (ILE) at Osaka University (**www.osaka-u.ac.jp/en**) has announced the successful firing of an enhanced version of its Laser for Fast Ignition Experiment (LFEX), delivering two petawatts (i.e., 2,000 trillion watts) of power. Using this laser, it is possible to generate a range of high energy quantum beams, including electrons, ions, gamma rays, neutrons, and positrons.

This is said to be a big step forward for "creating such new fundamental technologies as cancer therapy for medical applications and non-destructive inspection of bridges and buildings; to contribute to our ... longevity, safety, and security; and for the realization of fast ignition as an energy resource." The fast ignition scheme proposed by ILE is also expected to produce fusion energy from a relatively small laser system, which could produce a breakthrough in laser fusion.

The researchers proudly noted that two petawatts is equivalent to 1,000 times the total power consumed by the entire world. This is an impressive number, but one shouldn't forget that wattage without reference to time doesn't tell us much about the total energy usage, and this zap lasted only one trillionth of a second. The total energy needed for the beam is only enough to power a microwave oven for a couple seconds. Still, you probably don't want to look directly into the beam. ▲



■ The ILE pulse compressor delivers 2,000 trillion watts.

Looney Tonnes

When 3D printers began to surface, we noted here that users had only begun to imagine the plethora of possible applications. Proving the point, it was recently reported that NASA's Solar System Exploration Research Virtual Institute (**sservi.nasa.gov**) is kicking around the idea of building a complete lunar base that way. Apparently, some scientists were thinking about solving the problem of lugging the construction materials 238,000 miles, and it occurred to them that printing it using local materials (lunar dust) might be a better approach.

The European Space Agency (ESA) liked the idea and is collaborating with architects to see how and if it can be done. Reportedly, they are now thinking in terms of a weight-bearing "catenary" dome design that employs a cellular structured wall to shield against micrometeoroids and radiation. It also incorporates a pressurized inflatable shelter for astronauts and a hollow closed-cell structure to add strength and weight.

According to ESA, "a mobile printing array of nozzles on a 6 m frame sprays a binding solution onto a sand-like building material. First, the simulated lunar material is mixed with magnesium oxide to turn it into 'paper' to print with. Then, for the structural 'ink,' a binding salt is applied to convert the material to a stone-like solid."

The agency has already demonstrated the ability to print stone-like building blocks that weigh 1.5 tonnes (1.65 tons). Envisioned is a next-gen printer that can attain print speeds of 3.5 m per hour, so an entire building could be completed in about a week. What happens if it is successful on the lunar surface? Next stop, Mars. ▲

■ Multi-dome lunar base under construction using 3D printing.

■ Completed 3D printed lunar base.

ADVANCED TECHNOLOGY Continued

Clock to Measure Force of Gravity

Okay, the accuracy provided by your $620,000 Bulgari Magsonic Sonnerie Tourbillon watch is pretty good. However, the king of precision chronometry is still the National Institute of Standards and Technology (NIST, **www.nist.gov**) and their JILA (Joint Institute for Laboratory Astrophysics) facility's strontium lattice atomic clock that now achieves precision and stability levels ensuring that it will neither gain nor lose one second in 15 billion years (approx. the age of the universe). This is accomplished via two specialized thermometers that are inserted into a vacuum chamber filled with ultra-cold strontium atoms confined by lasers. (Strontium "ticks" at 430 trillion per second.)

While this won't be of much interest to most, it's worth noting that it has the potential to affect things like advanced communication and positioning systems, and it could form the basis of an altimeter that is so sensitive that it can measure tiny changes in the passage of time and the force of gravity at slightly different heights.

■ Vacuum chamber containing two thermometers that "take the temperature" of strontium atoms. Credit: Marti/JILA.

(Recall Mr. Einstein's theories.)

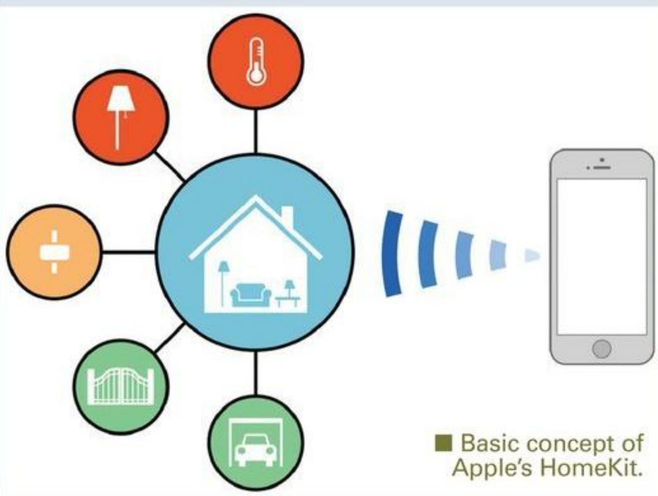
At its present level of precision, the clock could be used to sense a gravitational shift between distances of as little as 1 cm. ▲

COMPUTERS and NETWORKING

Are You Ready for Home Automation?

As recently as August 2014, the IT research firm Gartner, Inc. (**www.gartner.com**), listed the Internet of Things at the top of its "Hype Cycle for Emerging Technologies" list, speculating that the technology will take 10 years or more to reach a "plateau of productivity." Gartner may have to rethink that, as more and more companies are pumping out home automation products, including Chamberlain (garage door openers), Honeywell (thermostats), Incipio (AC power outlets), Schlage (door locks), Withings (security camera/air quality monitors), Lutron (lights), and many others. By the time you read this, Apple's iOS 9 should be out of beta test and ready for download, which is significant because it will include upgraded support for the company's HomeKit home automation software.

Basically, HomeKit allows you to configure and name HomeKit-enabled devices throughout your house, and control them using an iPhone or iPad (and, eventually, an Apple Watch). Setting things up involves creating a unique name for every device and every function, but the process is less daunting than it might have been since you can use Siri to set up HomeKit — no need to laboriously type everything in. Furthermore, you can use a grouping feature to control multiple appliances with a single spoken command.



■ Basic concept of Apple's HomeKit.

It sounds fairly exciting, but at this point, many questions remain. For example, can you use devices that are designed to work with the competing Google Nest system? (Maybe, if you have a HomeKit bridge, and some products that will be compatible with both.) Can you control HomeKit devices from outside the home? (Yes, but because HomeKit communicates via low power Bluetooth, you may need to pick up an Apple TV to make it work.)

It's all still somewhat murky, but nevertheless, it looks like home automation isn't going to wait 10 years. ▲

COMPUTERS and NETWORKING Continued

Mini Android PC, Mini Price

■ Jide's Remix Mini offers a 64-bit chipset and costs less than an electric toothbrush.

When you introduce a Kickstarter project with a $50,000 goal and end up raising well over $1 million, that's a pretty good sign. Well, that's what happened with the Remix Mini — said to be the world's first true Android PC.

The little computer built in China by Jide (**www.jide.com/en**) is based on the Android-based Remix OS, so you get to use the entire range of Android apps plus standard PC features like multiple windows, a taskbar, and the ability to hook up a mouse and keyboard. Sure, you're not likely to use it for high-level gaming or tracking satellites, but the Mini does include an Allwinner 64-bit, 1.2 GHz, quad-core Cortex-A53 CPU, and your choice of a device with 1 GB RAM and 8 GB of eMMC Flash storage (slower than SSD, but cheaper), or 2 GB and 16 GB. Not exactly a powerhouse, but the benefit is that it doesn't suck up much power, either.

A typical desktop will gulp down between 65W and 250W, but the Mini sips a mere 10W. The machine includes 802.11b/g/n Wi-Fi and Bluetooth 4.0 connectivity, and a 4K HDMI 1.4 port, Ethernet port, a headphone jack, and two USB 2.0 ports. Best of all, they are slated to sell for $40 and $50 after manufacturing ramps up, so your Christmas shopping can be pretty simple this year. ▲

CIRCUITS and DEVICES

Sensor to Avoid Crowds

■ The Density sensor and API help to avoid crowds and smooth out foot traffic.

Let's say you're one of those folks who doesn't like crowds and hate standing in line. As a result, you often decide to stay home to avoid the frustration of waiting for a table at your favorite restaurant, fighting the hoards for sale items at Macy's, or bumping up against a herd of sweaty people at the gym. If so, you'll be happy to hear about a new device from recent startup, Density (**www.density.io**).

The Density IR sensor is a basic people counter that — installed at the entrance to a public place — keeps track of how many people have entered and how many have exited, thus allowing it to provide both real time and historical data to help you decide when and if to drop in. Gathered information is collated by the Density Application Programming Interface and transmitted to a web application for use by a custom app.

For example, a company team is installing them in UC Berkeley gyms and other workspaces, and a Sacramento-based outfit called Requested (**requestedapp.com**) uses the system to generate restaurant discounts during off-peak periods. It doesn't appear that a significant number of locations have Density installed at present, but who knows? It could catch on — especially given that hardware and installation are free. There is a monthly fee of $25, however, to use Density, but if it generates even a few extra customers, it seems well worth it. ▲

CIRCUITS and DEVICES Continued

E-Bike for Hard-Core Trailblazers

For those who are loathe to generate internal combustion noise and corrupt the atmosphere with petroleum-based exhaust but nevertheless appreciate the feeling of blasting through the woods with wind in their hair and tree branches in their faces, Grace Bikes (**www.grace-bikes.com**) has a solution.

The latest Grace MX II trail bike now sports 27.5 inch wheels for improved traction and better road holding during acceleration and cornering, in addition to its precipitation-hardened aluminum alloy frame, Rock Shox Sektor Gold RL fork, SRAM X9 nine-speed shifting system, and Bosch Gen2 Performance Line motor. The bike weighs in at a fairly hefty 22 kg (48.5 lb), but that includes the 11.6 Ah/400 Wh battery pack. Range is specified at up to 185 km (115 mi) on a charge, but "is measured under optimal conditions and therefore may vary greatly in practice."

To put it another way, only an idiot would stray too far from the road. Looks like a lot of fun, but it will set you back close to $4,000, which is $600 more than a 144 cc Kawasaki KLX off-road motorcycle. Hmmm ... ▲

■ The Grace MX II electric trail bike: quiet and emission-free.

INDUSTRY and the PROFESSION

Top 10 World Changers

According to technology market analyst, IHS Technology (**technology.ihs.com**), the three technologies that are most likely to change the world over the next five years are: (1) the Internet of Things (surprise, surprise); (2) cloud computing/big data; and (3) 3D printing. To arrive at that conclusion, IHS gathered experts from the technology supply chain from electronic components to finished products, and looked at markets ranging from consumer, media, and telecom, including industrial, medical, and power sectors. Experts were asked to nominate their top 10 most impactful technologies over the period.

Finishing up the top 10 are artificial intelligence, biometrics, flexible displays, sensors, advanced user interfaces, graphene, and energy storage/advanced battery technology. **NV**

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

• Phone Service Blues • CPLD vs. FPGA • Sump Pump Alarm

Plain Old Telephone Service Blues

Q Many people are going to DSL or are getting phone service from their cable, and those new systems don't accept pulse dialing. A friend whose home is stocked with antiques — including the phone — complains that he is stuck with the AT&T POTS line. Our state has given permission to pull the plug on POTS in a few more years. I'm wondering how difficult it would be to program one of those computers on a chip to count the pulses from the dial mechanism (as I recall, the last/end pulse of the group is longer) and generate the corresponding touch tone (or control a tone chip).

There may be space to hide a circuit in those old bulky phones but probably not a universal board, and it would have to be powered from the phone line DC. **Plan B** — the easier solution — would be to exploit the unused second pair of wires of the phone system to send the dialer pulses separately to a central interface, since my friend doesn't have a second line but does have a second phone. The one old phone I found has a three-wire cord with two wires paralleled in the phone, so a common wire would be necessary to keep the old cord.

Dennis L. Green
via e-mail

A For our "younger" readers, POTS stands for Plain Old Telephone Service (a.k.a., rotary dial, which was introduced in 1876 by Alexander Graham Bell) to distinguish it from updated telephone systems such as touch-tone dial (started being used over wide areas in the 1980s), fiber optic based systems, ISDN, VoIP, cellular wireless systems, etc. The Digital Subscriber Line (DSL) to me is the telephone company's gateway to the Internet, but can be used for economical worldwide telephone service via Skype, Vonage, Magic Jack, etc.

The POTS used 48 VDC modulated with the voice/audio signal over a frequency range of 300 to 3,400 Hz for communicating analog signals and a 120 VAC ringing signal. DSL uses the same lines as the POTS with modulated channels, with 4,315.5 Hz bandwidths between 10 kHz and 100 kHz to send digitally encoded signals.

■ FIGURE 1.

■ FIGURE 2.

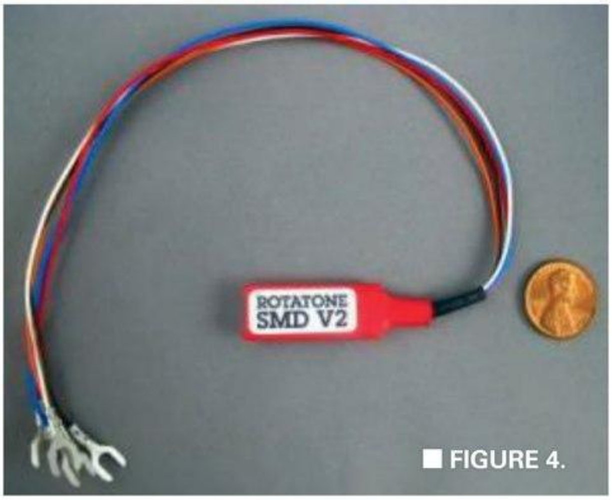
The POTS depended on the rotary dial shown in **Figure 1** (literally a round dial with 10 finger holes on the front of the phone) which opened and closed a switch a number of times based on the number selected to dial out (as kids we found early on that by clicking the handset switch we could dial equally effectively). I cannot remember when I have seen a rotary phone in service (I may have one somewhere in my junk collection).

The real forte of the rotary phone was reliability because except for a strong lightning strike taking out the hybrid transformer (isolated the residential side from the phone company side), there wasn't much to break (as much as some of us tried). The transformer could be replaced in five minutes by a competent tech in a day

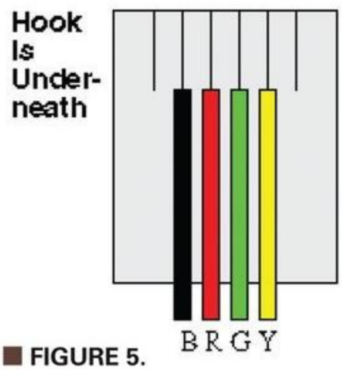
DTMF keypad frequencies ■ **FIGURE 3.**

	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Post comments on this article at
www.nutsvolts.com/index.php?/magazine/article/december2015_QA.



■ FIGURE 4.



■ FIGURE 5.

when we didn't throw away anything that could be repaired. Under POTS, the phone company owned the phone hardware and internal phone lines — not the customer.

The touch tone or Dual Tone Multiple Frequency (DTMF) is shown in **Figure 2** with the tone matrix scheme shown in **Figure 3**. The touch tones were a breath of fresh air to the mundane world of the old black rotaries, and even though the beige phone shown was soon replaced with various colors, it was a lot more aesthetic that the old POTS models. Early touch tone phones had a switch to allow pulse or tone dialing because in the '80s both systems were in service. I don't remember anyone mourning the end of the pulse dial because DTMF allowed the user to do other functions such as "Press 1 for Customer Service."

I found the Rotatone SMD Version 2 shown in **Figure 4** available from Old Phone Works (see Q&A SIDELINES) which converts the rotary dial signal pulses to DTMF. For the cost of parts (about the same as the Rotatone SMD V2) and the hassle of developing a one-off design, a proven device is the way to go. In my case, I even considered wireless phones for my home because one wire to the base unit would operate all of the phones I will ever need (we have seven), and would allow me to place the cordless handsets anywhere in the house without the hassle of rewiring each time (sorry Alf, my days of crawling under houses are over).

As far as residential telephone wiring, take a look at **Figure 5** to see the four-wire standard used in many areas with the modular jacks, and plugs RJ 11 for two-wire systems and RJ 14 for four-wire systems. Our area only uses the two-wire system, so I could use the black-yellow pair for a second phone line or in case the red-green pair fail (not super successful since what took out the R-G probably got the B-Y too).

A side note is that I have found phone lines in our area indicate reversed wiring on the phone line testers. I would really avoid an external device like the plague, and a central unit could be a real nightmare. The Rotatone device appears to do what you need and should be able to fit inside the old POTS phones which were manufactured before the days of "engineering Olympics" of trying to stuff more and more into smaller and smaller spaces.

I hope this information helps your friend maintain his nostalgic phone within the modern phone network.

CPLD vs. FPGA

Q I noticed in the July *NV* magazine the subject on FPGAs written on by Ryan Clarke. I have been using a CPLD in different applications, but have never used a FPGA. Can someone explain the differences between an FPGA and CPLD?

Also, there is a wide range of parts with different speeds and operating voltages. Is there a table that compares the different parts? This subject could make an interesting article.

John Roselle
via e-mail

Q&A *SIDELINES*

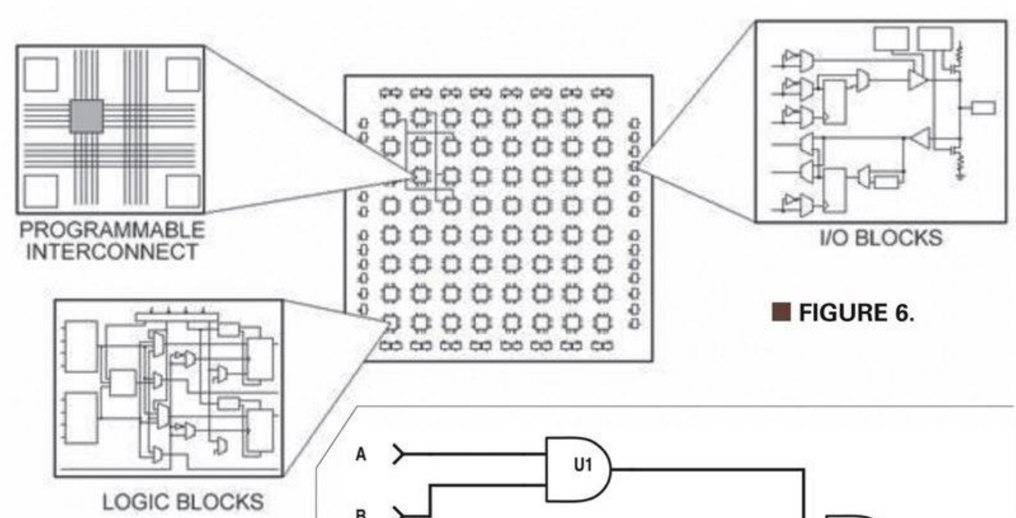
Plain Old Telephone Service Blues

Old Phone Works
www.oldphoneworks.com/rotatone-pulse-to-tone-converter.html

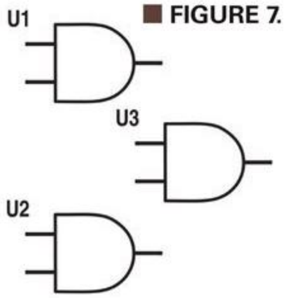
Breaker Tripped/Sump Pump Alarm
Amazon.com

Power-out Power Failure Alarm and Safety Light LED
www.amazon.com/gp/product/B00IA9BF70/ref=pd_lpo_sbs_dp_ss_1?pf_rd_p=1944687602&pf_rd_s=lpo-top-stripe-1&pf_rd_t=201&pf_rd_i=B0016P7ERK&pf_rd_m =ATVPDKIKX0DER&pf_rd_r=0B4E7Z6F49J0RT9MK3MY

Failed Circuit Alarm
www.amazon.com/CSH-Incorporated-FCA2-05-Failed-Circuit/dp/B0016P7ERK

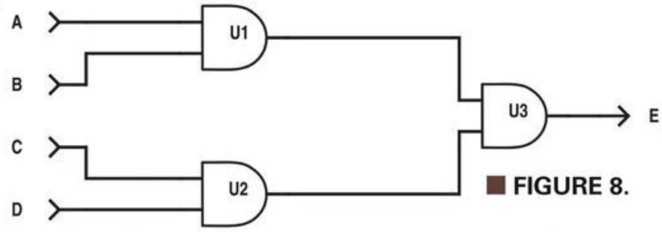


■ FIGURE 6.



■ FIGURE 7.

Simple Example of FPGA Logical Block



■ FIGURE 8.

Example of FPGA Logical Block after Programming Interconnects

A Good question, John! I cannot write a complete article in this column due to space constraints (maybe Ryan Clarke can write a follow-up article on CPLDs and PALs), but I will try to explain the differences between FPGAs and CPLDs.

CPLD stands for Complex Programmable Logic Device and FPGA stands for Field Programmable Gate Array.

Figure 6 shows the architecture of an FPGA which consists of programmable interconnects, logical blocks (consists of look-up tables, flip-flops, and a routing matrix), and Input/ Output (I/O) blocks. The FPGA user programs the interconnects to "wire" the logical blocks using the look-up tables and routing matrix in a fashion which produces a specific output for a given input (e.g., an input of 0001000 may produce an output of 10010001).

Look at **Figure 7** for a simplified FPGA logical block based on AND gates. **Figure 8** shows the FPGA logical block after the user has programmed the interconnects with the FPGA truth table in **Figure 9**.

A real FPGA project is much more complex than my example, but you should get the idea. The FPGA manufacturers provide Integrated Synthesis Environment (ISE) software which allows you to program your FPGA logical blocks via a PC using keyboard symbols. Then, the program can be coiled on the PC and downloaded to an FPGA programmer.

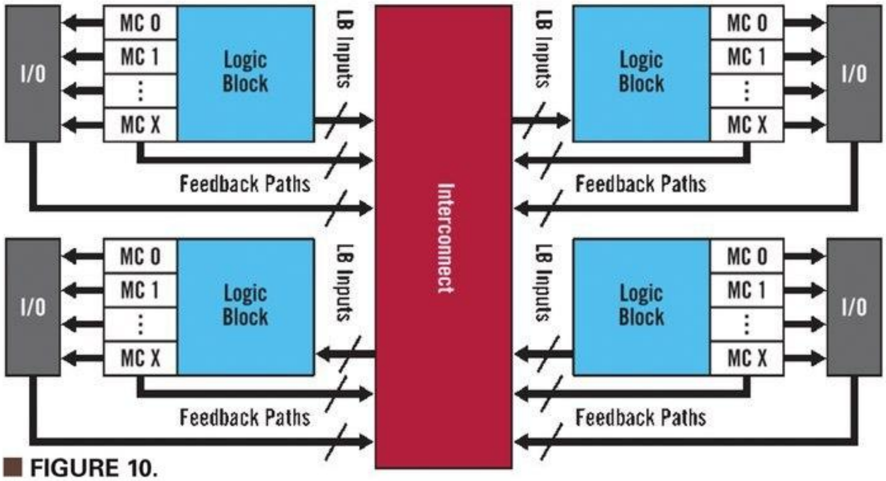
I remember in a training seminar having to create a logic circuit using the basic gates (AND, OR, NAND, NOR, and flip-flops) that would read out the numbers of my birthday when I pushed a switch. This took a couple of 12 hour days and the board looked like a rat's nest with so many wires. Then, we used FPGAs to produce the same result. I already had the truth table from the previous project which took only 30 minutes to derive, so the FPGA programming, compilation, and download took less than 15 minutes. WOW!!!

In **Figure 10**, I have a CPLD block diagram. The MC is a macrocell which performs combinational and sequential logic, complementing, and feedback functions. In essence, the CPLD is an FPGA without the look-up tables and complex functionality. CPLD manufacturers provide software programming aids like the FPGA ISE. CPLDs and FPGAs with JTAG (Joint Test Action Group) ports can be programmed using the same JTAG compatible programmer.

Comparing FPGAs and CPLDs: FPGAs do not store their configurations which must be reloaded every time the

A	B	C	D	E
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Example FPGA Truth Table ■ FIGURE 9.



■ **FIGURE 10.**

device starts up, whereas CPLDs store their configurations and are thus ready for action upon start-up. FPGAs contain complex functions such as adders, multipliers, memory, and SerDes (Serializer/Deserializer), whereas CPLDs do not. FPGAs are more capable than CPLDs, but are slower in their operation.

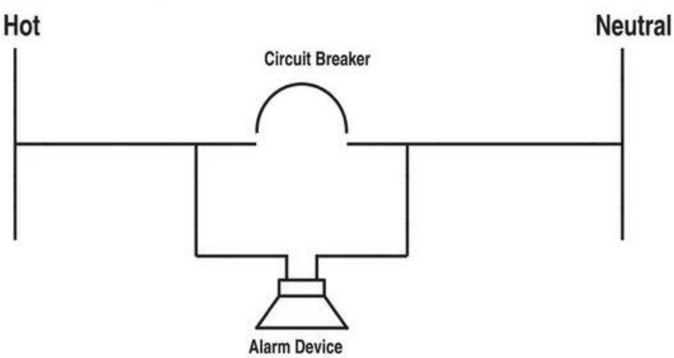
This is an extremely oversimplified discussion of CPLDs and FPGAs but I hope it will help you further understand these very useful devices.

Breaker Tripped/Sump Pump Alarm

Q Since my septic tank is higher than the drain line from my house, I have a holding tank with a pump to push the wastewater up to the septic tank. I found out this weekend that the pump, sump pump, and crawl space lights are all on the same breaker when the lights would not work. Fortunately, it was my sump pump that had caused the overload. I need something to alert me if this breaker is off due to the septic backup possibility. I guess an indicator light would be easiest. However, I was thinking I would like to have a small alarm that would sound when this breaker is tripped. Can you give me some suggestions? Love your column.

Les Manis
via e-mail

■ **FIGURE 11.**



Schematic of Simple Breaker Trip Alarm

A In **Figure 11**, I have shown a simple way to alarm when a breaker is tripped. Be sure to have an alarm device that is compatible with the voltage between the hot and neutral lines; be sure the alarm device has the proper circuit interrupter (fuse, etc.) to avoid fires or damage in the event of a failure; and be sure that there is no switch between the alarm device and the neutral line.

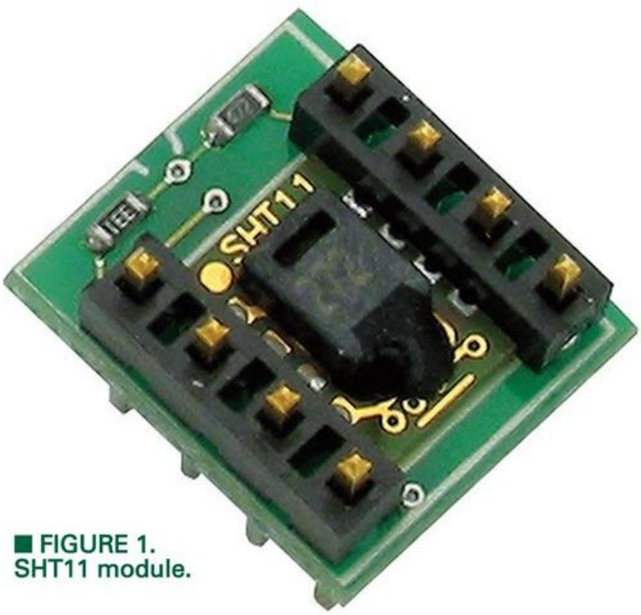
The alarm device could be a lamp or buzzer that is compatible with the line voltage (I assume 120 VAC for a sump pump circuit). If you want to use an electronic (i.e., low voltage) device, a relay could be hooked up as my alarm device and its contacts used to trigger the low voltage device. REMEMBER: Be careful when working around line voltages to avoid electrocution and/or DEATH. If in doubt, call a licensed electrician.

If you want a simple plug-in alarm solution, I found a couple of devices on **Amazon.com** for reasonable prices (see **Q&A SIDELINES**). **NV**

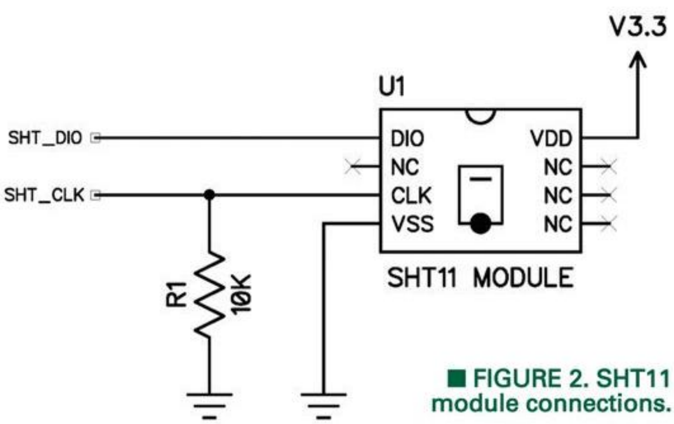
BY JON MCPHALEN

Some Like It Hot

Over the summer, I was able to escape the heat and traffic of Los Angeles for 10 days of heaven in Wasilla, Alaska — a wonderful birthday gift from my sister, Reenie and her husband, Lew. It was as a vacation should be: Mostly relaxing, enjoying the tranquility of the forest that is their back yard, though we did mix in a fantastic day of fishing in Seward (I caught two salmon, two halibut, three rockfish, and a sabelfish!). The night we got home from fishing, we enjoyed an incredible feast: halibut and rockfish fresh from the ocean, with vegetables fresh from Lew's greenhouse. There were even wild raspberries picked from the perimeter of the yard. It was amazing!



■ FIGURE 1. SHT11 module.



■ FIGURE 2. SHT11 module connections.

As you might imagine, the growing season in Alaska isn't long — even with a greenhouse — so Lew works very hard tending to his crops to get the most out of them. Of course, there's the occasional moose that likes to pop in and steal a snack — something most gardeners don't have to worry about. Temperature control of his greenhouse is completely manual: opening and closing louvers and using fans as needed. I really like Lew, and while flying home I started noodling ideas for some kind of automation to help him maximize production from the greenhouse. I've got between now and June to come up with something neat.

To that end, I thought I'd start with temperature and humidity monitoring. I've had a Parallax SHT11 module sitting on my desk for ages, and many years ago I used it with a BASIC Stamp 2. Seems like now would be a good opportunity to write a Propeller object for it (yes, others exist, but I really like to write my own objects, and I encourage you to do the same when the opportunity presents itself).

The SHT11 module combines a surface-mount sensor, a pull-up (to Vdd) on the data pin, and an inline series resistor onto a breadboard-friendly DIP-8 module (**Figure 1**). The clock pin of the module is a driven input, but there have been reports of the module locking up if the connection between the host and the SHT11 is not pulled down. I figure a 10K resistor is cheap insurance from possible frustration. **Figure 2** shows the connection between the module and the Propeller.

Communications with the SHT11 are fairly straightforward; we just have to remember that we're using an open-drain connection on the data (DIO) line. Just for clarification, this means the pull-up on the module will create the "1" state on the DIO line when we float that pin. To write a "0," we will make the control pin an output and low. All this is accomplished by write a zero to the `outa[dio] bit`, then manipulating the `dira[dio]` to cause a 1 or 0 state on the DIO line.

After power-up and assigning pins, we need to run a *connection reset* sequence to the SHT11. This is defined as nine or more clock pulses while the data line is high (left side of **Figure 3**). Again, floating the DIO pin will allow the pull-up on the module to take that line high:

```
pub connection_reset

  dira[dio] := 0

  repeat 9
    outa[sck] := 1
    outa[sck] := 0
```

We should only need to do this once, but the method

Post comments on this article and find any associated files and/or downloads at
www.nutsvolts.com/index.php?/magazine/article/december2015_SpinZone.

is public in case communications are lost with the device.

Generally speaking, communications with the SHT11 are similar to I²C in that we're using an open-drain connection on the data line, we transmit eight bits at a time, and the receiving end will acknowledge (pull the data line low) on the ninth clock pulse. All communications begin with a *transmission start* sequence (see right side of **Figure 3**), followed by write and read transactions. As we need it to initiate all transactions, the *transmission start* sequence is a stand-alone method:

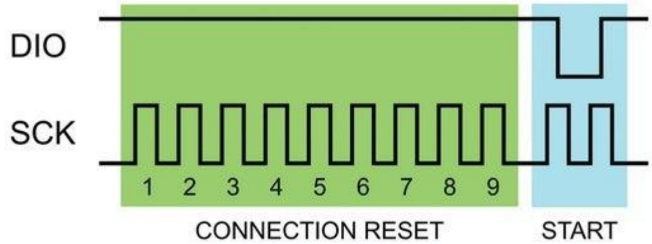
```
pub trans_start

  dira[dio] := 0

  outa[sck] := 1
  dira[dio] := 1
  outa[sck] := 0
  outa[sck] := 1
  dira[dio] := 0
  outa[sck] := 0
```

The **trans_start()** method begins by releasing the DIO pin to the module's pull-up, taking the line high as required. The clock line goes high, followed by writing 1 to the `dira[dio]`. As the **outa** bit for the DIO pin was initialized to 0, this action will cause the DIO pin to become an output and low, pulling the DIO line low as required by the sequence. The first clock is finished, the second started, then the DIO line is released to the pull-up again by writing 0 to `dira[dio]`. This sequence illustrates using open-drain output on the DIO pin while the SCK (clock) pin is driven.

Communications with the module are accomplished with the **wr_byte()** and **rd_byte()** methods. Let's start with **wr_byte()** as it can be tricky to understand at first:

```
pub wr_byte(b) | ackbit

  b := (b ^ $FF) << 24

  repeat 8
    dira[dio] := (b <-= 1)
    outa[sck] := 1
    outa[sck] := 0

  dira[dio] := 0

  outa[sck] := 1
  ackbit := ina[dio]
  outa[sck] := 0

  return ackbit
```

The purpose of this method is to write byte *b* to the SHT11 and return the state of the acknowledge bit. Again, we do not want to drive the DIO pin high, so we have to manipulate *b* for control of the `dira[dio]` bit. We start by inverting *b* by using XOR with $FF. Remember, the `outa[dio]` bit is already 0; what we want to do is pull the DIO line low when a bit in the original byte *b* is 0. This



■ **FIGURE 3. SHT11 Connection Reset and Transmission Start.**

is accomplished by flipping the bits in *b* and then writing them to `dira[dio]`. Data to and from the module is MSBFIRST, so we shift the inverted bits of *b* left by 24 to move them to the upper byte (byte[3]) of *b* — remember, all parameter variables in Spin are longs (four bytes).

Now, we can drop into the actual transmission loop that will run eight times; once per bit. The first line rotates *b* by one bit; what this does is shift all bits left by one, then write what was in bit31 to bit0 (that last part is the difference between a shift and a rotate). By rotating left, we are accessing the bits from the MSB side of the value. Now, we can write that to the `dira[dio]` bit to control the DIO line. A 1 will cause the line to be pulled low; a 0 will allow the line to be released to the pull-up. When DIO is set up, the bit is clocked out by taking the SCK pin high, then back low.

After the byte is sent, we must release the DIO pin so that the SHT11 can create the ACK/NAK bit. With the DIO pin set to input mode, we take the clock pin back high and then read the state of the DIO pin. A good transmission will result in the DIO pin being pulled low by the SHT11 during this ninth clock pulse. The state of the ACK/NAK bit is returned to the caller. As you'd expect, the **rd_byte()** method reverses the process:

```
pub rd_byte(ackbit) | b

  dira[dio] := 0

  b := 0
  repeat 8
    outa[sck] := 1
    b := (b << 1) | ina[dio]
    outa[sck] := 0

  dira[dio] := !ackbit
  outa[sck] := 1
  outa[sck] := 0

  dira[dio] := 0

  return b
```

We start by setting the DIO pin to input mode and clearing the result variable *b*. Inside the loop (which runs eight times), we take the clock high, sample the DIO pin into the LSB side of the result, then take the clock back low. Note that the first part of the sample line shifts the

Jon "JonnyMac" McPhalen
jon@jonmcphalen.com

Parallax, Inc.
Propeller boards, chips, and programming tools
www.parallax.com

result variable *b* left by one bit; this makes space in the bit0 for the new sample.

Next is to send the ACK/NAK bit to the device. Again, we're using open-drain output so we write the inverse of the ACK/NAK value to the `dira[dio]` bit. The clock is taken high, then back low to send the ACK/NAK bit. We do a bit of clean-up by releasing the DIO pin (in case it had been pulled low) before we return the result.

Getting to the nitty gritty, a typical transaction between the host and the SHT11 works as diagrammed in **Figure 4**. At the beginning, there is the transaction start sequence, then a command, then a write or read depending on the command. In this case, I've diagrammed a temperature read since that's what we'll use most with the SHT11 module.

Note the section marked "Measure." After sending the temperature read command, the module will take some time to provide a result. When the result is ready, the SHT11 will pull the DIO pin low. If this doesn't happen in a reasonable amount of time (we give it 320 ms), there was a problem and the read is thrown away:

```
pub wait | t

  dira[dio] := 0

  t := cnt
  repeat 320
    waitcnt(t += clkfreq/1000)
    ifnot (ina[dio])
      return ACK

  return ina[dio]
```

This is easy. The DIO pin is set to input mode and we initialize a variable for a synchronized loop that is set to run every millisecond. Inside the loop, we check to see if the DIO pin has been pulled low. When that happens, we return ACK. If the loop runs out, the state of the DIO pin is returned; this will probably be 1 (NAK), but we give the device one last chance by reporting the current state of the DIO pin.

Okay, then. Let's read the temperature. My object assumes we're running in the default mode of reading a 14-bit temperature value:

```
pub temperature

  ifnot (inuse)
    return negx

  trans_start

  if (wr_byte(MEASURE_TEMP) == NAK)
    return negx

  if (wait == NAK)
    return negx

  sot := 0
  sot.byte[1] := rd_byte(ACK)
  sot.byte[0] := rd_byte(NAK)

  tc := sot - 39_70

  return tc
```

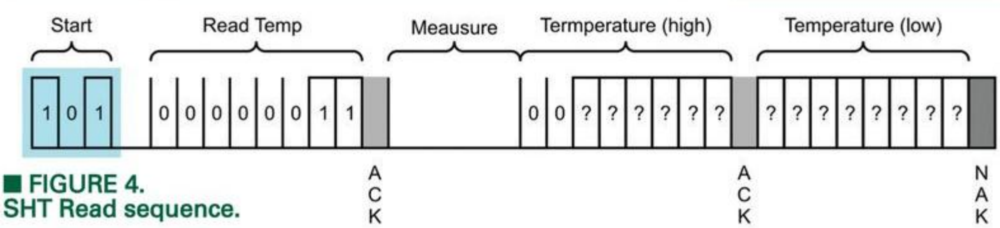


FIGURE 4.
SHT Read sequence.

The key to reading the temperature is doing the transmission start sequence and then reading back a two-byte (14-bit) value. There are checks to ensure we've defined the connection pins, that the device writes back an ACK when the command is sent, and doesn't time-out while measuring temperature. If any of these areas fail, the routine returns **negx** as a "bad reading" flag.

The value returned by the SHT11 requires some processing before returning to the caller. The formula for converting counts (*sot*) to degrees Celsius is:

tc = sot * 0.01 – 39.7

Floating point numbers are not native to the Propeller, but with plenty of space in a long we can simply multiply the conversion terms by 100. The result returned is temperature in 0.01 degrees Celsius. The variable *tc* is global to the object, so it can be called later and used for converting to Fahrenheit:

```
pub temp_c

  return tc


pub temp_f

  return tc * 9 / 5 + 32_00
```

Reading humidity is the same, but the conversion is a bit more involved:

rh% = -4 + (sorh * 0.0405) - ($sorh^2$ * 0.0000028)

If we multiply the conversion terms by 100 again, we get this:

```
  rhl := -400                        {
  } + (sorh * 405 / 100)             {
                             } - (sorh * sorh **
                             $12599E)
```

Note that the Propeller (like the BASIC Stamp 2) has a ** operator which allows us to multiply by fractional numbers. To calculate the term used by

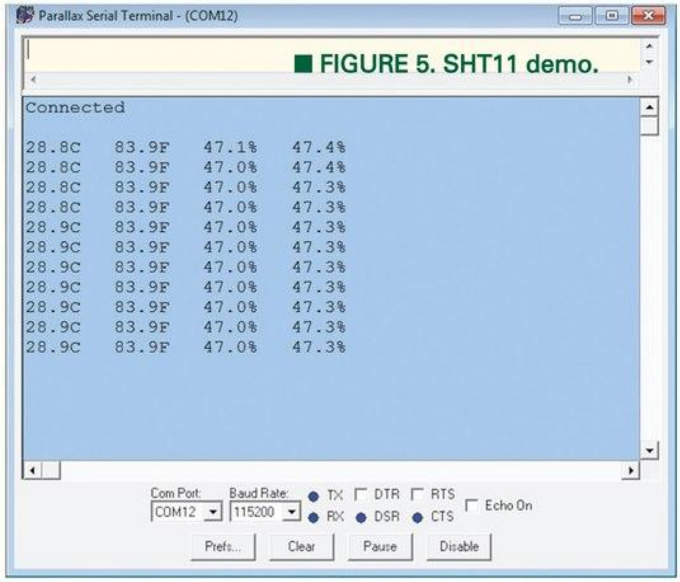
, we multiply the fractional value (0.00028) by 2^32; this is where we get $12599E. It's always a good idea to whip up a simple demo to test a new object. My demo verifies the SHT11 connection, then displays temperature and humidity every second (see **Figure 5):

```
pub show_temperature | tmp

  sht.temperature

  tmp := (sht.temp_c + 5) / 10
  term.dec(tmp/10)
  term.tx(".")
  term.dec(tmp//10)
  term.tx("C")
  term.tx(term#TAB)

  tmp := (sht.temp_f + 5) / 10
  term.dec(tmp/10)
  term.tx(".")
  term.dec(tmp//10)
  term.tx("F")
  term.tx(term#TAB)


pub show_humidity | rh

  sht.humidity

  rh := (sht.rh_linear + 5) / 10
  term.dec(rh/10)
  term.tx(".")
  term.dec(rh//10)
  term.tx("%")
  term.tx(term#TAB)

  rh := (sht.rh_true + 5) / 10
  term.dec(rh/10)
  term.tx(".")
  term.dec(rh//10)
  term.tx("%")
  term.tx(term#TAB)
```

As I'm writing this, it's early fall in Los Angeles — that time of year where the days are warm and dry and the mornings can be cool and damp, even in the valley where I live. I thought it might be fun to create a little data logger for the SHT11 with an Activity board which comes pre-wired with a microSD socket. The PAB (Parallax Activity Board) also has an XBee socket which I can populate with a Bluetooth radio module. By using my command parser, I can access the data logger from a terminal on my phone.

The project is a little bigger than I can fully explain in a magazine, and I did cover the command parser and most of the other objects used in previous columns. What I do want to share is simple file access — this seems to be a topic that stresses many newcomers in the Propeller forums.

For file access on the Propeller, I use the FSRW object; I believe it was the first file object created for the Propeller, and in my tests it's the fastest. FSRW might not have all the features of some newer objects, but that's secondary to me. In fact, FSRW is at the heart of my audio player code for the Propeller because I need the best speed. Some of the basic file access methods in FSRW are a little different, so I created shell access methods in my



■ FIGURE 5. SHT11 demo.

Propeller template to simplify things. First things first. We have to mount the SD card:

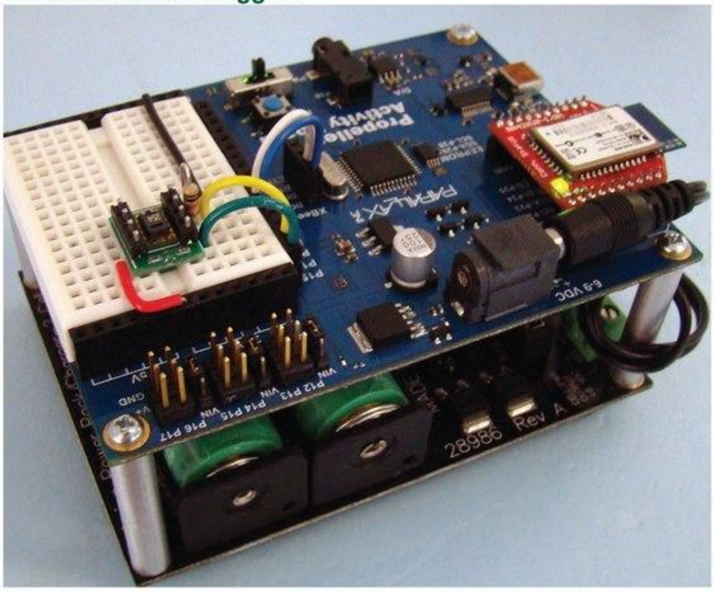
```
pub mount_sd | check

  check := \sd.mount_explicit(SD_DO, SD_CLK,
  SD_DI, SD_CS)

  mounted := (check == 0)

  return mounted
```

This points out an aspect of FSRW that can be a tad confusing at first: When something is okay, the FSRW routine returns 0, which the Propeller defines as **false**. The reason for this is that many of the methods in FSRW can return multiple ones of several error codes. In this case, 0 is good, so I assign the return value of **true** or **false** to *mounted*. If *mounted* is **false**, there's no SD card inserted, or the card is bad. Note, too, that we're calling the FSRW methods with an abort trap (\) — this allows the code to handle problems cleanly. If the SD card is mounted, we can open a file:

```
pub open_file(p_str, mode) | check

  if (mounted)
    check := \sd.popen(p_str, mode)
    if (check == 0)
      if (mode == "r")
        \sd.seek(0)
      return true
    else
      return false
  else
    return false
```

To open a file, we pass a pointer to a string that holds the file name, and a lowercase letter that defines the mode. If we want to open an existing file to read, we might do something like this:

■ FIGURE 6. Data logger.



```
if (open_file(string("TH151022.CSV"), "r")
  ' read the file
```

I found through experimenting that using the **seek()** method after opening a file seemed to improve initial read speed. If we want to create a new file or replace one that exists, the mode would be "w." We can add to an existing file with "a" (append), or delete a file with "d."

For the data logger application, I want to write values to a file in CSV format; this will allow me to open the file as a standard spreadsheet. To simplify writing a line of text to a file, I have this method:

```
pub wr_line(p_str, usecr,
uself)

  \sd.pputs(p_str)
  if (usecr)
    \sd.pputc(13)
  if (uself)
    \sd.pputc(10)
```

This takes a pointer to my text buffer and flags for appending CR (carriage return) and LF (line feed) characters to the string. As I'm a Windows user, I usually set both flags to true to match common Windows text files.

We can also read strings from a file with the Propeller. In fact, my data logger has a command called VIEW that will open and display the contents of the text file to the terminal. The FSRW object doesn't have a method for reading strings, so we're going to do this manually:

```
pub rd_line(p_str, n) | c, len

  len := 0

  repeat n
    c := \sd.pgetc
    if (c < 0)
      if (len == 0)
        return -1
      else
        quit

    if (c <> 10)
      byte[p_str++] := c
      len++
    else
      quit

  byte[p_str] := 0

  return len
```

This method requires a pointer to a string buffer and a maximum length. It will read the string from the SD card one character at a time until it encounters a line feed (10). When we're done, we return the length of the string just read in. If the result is -1, we have reached the end of the file. Here's an example of that method being used to read and display a text file from the SD card:

■ FIGURE 7. Bluetooth coms with logger.

```
pub show_file(p_file) | len

  ifnot (mounted)
    term.str(string("SD not
    mounted."))
    term.x(term#CR)
    return

  if (open_file(p_file,
  "r"))
    repeat
      bytefill(@linebuf,
      0, 81)
      len := rd_line(@line
      buf, 80)
      if (len > 0)
        term.str(@linebuf)
        time.pause(5)
      else
        close_file
        quit
```

As you can see, this method opens the file (defined by pointer *p_file*) for read mode, then iterates through the file reading and displaying one line at a time. There is a short delay after each line to prevent the terminal receive buffer from being overrun. When the resulting line length is zero or smaller, we can close the file and exit.

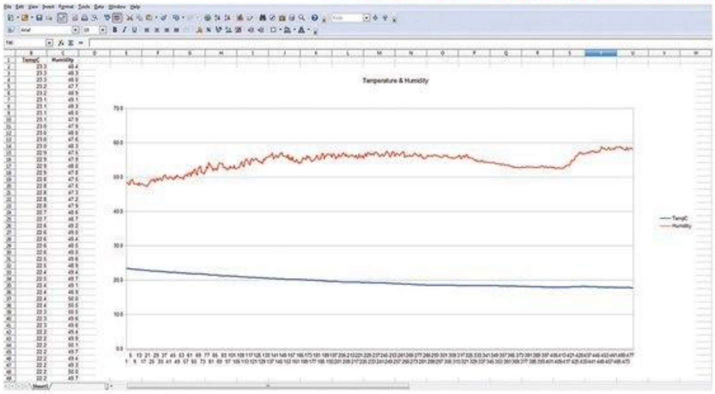
Figure 6 shows my little battery-powered data logger with the SHT11 sensor module. I attached the batteries so I could put it in my car overnight where I knew the temperature and humidity changes would be more dramatic than inside my home.

Figure 7 shows communications with the data logger via my Android phone. Sadly, iPhone users are not allowed access to serial Bluetooth modules, so this won't work for you (I'm exploring options for a future column). You'll notice the last command displayed, THNOW, shows the time, temperature, and humidity. When logging is activated, a string like this is written to the target file every minute (the logger lets you set the number of samples to read).

In **Figure 8**, you can see that I opened the file using OpenOffice Calc, and then created a graph of the temperature and humidity measured in my car while I was sleeping. What I find particularly interesting is that the humidity jumps near sunrise, even though the temperature is still falling slightly.

Okay, then. Whether you're building a greenhouse controller as I hope to do, or just want to know what's going on in your environment, the SHT11 module will let you do it. Have fun with this data logger. Since creating my parser, I have leaned toward interactive versus menu-based programs, and I'm having a great time. Give it a try. I think you will enjoy it too!

Until next time, keep spinning and winning with the Propeller! **NV**



■ **FIGURE 8. Graph logger values.**

BOM

Item	Source
Propeller Activity Board	Parallax #32910
SHT11 Sensor Module	Parallax #28018
10K Resistor	
Li-Ion Power Pack	Parallax #28989 (optional)
RN-42 Xbee Module	SparkFun #WRL-11601 (optional)

NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

ACTOBOTICS SERVO CONTROLLER

The Actobotics® Servo Controller is a compact two-channel servo controller that is extremely simple to use, while providing some very unique features. Connecting servos can be done in one of two ways: 1. NEARBY (via Direct Connect). The servos can simply plug into the row of pins on the top of the board just as they would on a receiver or regular servo controller. This option is commonly used when the servos are located near the controller and there's no need for lengthy wires. 2. LONG DISTANCE (via CAT6). A CAT6 cable can be plugged into the CAT6 port and run out to a CAT6 receiving board; the servos would then plug into the row pins located on the board. This setup is excellent for operating the servos from a long distance, and provides a clean and inexpensive way to extend the servo wires.

While the servo controller is ready to run right out of the box, the PWM (pulse width modulation) signal range can easily be changed in order to tailor the controller to a specific application. The stock PWM range is set to approximately 1,000-2,000 µsec (somewhat standard within the servo industry), which will cause most servos to rotate 90 degrees total (45 degrees each way from center). To change the range of the servo, the PWM range can be reprogrammed with a single button located on the circuit board. No computer is needed.

The servo controller is capable of handling 4.8V-7.4V nominal voltage which can be provided through the 2.5 x 5.5 mm power jack or into the row pins on the top of the board. The contoured aluminum box provides excellent protection for the board and fits well in the user's hands.

The tapped holes in the back of the box make mounting it to Actobotics channel (or any other component utilizing the standardized .770" hub pattern) simple. This controller comes fully assembled, ready to plug in and put to use. Price is $79.99.

4WD and 6WD MANTIS OFF-ROAD ROVERS

The Mantis™ 4WD and 6WD rovers also from ServoCity will take off-road robotics excursions to the next level. The bug-like chassis offers extreme A-arm style suspension and nearly 5" of independent wheel travel. With 313 RPM ball-bearing precision planetary gearmotors, there's plenty of torque to turn the aggressive 5.4" off-road tires.

The suspension incorporates 4.62" aluminum beams and 130 mm oil-filled aluminum bodied shocks. An 18" long boxed channel chassis provides a rigid backbone with endless mounting and customization options, making it easy to bolt on Actobotics components and electronics directly to the chassis of the rover.

Rubber grommets and end caps are included for protecting the motor wires as they get routed to the motor controller of choice.

The 2.2" diameter black revolver wheels are driven using 12 mm aluminum hex wheel adaptors to ensure a solid, no slip connection. The 4WD version is priced at $329.99; the 6WD version is $479.99.

For more information, contact:
ServoCity
www.servocity.com

DUAL MODE PROXIMITY AND TOGGLE SWITCH

The J2 LED Lighting, LLC AIR2 is a dual mode touchless switch for low voltage LED lighting control. The modes of operation are door proximity and hand wave toggle, both providing on/off control. The AIR2 is software driven providing automatic changes between mode functions. The body of the AIR2 is very small with a body length slightly under 1.80", a width under 0.70", and a thickness less than 0.30".

The operation of the AIR2 is such that in the door proximity mode when a surface (the door) is in near proximity (less than 100 mm/four inches), the switch is off. When the door surface moves out of range of the AIR2 sensor face, the switch turns on the controlled light. When no surface is in front of the AIR2 sensor face for more than 10 seconds, the software switches to the hand wave toggle mode. Moving a hand past the sensor face within the sense range toggles the light off and the switch is latched into this off position. The AIR2 will stay latched off until the hand wave in front of the AIR2 sensor face is repeated for on latching. The AIR2 will revert back to the door proximity mode when a surface is in front of the sensor face more than 10 seconds. The AIR2 may be operated as a door proximity switch or a hand wave toggle switch, depending on the needs of the application.

The AIR2 has a sleek profile with an extruded aluminum case with moisture resistant sealed front and back covers. The AIR2 operates from 7-24 volts DC. The maximum current switched at 7-12 volts is 2.5 amps, while the maximum current at 24 volts DC is 2.0 amps. The AIR2 can be operated from a nine volt battery for portable applications such as theatrical props or scenery.

The AIR2 is equipped with male and female DC barrel type connectors of the 5.5 mm x 2.1 mm size jack and plug. The cable length is a nominal 18" (460 mm) flexible UL2464 rated black PVC.

Securing the unit for general illumination applications in cabinet work for either door proximity mode or under a counter hand wave toggle mode, double-sided tape can be used on the unit's flat body shape. For applications in prop and scenery use, the unit can be temporarily secured with Gaffer's tape or hot glue. For general lighting applications, the AIR2 can be used with a 30 watt/12 volt DC constant voltage LED driver power supply.

The AIR2 is priced at $16.99 and is well suited for various LED lighting applications, for example:

- Theatrical prop and scenery LED light control
- DJ lights and audio rack lights
- Costume lights
- Architectural cabinet lighting
- Commercial and industrial applications

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

RF SIGNAL GENERATOR EXTENDS TEST PORTFOLIO

RIGOL Technologies, Inc., has expanded its portfolio of RF Test solutions with the introduction of the DSG800 Series RF Signal Generator. The DSG800 series covers

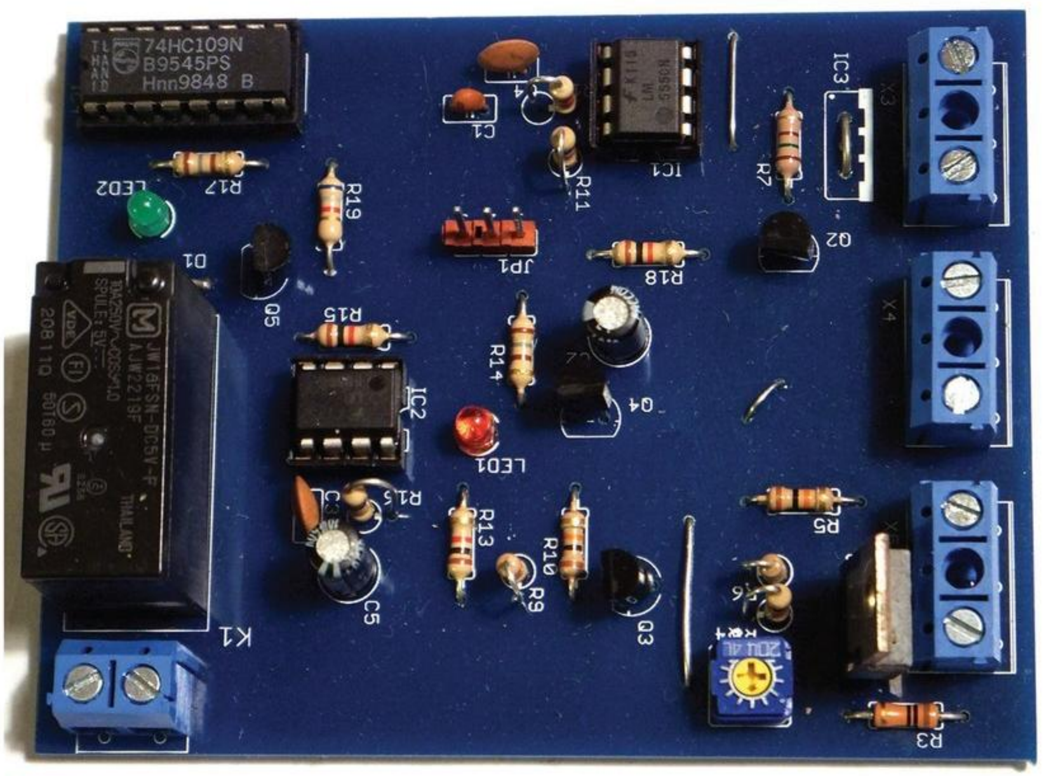
Continued on page 61

AUTOMATIC HALLWAY LIGHT

By Derek Tombrello

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/december2015_Tombrello.**

It was a dark and stormy night ... walking down a long dark hallway, when all of a sudden ... I tripped over something furry. It was Damien, my solid black cat, stretched out lazily, hidden in the dark, smack in the middle of the floor! Sure, turning on a light would have helped, but the only light switch had brilliantly been installed in the middle of the long hallway. Time for a creative solution.



I thought about just installing a passive infrared system — like everyone else does — but aside from my being a rebel who likes to go my own way, that has two distinct problems. First, being a narrow hallway, it's almost impossible to mount a single PIR system in such a way that both entrances are covered and would ensure prompt triggering. Secondly, the aforementioned feline would inevitably (in the course of his running laps around the house) trigger the light on and off all night long. Nope. I needed something else.

After a little thought, I came up with the idea of focusing an IR beam across each doorway about waist high. The cat could then run through the hall to his heart's content without triggering the light. For us, the light would come on regardless of the direction from which the hall was entered, and would remain on until we exited either end.

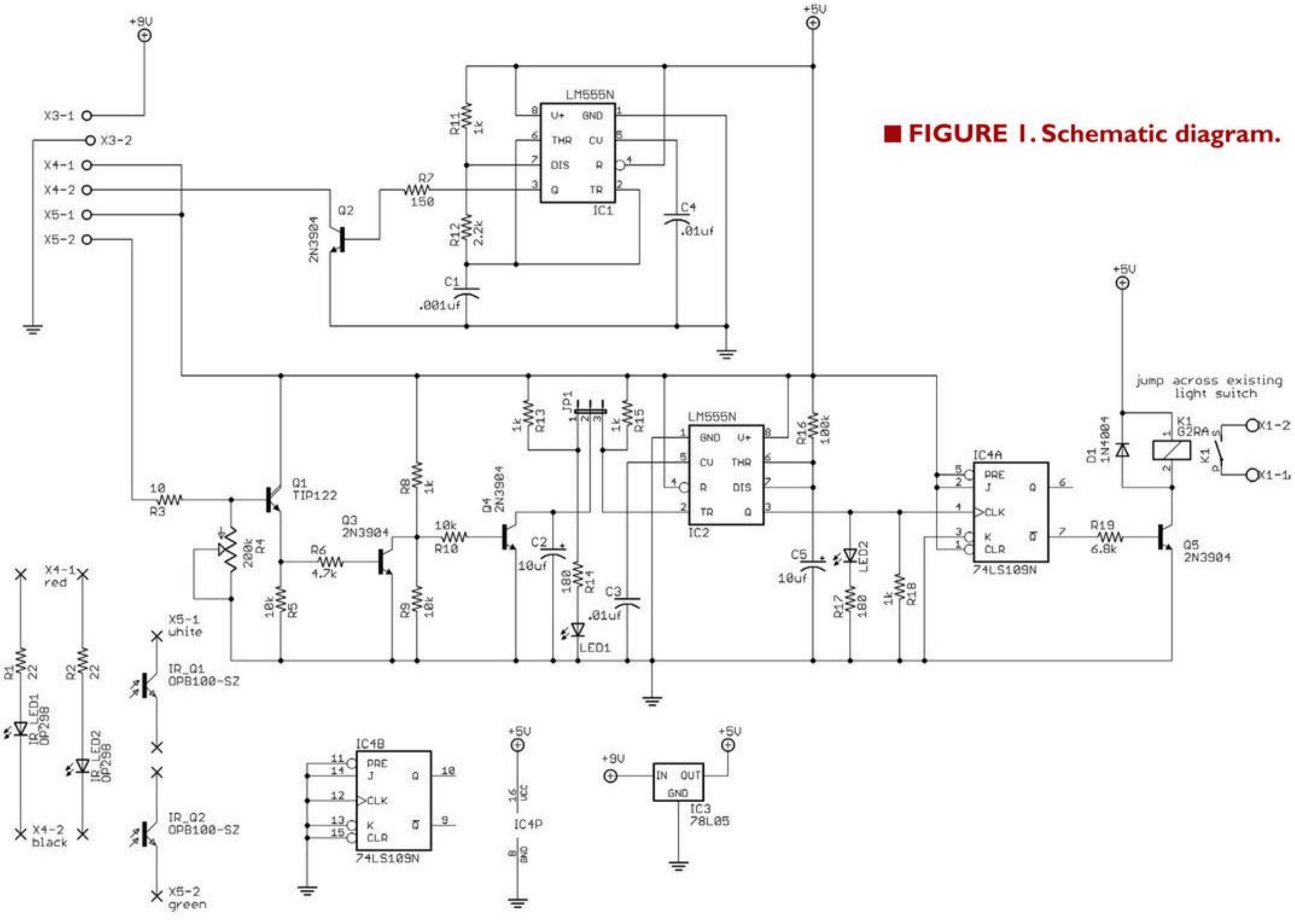
The Boring Stuff or How It Works

I began the design around a couple of photodiodes and Electronic Goldmine "high intensity" IR LEDs, but quickly found out that with this combination, I was not going to get a distance between the pair of any more than a few inches — not hardly large enough to squeeze through! A quick search through Digi-Key's database provided the answer: an IR receiver/transmitter pair made by Optek (part number OPB100Z). According to the datasheet, a range of up to 36" is possible — just far enough for a typical doorway. However, to achieve that range without the use of optics, the LEDs (OPB298) needed to be pulsed at a higher than typical continuous current, and the optotransistors (OPB598) needed to be amplified.

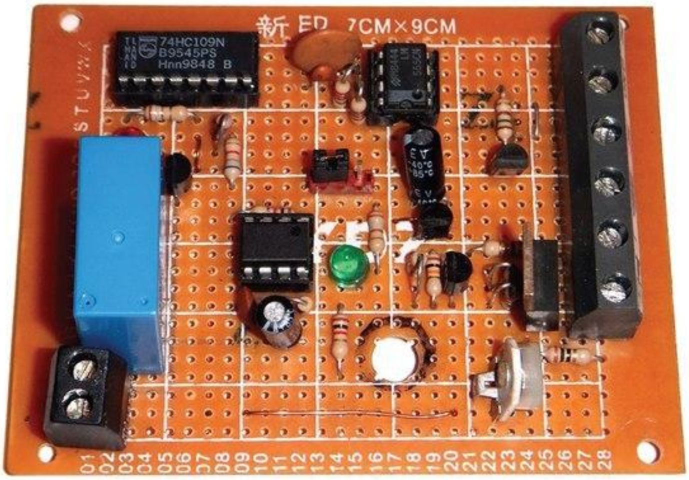
To pulse the LEDs (IR_LED1 and 2), I'm using IC1 (you can follow along on the schematic in **Figure 1**) which is a 555 IC timer configured as an astable multivibrator with a frequency of 267 kHz, a duty cycle of 59%, and an on-time of 2.218 µs as determined by R11, R12, and C1. I know that the datasheet for the 555 specifies that it is accurate from <1 MHz up to 100 kHz, but there is no maximum limit explicitly stated, and several online sources propose an upper limit of around 300 kHz. In any event, it works, so ... using Ohm's Law: If = (Vcc - Vf)/R — where Vcc = 5V, Vf = 2V (as per the datasheet), and R = 22 ohms, we can see that R1 and R2 limit the current to the IR LEDs to 136 mA — well below the maximum current of two amps allowed. I actually think we could get a much greater distance by pulsing the LEDs at a higher current provided the modules were adequately heatsinked, but I only wanted to push these as hard as necessary to get the job done. The less current allowed through the LEDs — even when pulsed — the longer they will last.

To amplify the optotransistors (IR_Q1 and 2), I'm using a TIP122 NPN Darlington transistor Q1. R4 from the base of Q1 to ground adjusts the sensitivity and ensures that when the beam is broken, the base is fully grounded and not left floating. The output of this is fed through R6 to the base of Q3 — a small signal NPN transistor which is tied to the junction of a voltage divider formed by R8 and R9. With the beam unbroken, this junction is held low by Q3. However, once the beam breaks, Q3 is turned off, and R8 brings this junction high.

In order to keep the system from repeatedly triggering



■ **FIGURE 1. Schematic diagram.**



■ **FIGURE 2. The completed circuit board.**

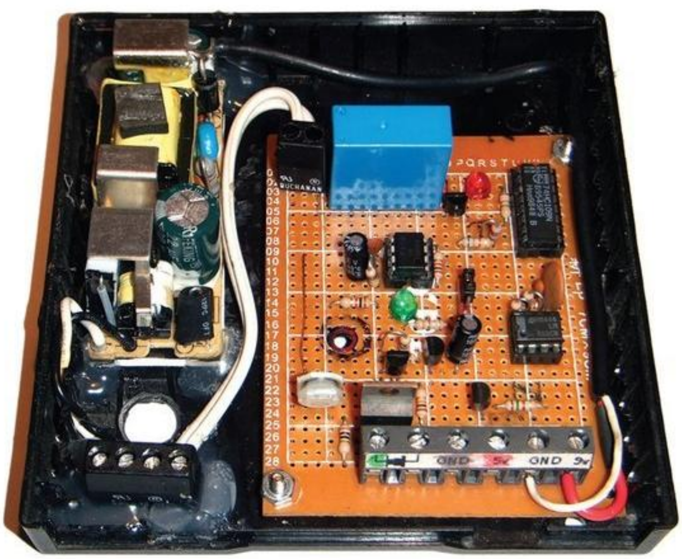
■ **FIGURE 3. A Westell modem waiting to breathe new life.**

as you walk through the beam, we employ a second 555 timer IC2, this time configured as a "one-shot" monostable multivibrator whose output pulse width is set to a little over one second by R16 and C5. If a longer or shorter delay between detection pulses is desired, R16 can be replaced with a higher or lower value or — for more versatility — a trimmer pot. LED2 is just for diagnostics and shows that the timer output is indeed going high and the length of that pulse.

IC2 requires a negative going pulse to trigger a positive output, and since the output of the junction between Q3, R8, and R9 goes high when the beam is broken, we need to invert this. Therefore, this signal is fed through R10 to Q4 — another small signal NPN. R15 holds the input of IC2 high until the beam is broken, which places a positive signal on the base of Q4, bringing the input of the 555 to ground. C2 smooths out any potential spikes that might falsely trigger the timer.

The output of IC2 is then fed to the input of IC4 — a J-K flip-flop. With PRE, CLR, and J tied high and K tied low, for every positive pulse received on the CLK input from timer IC2, pin 7 of the flip-flop alternates between high and low. R18 ensures that the clock pin is held low when not being triggered by IC2.

The output of IC4 is finally fed through R19 to Q5 — our final NPN — which, in turn, toggles SPST relay K1 on and off.



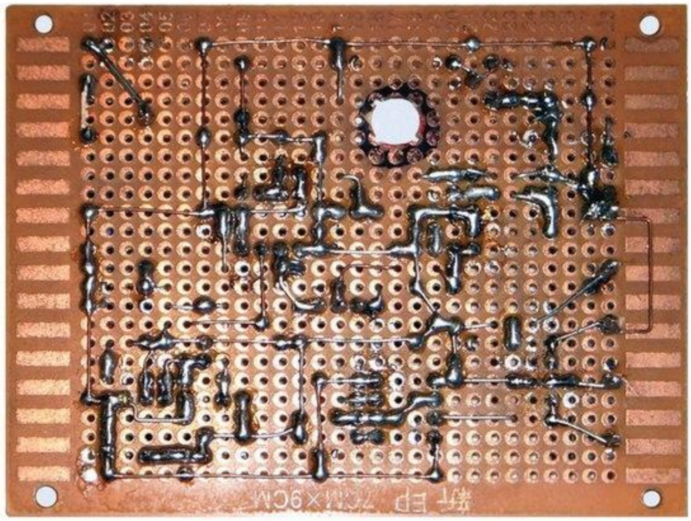
■ **FIGURE 4. Everything mounted in the modified modem housing.**

The Fun Part or Hooking It All Up

I prefer re-purposing the things that other people would throw out whenever I can, so most of my projects are built around whatever I happen to have on hand at the time — the IR pairs being the sole exception in this case. With that in mind, I built my system (**Figure 2** and **Figure 3**) into an old gutted Westell DSL modem case. Hot glue holds the power supply and the screw terminals down, while two 4-40 stand-offs hold the main board. As you can see from **Figure 4**, everything fits perfectly and neatly. Of course, any plastic housing would be fine — building this into a two- or three-gang wall box mounted in the wall with the original light switch is a wife-friendly option.

Rather than reinvent the wheel and spend more money, I have found that splitting orphaned wall wart transformers open is an excellent source of cheap, ready-

■ **FIGURE 5. A wall wart supply — before and after.**



■ **FIGURE 6. A well laid out perfboard.**

made power supplies when the use of an external supply is not possible or desired. These can be found almost anywhere for next to free. Just place the supply in a bench vice right on the seam and apply slight pressure. Most of the time, you'll hear a 'pop' and the glued seam can easily be pulled apart. You may have to put pressure on a couple of spots, and occasionally you may have to help it out with a flathead and a hammer, but it's just like shucking an oyster! **Figure 5** shows the before and after of this painless operation.

My original schematic called for a higher voltage with lower current for the IR LEDs I was using at the time, so I initially designed this around a 9V power supply fed into Vcc at X3-1 with a 7805 5V regulator to power the rest of the circuit. Pulsing the LEDs at the higher current, however, meant that the 9V supply (a small 200-300 mA type) could no longer supply the necessary power, and the regulator was struggling to keep up — even after adding a heatsink. Both were running way too hot.

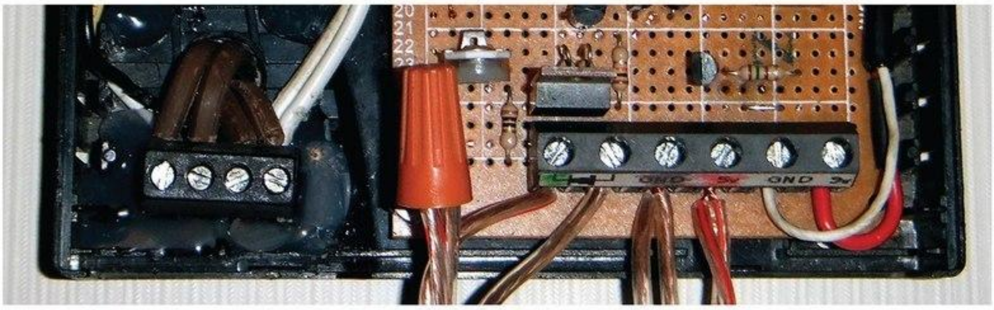
Since everything is now powered by 5V anyway, I found a 5 VDC 2.5 adapter in my wall wart drawer (everybody has one of those, right?) and did away with the regulator. Of course, you are welcome to try the 7805 with a 9V supply if you like, but I highly suggest the use of a 5V supply rated at minimally 1.5 amps (if not higher) fed into Vcc instead. If you are using the printed circuit board (PCB) pattern and go with the 5V input, just add a jumper across pads 1 and 3 of IC3 and leave that component out. Of course, a carefully laid out perfboard works very nicely, too (**Figure 6**).

*** WARNING ***

Before going any further, I would be remiss in this day and age if I didn't add the obligatory "don't try this at home" warning, so ... this design is perfectly safe so long as you understand AC line voltage and are comfortable attaching devices directly to the AC line. The only two potentially dangerous points on this circuit

PARTS LIST

Semiconductors	
Q1	TIP122 NPN Darlington
Q2-5	2N3904 NPN transistor
IC1,2	LM555 timer
IC4	74LS109A J-K flip-flop
D1	1N4004 rectifier
LED1	3 mm Red LED
LED2	3 mm Green LED
IR_LED1,2	OP298
IR_Q1,2	OPB598
• IR_LED and IR_Q sold in matched pairs — Digi-Key # OPB100Z	
IC3	LM7805 (see text)
Resistors (all 1/4 watt)	
R1, R2	22 ohm
R3	10 ohms
R4	200K trimmer
R5, R9, R10	10K
R6	4.7K
R7	150 ohms
R8, R11, R13, R15, R18	1K
R12	2.2K
R14, R17	180 ohm
R16	100K
R19	6.8K
R20	100 ohms
Capacitors	
C1	.001 µf
C2,5	10 µf @ 16V
C3,4	.01 µf
Misc	
K1	5 VDC SPST relay – Digi-Key # 255-3442-ND
JP1	Three-pin male header
X1	Two-position screw terminal — Digi-Key # ED2609-ND
X3-5	Two-position screw terminal — Digi-Key # ED2653-ND
5 VDC @ 2A power supply	
Speaker wire (to run emitter/receiver pairs)	
Small perfboard (for mounting emitter/receiver pairs)	



■ **FIGURE 7. Close-up of the terminal connections.**

are the power supply connections to the AC line (if you opt for the internal supply method as I did) and the connection from the light switch to the relay.

When making these connections, make SURE that the circuit breaker feeding the light switch is turned off. The safest way to do this is to turn the light switch on where you want this installed, and flip the breakers one by one until that light goes out. Place a piece of masking tape across the breaker to prevent anyone from accidentally resetting the breaker while you are working. If you have a no-contact voltage detector, double-check that the line is dead using that. It is always better to be safe than sorry — especially when working with line voltage. Once the circuit is fully enclosed, there is no danger of electric shock. With all of that out of the way, let's get back to the installation.

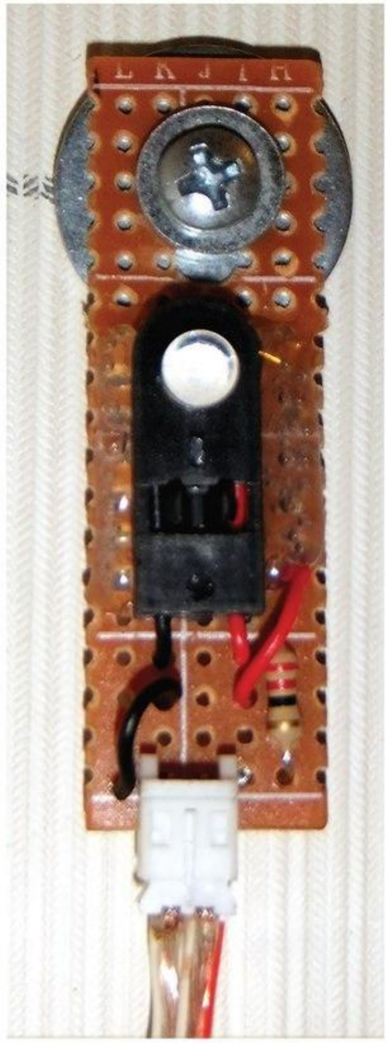
All of this was mounted directly below the light switch (the modem case conveniently provided two wall mounting screw slots) and the AC lines were run through the wall under the case. I tied my power supply directly to the AC wiring since it was handy, but if you are using an external supply, you can just plug it into an existing outlet. The contacts of K1 are wired across the existing light switch with a short length of standard lamp cord.

Wiring across the existing switch instead of replacing the original switch with this circuit allows the system to be overridden and allows the light to be kept on regardless of the state of the circuit. Using screw terminals for the AC connections makes wiring easy and safe. After the AC lines are connected and the switch is mounted back in the wall, you can turn the breaker back on in preparation for the alignment procedure (**Figure 7**).

The Hard Part or Lining Up the Sensors

The Optek sensors are designed to be snapped into mounts facing each other, but since I am mounting them to a wall, there really isn't an easy way to attach them. What I did (**Figure 8**) was to hot glue the sensor to a small piece of perfboard with a hole drilled in the top. Screwed into a sheetrock anchor, this allows stability as well as a place to solder the current-limiting resistor (in the case of the LED) and the connector. Once you have determined which doors you'd like to cover, mount the LEDs on one side of each door about waist high. Now, it gets a little more difficult.

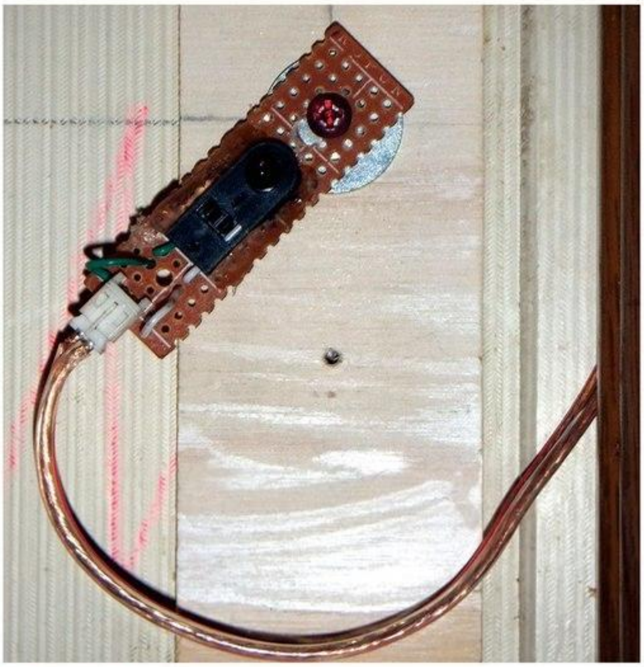
JP1 allows the initial alignment of the beams without triggering the circuit continuously, causing the light to flicker on and off. With pins 1 and 2 jumpered, the 555 trigger IC2 is held high, and instead LED1 lights when the receiver and the emitter are aligned and extinguishes when there is a misalignment.



■ **FIGURE 8. This is how the emitter should be mounted.**

Only connect one receiver at a time across X5-1 and X5-2 while performing the setup, otherwise you'll never get the sensors aimed properly. Both would have to be perfect for the LED to light. To avoid any external light source from giving a false positive, turn off all the lights while performing this step and work by flashlight. Adjust R4 to about its middle position. Now, hold the detector (mounted on its perfboard) directly across from the LED and slowly move it around until LED1 shines the brightest. Mark that spot, secure the board to the wall, and repeat for the other detector. This may take a little time and patience. I know it did for me!

Once both receivers are correctly mounted (don't do what I did — align the sensors BEFORE mounting or you'll wind up with **Figure 9**!), connect both receivers in series across X5-1 and X5-2 using a small wire nut to connect the emitter of one to the collector of the other. With both phototransistors connected in series, you may find that LED1 no longer lights. You can adjust the sensitivity by



■ **FIGURE 9. How NOT to mount the receiver! Live and learn.**

rotating R4 until LED1 just goes out and then back off R4 until LED1 comes back on. Remove the jumper from pins 1 and 2, and move it to pins 2 and 3 instead for normal operation.

If all has gone according to plan, you should now be able to walk your hallway without ever having to turn on another light again!

In Conclusion

Now that my hallway is automated, I plan on building additional modules to control other rooms in the house. The laundry room was built with the same problem — you have to walk across the room in the dark before you can turn on the light. In the bedroom — where I'd want the light to come on when I enter but turn off when I'm ready to go to sleep — I'll add a SPST momentary pushbutton switch between pin 2 of IC2 and ground. Then, when I'm ready to turn in for the night, I just push the button and ... lights out! In the master bathroom, there's a double-door that stays closed (due to el gato). There, I will replace the IR detector with a magnetic reed switch pair, increase the delay time constant for IC2, and do away with IC1 and its related components. As you can see, the base circuit is quite versatile.

Don't think that lights are all you can control with this! The relay I used (scrounged from an old CRT TV chassis) can switch loads up to 10 amps at 250 VAC or 24 VDC. Depending on the exact relay you select, your mileage may vary. Hopefully you found this interesting, or at least useful! **NV**

By E. Paul Alciatore III

Fax On! Fax Off!

THE FAXER

The printer/copier/fax machine in my home office shares the phone line with normal phone service. The phone service is through the cable company, so the phone line originates at their signal box which is also sitting on the desk. If the fax machine is left connected to that line, things don't work well. It answers all calls — fax or not — and it also seems to prevent incoming calls to my cell phone and perhaps from others.

There must be some combination of settings to make all this work together, but the fax is not used often and my time is limited, so I have been just connecting and disconnecting the phone lines on the rare occasions when actually sending or receiving faxes. The jacks for these connections are not conveniently located, so I frequently forget to disconnect the line when the fax is done. This yields complaints from the other frequent user of the phone: my wonderful wife. So, a hardware solution seemed to be the most practical.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/december2015_Alciatore.**

■ **Photo 1. The parts.**

■ **Photo 2. Mock-up.**

The Faxer

This is a switch that allows the phone line to be either connected to the fax machine or to be bypassed, so the desk phone will work normally without problems. Of course, a somewhat neat installation was highly desirable.

The necessary components were purchased at a nearby home supply store and at a local electronics supply company for the switch itself. They can also all be easily obtained on the Internet (**Photo 1**).

The two-port surface box holds two modular RJ-25 phone connectors which provide connection points for the phone line and desktop telephone, and has a nice appearance. It came with double-sided adhesive pads which made it simple to attach it to the base of my printer for easy access.

The 25 foot phone cord — with connectors — was cut to provide the two cables that run to the fax connections on my printer. The remainder of the wire went to the scrap box for another project.

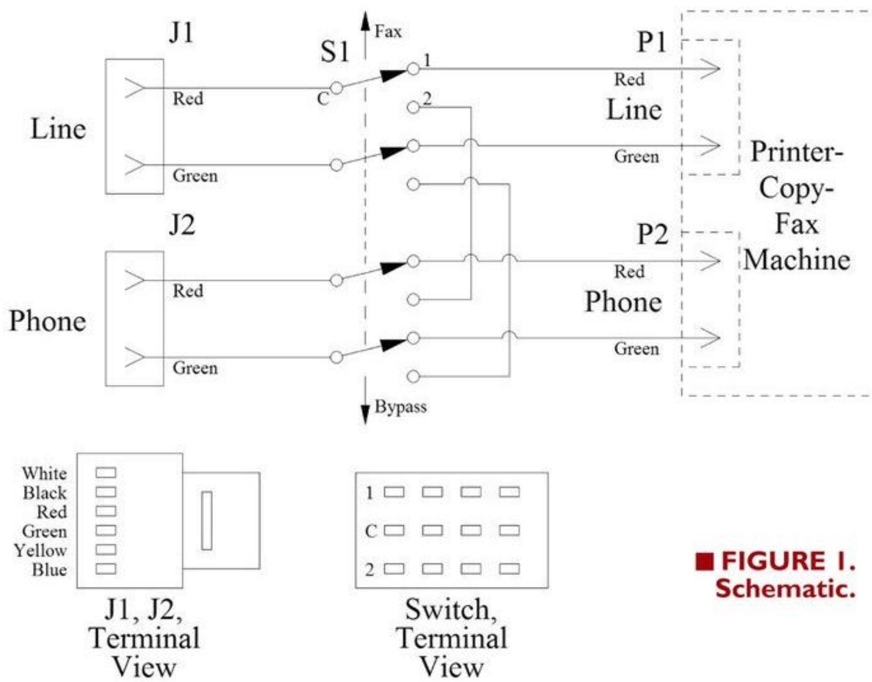
The first step was to do a quick mock-up to confirm the needed clearances when the parts were assembled (**Photo 2**).

This confirmed that there was sufficient room for the wiring, so it was "all systems go."

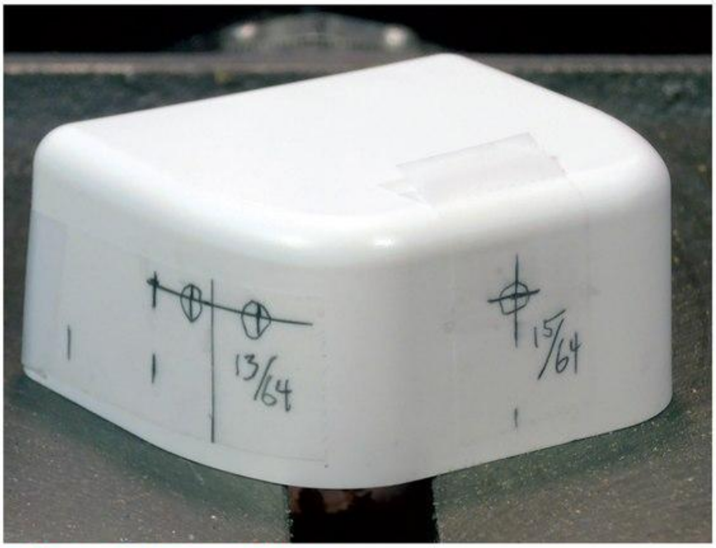
The original wiring diagram was done quickly on a scrap of paper, but **Figure 1** is a neater version made with CAD. It also includes base diagrams of the jacks and switch, but confirm that yours will match before wiring.

Several holes are needed in the plastic cover for mounting the switch and for the cables to the fax machine to pass through. By using hard wired cables for the connections to the fax machine instead of two more jacks, much space is saved and the overall device is kept smaller. Plus, the cost of two jacks and two connectors is saved.

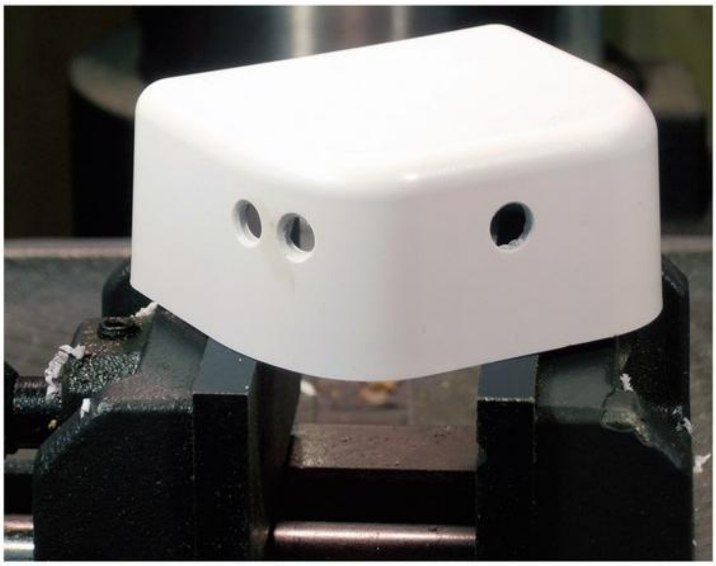
In **Photo 3**, the locations for these holes have been carefully determined and laid out on Scotch Magic Tape™



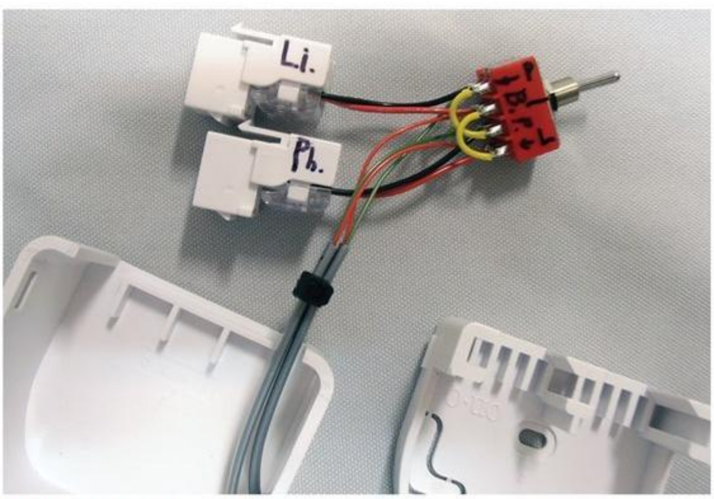
■ **FIGURE 1. Schematic.**



■ **Photo 3. Layout of holes.**



■ **Photo 4. Drilled and cleaned holes.**



■ **Photo 5. The wiring.**

that has been placed over their locations. This is not the most accurate way to locate the holes, but this device does not need a lot of accuracy so it's just fine. Be sure to take the interior features and location of the jacks into account when doing this. Each hole location was punched to make a dimple which helps to start the drill on center. The holes were drilled on a drill press, starting with a 3/64" drill to make small pilot holes and then with the final size drill (as shown in **Photo 3**) which follows those pilot holes. When using this two drill procedure, the location of the final hole will be accurate to a small fraction of the size of the small pilot drill.

The plastic housing was held in a drill press vise to prevent it from being grabbed by the drill and dangerously spun around. The drill was fed downward slowly. A hand drill could also be used with the cover held in a bench vise. The 15/64" hole is just a few thousandths too small for the stem of the switch, so a round file was used to enlarge it just a bit. About a half dozen strokes were enough to do this. A countersink was used to remove the sharp edges of the holes both inside and outside, and to bevel the edges of the cable holes (**Photo 4**).

A trick was used in mounting the switch. Most toggle switches have a keyed washer that prevents the switch from rotating in its hole. These keyed washers have an internal key that fits in a key slot on the barrel of the switch and a bent tab on the outside diameter that is intended to fit in a small hole drilled near the main hole for the switch. In spite of careful measurement and drilling, this bent tab often does not fit properly, so a different method was used in place of drilling.

The switch was placed in the hole in the case and the keyed washer was placed on it on the outside, with the tab resting against the plastic. Then, the nut was put on just finger tight so it would not move around. A hot (600º F) soldering iron with a small tip was placed on the washer just above the bent tab, and light pressure was applied until the tab melted in and the washer was flat on the surface of the cover.

Be careful not to go too far; a gentle touch is best. This worked perfectly and the appearance is very nice. It looks a lot better than a drilled hole. I wish I had thought of this trick about 30 years ago.

The wiring is done on the switch first, one row at a time. The two sides of the switch are labeled "Fax" and "B.P." (bypass) to help. Keep in mind that most toggle switches make contact from the common terminals in the center row to the terminals on the side row that is OPPOSITE to the direction that the switch is thrown. This can be confirmed with an ohmmeter. The first row that is wired is side #2 on the schematic, or the bypass side. Two yellow jumpers on this row can be seen in **Photo 5**.

Next, four wires are soldered to the terminals of row C on the switch, or the center/common row. It would have been best to preserve the red/green color scheme,

PARTS LIST

QUANTITY	ITEM	DESCRIPTION
1	Two-port surface box	
Pk/5	J1, J2	RJ-25 phone jack to fit above the surface box
1	P1, P2	Modular phone cord (25 foot) with modular connectors
1	S1	Miniature toggle switch, four-pole, double-throw, On-On

but no green wire was available, so red and black were used instead. These leads should be about 1.5 inches long. Their other ends are left unconnected for now.

The two cables that go to the fax machine need to be soldered to the fax machine row, labeled #2 in the **schematic**. First, these two cables must be cut to the proper length and brought in through the two holes in the top of the case, one cable per hole. A very tight cable tie (the black object in the photo) is used to form a strain relief. This both absorbs any force on the cables and prevents them from rotating in the holes, so the internal connections are safe.

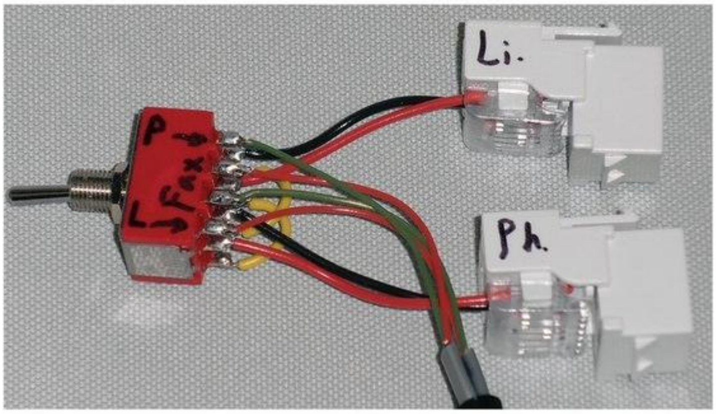
Standard telephone cable has four conductors: black and yellow at the outside two positions; and red and green at the inside two. The red and green inside conductors are designated for use as Line 1, and virtually all phones with a single line as well as virtually all fax machine connections will use them. So, the black and yellow wires (Line 2) are simply cut off; we only use the red and green ones. The two phone cables were labeled with a Sharpie pen at both ends, with "L" for Line or "P" for phone to keep them straight. That completes the wiring on the switch.

You can see the switch and the jacks are also labeled for Phone and Line. A mistake was made in connecting the jacks in the photo and it was not discovered until the final assembly, so disregard the markings on the jacks in the photo. They were correctly labeled on the outside after assembly.

Those four short wires on the C row of the switch are punched into the terminals on J1 and J2, labeled red and green. These should be the two center terminals. Instructions for this are included with these connectors. Keep in mind that this kind of insulation displacement connection relies on the wire being of a proper size. Although the instructions did not specify what this size should be, it should probably be between AWG 22 and AWG 26. That completes all the wiring (**Photo 6**).

Snap the case together with care so no wires are pinched. A quick label was generated with a CAD program and installed with double-stick tape. As mentioned, the jacks in the rear were marked "Line" and "Phone" with a Sharpie marker. Finally, the two cables to the fax machine were laced together with waxed dental floss for a neater appearance. The switch was installed with the double-stick foam squares that were provided, and worked perfectly when tested.

Now, faxes can be easily sent, and there is less chance of leaving the fax machine on the line when they are done. **NV**



■ Photo 6. Completed wiring.

THE SMART DOORBELL –
A Simple Programmable Relay Project

This article introduces programmable relays as a means of building a smart doorbell. Programmable relays are microcontroller-based devices that can replace older hard-wired electromechanical control systems. A single programmable relay can replace an array of relays, timers, and counters.

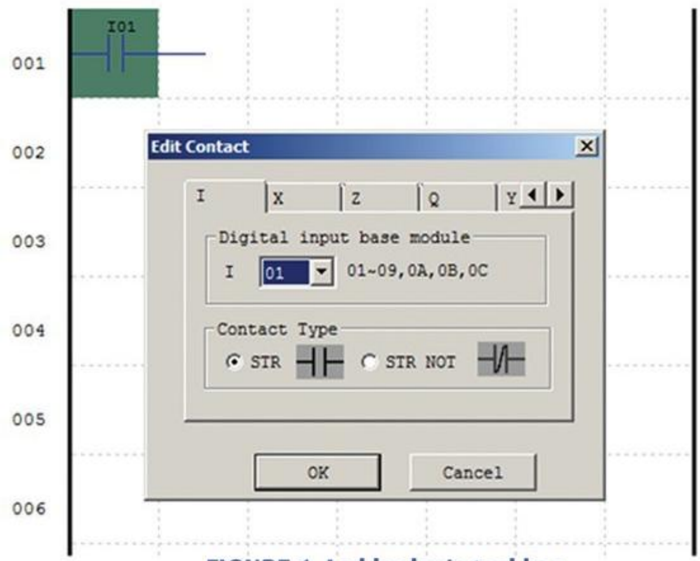


FIGURE 1. Ladder logic tool bar.

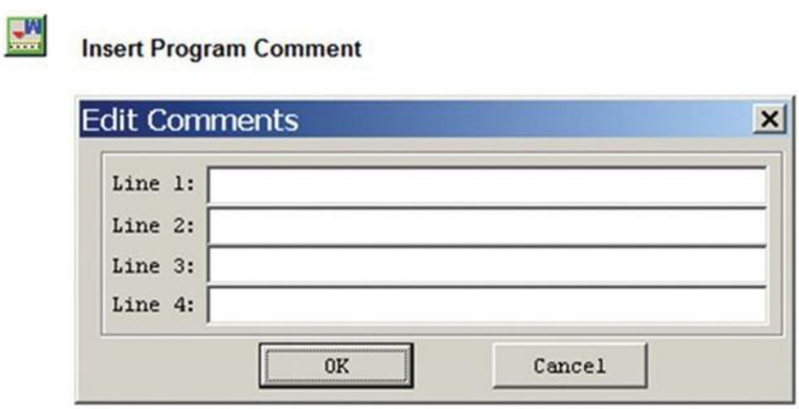


FIGURE 2. Program comments added to ladder logic provide helpful information.

Programmable relays use a visual language based on relay logic. Virtual devices are selected from a toolbar, configured from a drop down menu, and positioned on a ladder diagram between the power rails to achieve the required functionality (**Figure 1**).

The TECO SG2 Client Software Version 3.37 illustrated in this article is downloadable for free from **http://teco.us.com**.

The visual ladder logic language was developed to ease the transition of people familiar with older electro-mechanical systems to newer microcontroller-based control systems.

The software allows for the simultaneous monitoring of multiple inputs, outputs, and virtual devices. Control functions can be complex, so the addition of comments into the ladder logic program can help to explain the workings of the system when in monitor mode (**Figure 2**).

Monitor mode allows an online computer to display the status changes of all inputs, outputs, and virtual devices as they occur.

The system designed in this article is a smart doorbell. It produces different ring patterns depending on when the door bell is pressed. The system utilizes a standard door bell and buzzer (**Figure 3**).

The intelligence is added to the system by the programmable relay. The programmable relay used in this article is a TECO SG2-12HT-D. It is available from FACTORYMATION for $94. Additionally required is a SG2-Ulink cable for $33. Both are available at **www.factorymation.com**.

The status of the input and output terminals are indicated on the programmable relay's visual display.

The display in **Figure 4** shows that both input I01 and output Q01 are energized.

By Richard Agard

Post comments on this article and find any associated files and/or downloads at www.nutsvolts.com/index.php?/magazine/article/december2015_Agard.

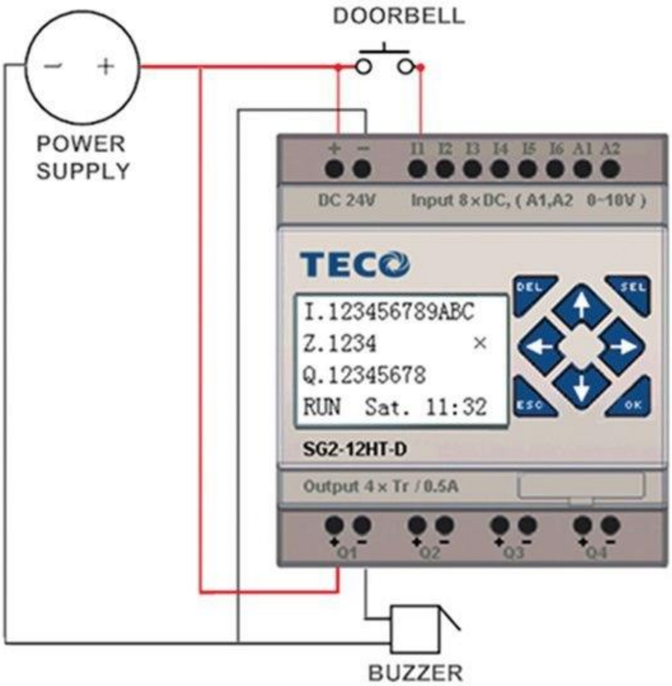


FIGURE 3. The smart doorbell utilizes a standard doorbell and buzzer.

FIGURE 4. Status display shows I01 and Q01 energized.

Smart Doorbell Listing of Circuit Requirements

1. When the doorbell is pressed M01 Latches on allowing 5 pulses to ring the buzzer
2. If doorbell is held on for more than two seconds T09 shuts off the buzzer after 2 rings
3. If door bell is pressed again on the third ring the buzzer goes into the first irregular pattern produced by M02, T06 and T08
4. If door bell is pressed on the fourth ring, the buzzer goes into the second irregular pattern produced by M03, T07 and T08

FIGURE 5. Smart doorbell listing of circuit requirements.

The software was developed from the *Listing of Circuit Requirements* and refined by means of off-line simulation. Simulation mode allows a designer to test the software at various points to confirm operation before downloading the program into the programmable relay. The exact settings used in the smart doorbell program were not calculations. They were determined experimentally offline in simulation mode (**Figure 5**).

The complexity of a programmable relay based system is in the programming. Programming and future troubleshooting will be easier if the project is well organized. Programming spreadsheets help with this organization. Programming spreadsheets include: a brief description of the function of a circuit; a listing of inputs, outputs, and virtual devices; device settings; circuit requirements; and a rung by rung explanation of the program's operation.

The top of the programming spreadsheet in **Figure 6** shows that there are two external devices in the smart doorbell project: a doorbell switch wired to input I01 and the buzzer wired to output Q01.

Programming of the system is broken down into a set of parallel rungs, with each rung performing a particular function. In **Figure 7**, the momentary closure of the doorbell switch closes NO contacts I01 on rung 002, energizing internal relay coil M01 and timer coil T01, and closes NO contacts T01. T01 and M01 are latched ON until timer T01 times out in seven seconds.

Programming Spread Sheet

Function:	Utilizing a standard doorbell and buzzer the Programmable Relay produces different ring patterns dependent on the time the door bell is pressed

Ext. Devices Output	Output Pin #	Contact Type	Voltage	Ground	Rungs
Buzzer	Q01+	relay	24VDC	Q01-	28

Ext. Devices Input/Contact	Input Pin #	Internal Coils	Internal Relay Contacts	Timer Coils	Timer Contacts	Rungs
Door Bell (DB/SW) /NO	I01					2, 5, 13, 16

FIGURE 6. Top of programming spreadsheet.

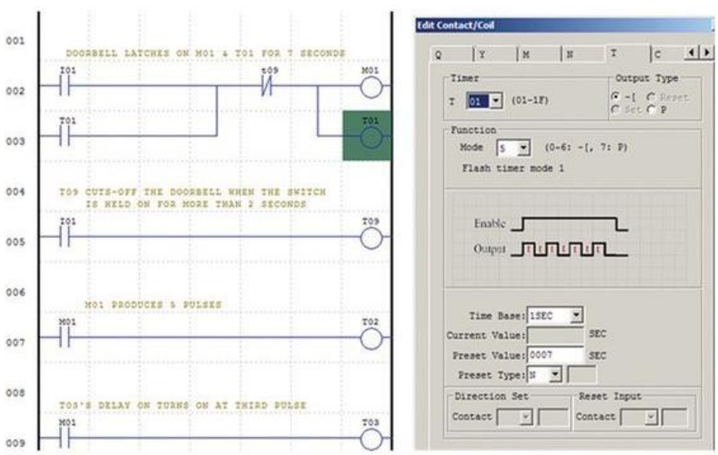


FIGURE 7. Rungs 001-003.

The proper configuration of T01 is shown in the drop down box on the right side of **Figure 7**.

If the doorbell is held for more than two seconds, timer T09 disconnects the doorbell. The doorbell will be disconnected after two rings and stay disconnected until the doorbell switch is released.

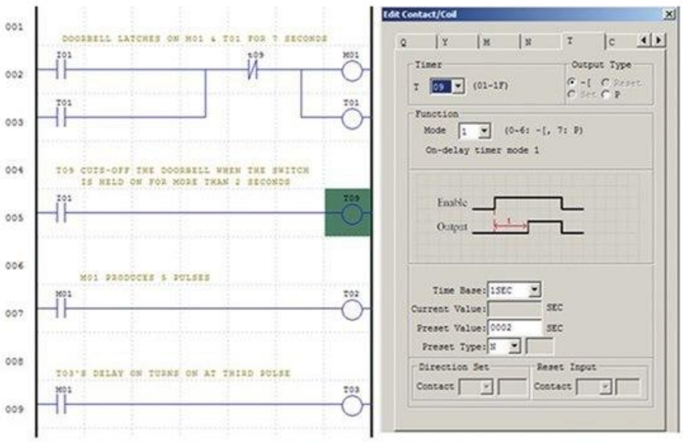


FIGURE 8. T09 disconnects the doorbell switch after two seconds.

The closing of NO contacts M01 on rung 005 enables T02 for seven seconds, giving T02 the time necessary to produce five pulses.

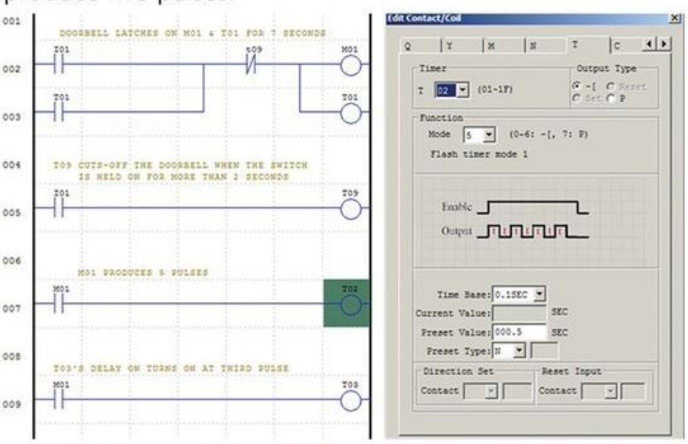


FIGURE 9. Rungs 004 and 005.

T02 on rung 028 pulses the buzzer five times when the door bell is pressed.

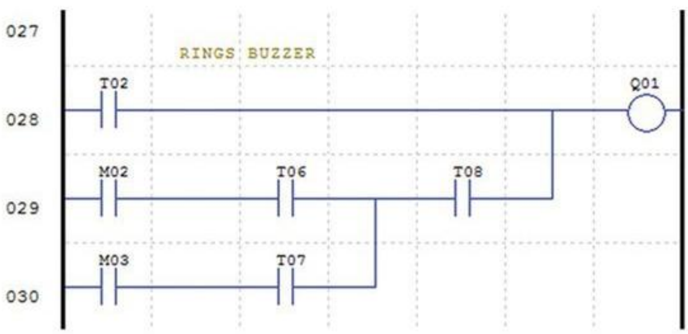


FIGURE 10. T02 rings the buzzer five times.

T03 on rung 009 is timed to turn on at the third pulse.

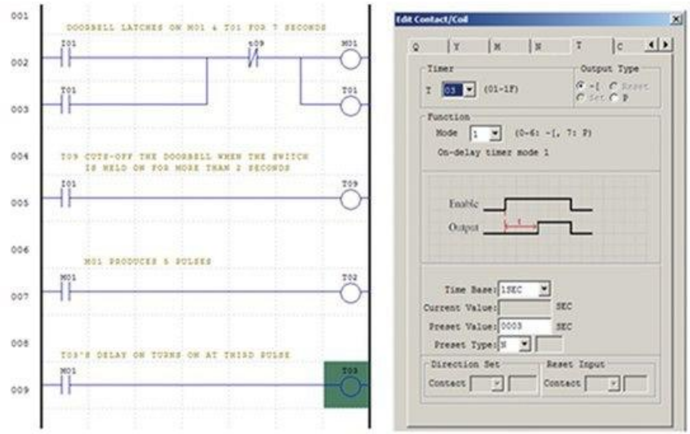


FIGURE 11. Rungs 008 and 009.

T04 on rung 011 is timed to turn on at the fourth pulse.

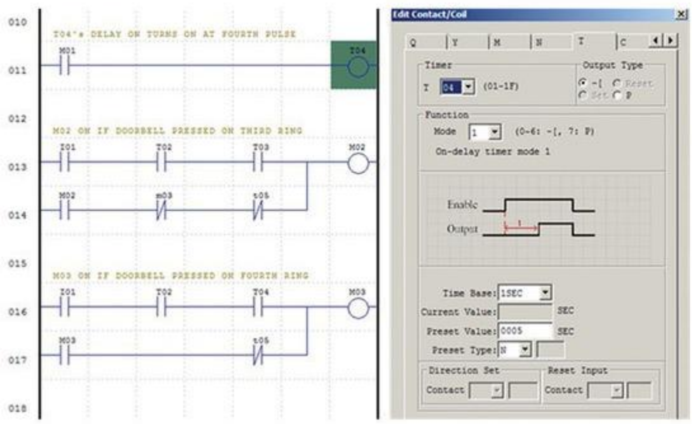


FIGURE 12. Rung 011.

M02 on rung 013 is on if the doorbell is pressed for a second time on the third ring.

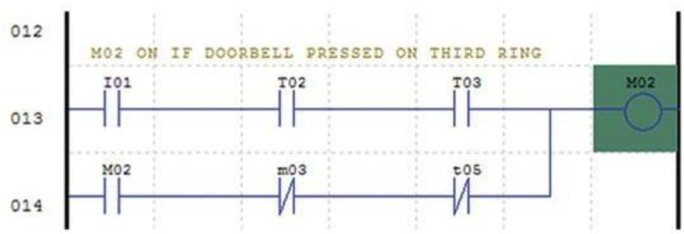


FIGURE 13. Rungs 013 and 014.

M03 on rung 016 is on if the doorbell is pressed for a second time on the fourth ring.

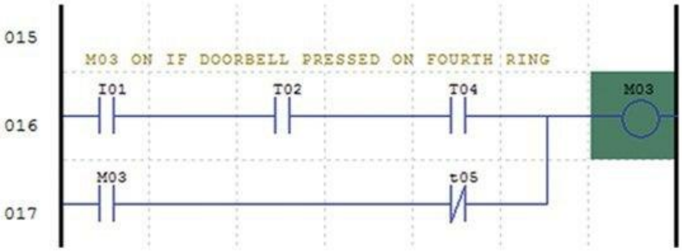


FIGURE 14. Rungs 016 and 017.

T05 on rung 019 is on if the doorbell is pressed during either the third or fourth ring. It causes M02 or M03 to stay latched on for 12 seconds, allowing adequate time for the specialty rings.

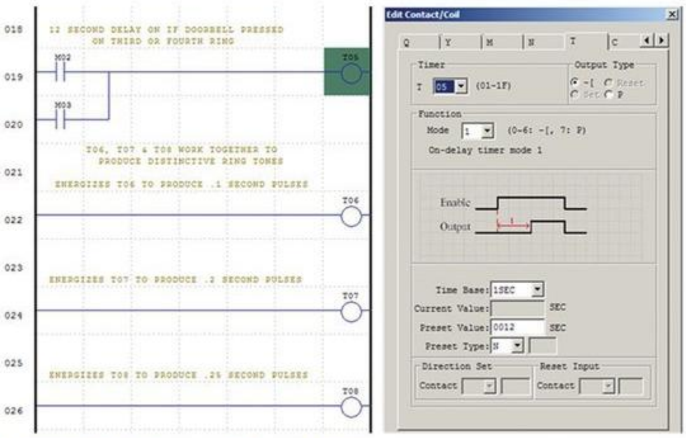


FIGURE 15. Rungs 019 and 020.

T06, T07, and T08 produce different period pulses. They are used together to produce complex identifiable ring patterns when the doorbell is pressed on the third or fourth ring.

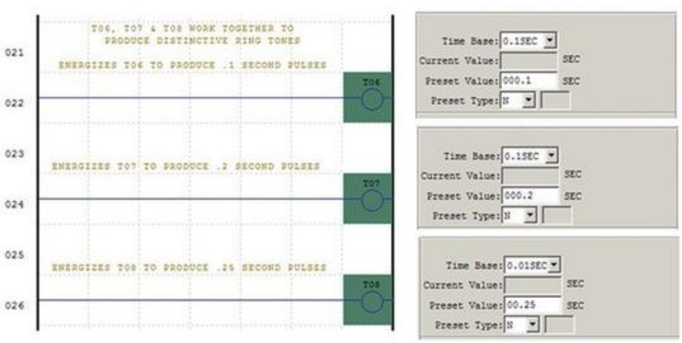


FIGURE 16. Rungs 022, 024, and 026.

The sound of the buzzer is modified on rungs 029 and 030 where the three pulses are mixed to produce the more distinctive pattern.

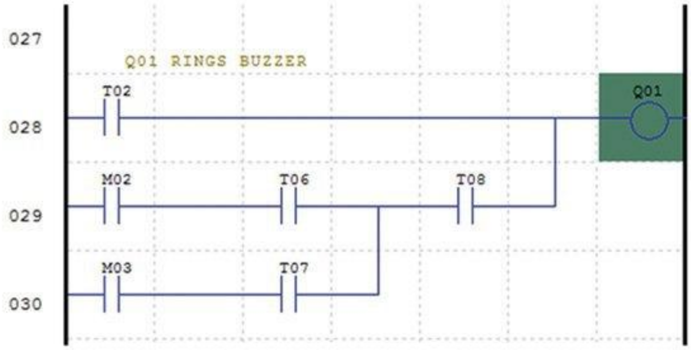


FIGURE 17. Rungs 028, 029, and 030.

The programming spreadsheet includes a record of all the device settings determined during the design and testing of the project.

Device	SETTNGS			Rung
T01	Mode 5	Time base 1 sec.	Preset 7 seconds	3
T02	Mode 2	Time base 0.1 sec.	Preset 0.5 seconds	5
T03	Mode 1	Time base 1 sec.	Preset 3 seconds	9
T04	Mode 1	Time base 1 sec.	Preset 5 seconds	11
T05	Mode 1	Time base 1 sec.	Preset 12 seconds	19
T06	Mode 5	Time base .1 sec.	Preset .1 seconds	22
T07	Mode 5	Time base .1 sec.	Preset .2 seconds	24
T08	Mode 5	Time base .01 sec.	Preset .25 seconds	26
T09	Mode 1	Time base 1 sec.	Preset 2 seconds	5

FIGURE 18. Smart doorbell settings matrix.

Programmable relay systems are electrically simple but operationally complex. The wiring is tested by the observation of indicators on the front display.

The programming spreadsheet allows the designer to organize the project, plus it provides valuable documentation for the future when technicians have to repair the system. **NV**

Building a Propeller-Based Mentoring Platform

By Dane Weston

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/december2015_Weston**.

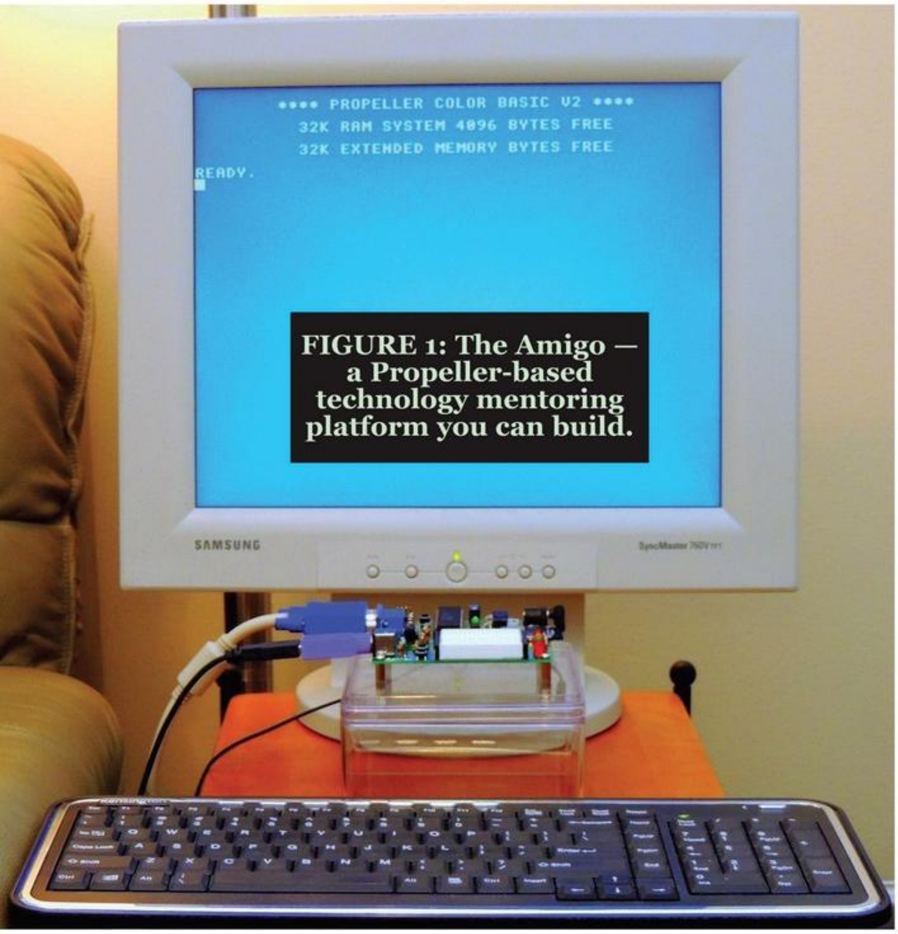


FIGURE 1: The Amigo — a Propeller-based technology mentoring platform you can build.

Are you looking for a way to contribute to the next generation of scientists and engineers, perhaps in your own family? What if you could build your own technology mentoring platform in a single afternoon, then spend hours of fun with your child or grandchild making simple computer games or exploring computer-controlled hardware circuits together? This article will explain my efforts to do just that, and then suggest how you can too (Figure 1).

Over the past few years, I've built several little "retro computers" to help young family members and friends learn introductory programming, and it's been fun and rewarding. I also wrote a book for parents and/or grandparents to guide the learning experience, mentoring their youngsters on the fundamentals (emphasis on "fun") of programming and computer-controlled hardware. Most of these "mentoring platforms" were built from kits based on the Parallax Propeller, and they used that amazing chip to host a tinyBASIC interpreter and drive all the I/O for a self-contained "old style" computer: keyboard, monitor, file storage, audio, and joystick.

My favorite of these kits was the Propeller Experimenter's Board (PEB 2014) which is a version of the "Pocket Mini Computer" from Jeff Ledger at **propellerpowered.com** — easy to build, full of features, and loads of fun. For me, the PEB 2014 and its Color BASIC "operating system" bring to mind the Commodore, Atari, and Tandy computers of the early 1980s, which were easy to understand and program, and were ready to go as soon as you hit the "on" button. (The PEB 2014 has just 4K of program memory, and to me is eerily reminiscent of my first TRS-80 — a 4K Color Computer.)

Those early machines were so much fun precisely because they were easy to understand and program, with no hidden layers of complexity or steep learning curves. They were directly responsible for the career choices of thousands of today's sweetly successful hardware and software engineers.

Hola Amigo!

I believe that this "simple and fun" formula still works. So, when the PEB 2014 was no longer available from Propellerpowered earlier this year, I was disappointed enough to design and build a simplified replacement that preserved the features most important to me: easy to build (perhaps with a youngster helping); easy to program (using Jeff Ledger's generously licensed Color BASIC); containing a breadboard with access to some Propeller I/O pins; and sporting an "old computer" monitor and keyboard.

I call the result the "Mentor's Friend," and nicknamed it "Amigo." In this article, I'll attempt to explain what the Amigo does, how it works, and how you can build one (**Figure 2**).

Figure 2 shows the features and interfaces of the Amigo. It uses an old PS/2 keyboard and VGA monitor as primary user interfaces, and comes with onboard connectors for a Wii classic controller (not the nunchuck) and earbuds or a powered speaker. The Propeller itself is a 40-pin DIP package (to allow easy DIY construction), with the standard 5 MHz crystal, reset switch, and Prop Plug connection. The EEPROM holds Color BASIC — the Propeller Spin binary that boots up a tinyBASIC interpreter and drives the various input/output channels via Prop I/O pins. The 2 GB SD card supports non-volatile storage of BASIC programs and data files, and the SRAM chip provides 32K of volatile data storage.

The Experimenter's Section contains a small breadboard (the same one used on several Parallax boards), two tact switches and four LEDs, and headers for access to Prop pins P0-P15, the switches and LEDs, and power and ground. The power supply provides 5 VDC and 3.3 VDC to all of the above as needed.

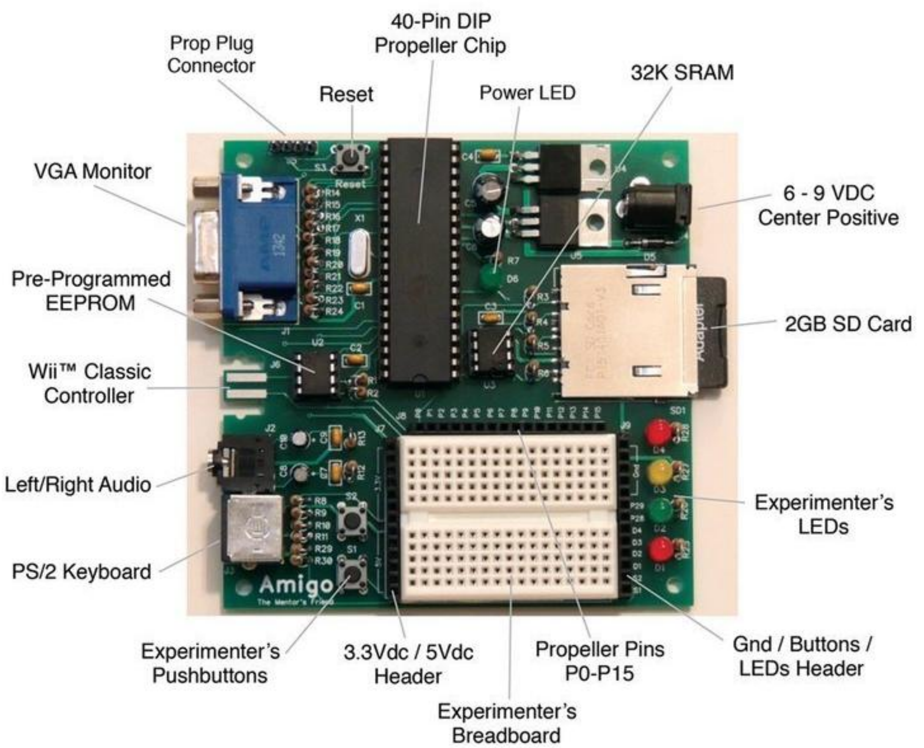


FIGURE 2: The Amigo, showing key interfaces and the Experimenter's Section.

(Way) Back to BASICs

The "secret sauce" of the Amigo is (as mentioned) Color BASIC: a Propeller Spin program with roots back to 2006 and a history of contributions by a plethora of Propeller gurus. Color BASIC was

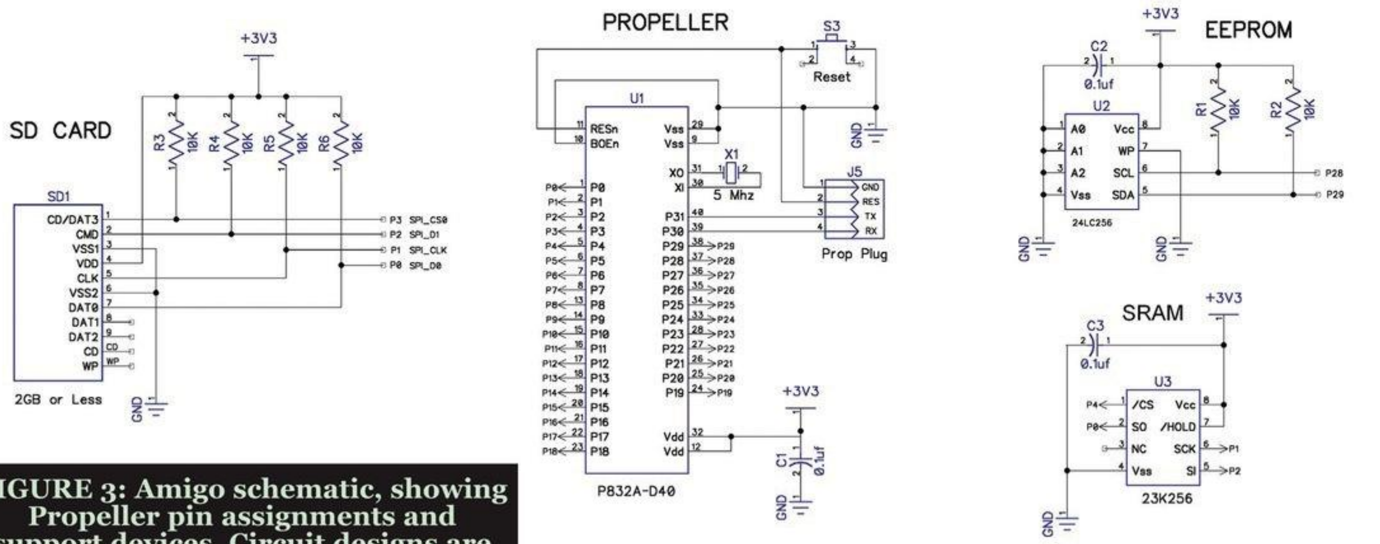


FIGURE 3: Amigo schematic, showing Propeller pin assignments and support devices. Circuit designs are "standard" for these subsystems.

begun in 2007 by Jeff Ledger as a software "fork" of FemtoBASIC — a version of tinyBASIC developed for the Propeller in 2006 by Mike Green and Tomas Rokicki.

Over the next few years, Jeff added and improved Color BASIC, with contributions from Green, Rokicki, and other Propeller notables like Marko Lukat, Jon Williams, Johannes Ahlebrand, Michael Park, and Andre' LaMothe. By 2012, Color BASIC was in use as the "operating system" of the Pocket Mini Computer — a self-contained "retro computer" that was a blast from the past for oldsters and an interesting technology throwback to youngsters who got their hands on it.

One thing that's different about the Amigo is that you can control Propeller I/O pins from Color BASIC instead of Spin — so you can write simple programs to control your homebrew circuits with a keyboard and a monitor without any other computer, application, or programming environment.

It's not sophisticated, and it certainly won't replace C, Spin, or Propeller Assembly Language, or those really cool boards from Parallax, but it is simple, understandable, and fun — perfect for a mentoring platform intended to introduce programming and computer-controlled hardware (**Figure 3**)!

Shown in **Figures 3-5**, all hardware subsystems in the Amigo are open source circuit designs from various Parallax boards or the PEB 2014 and its progenitors. So,

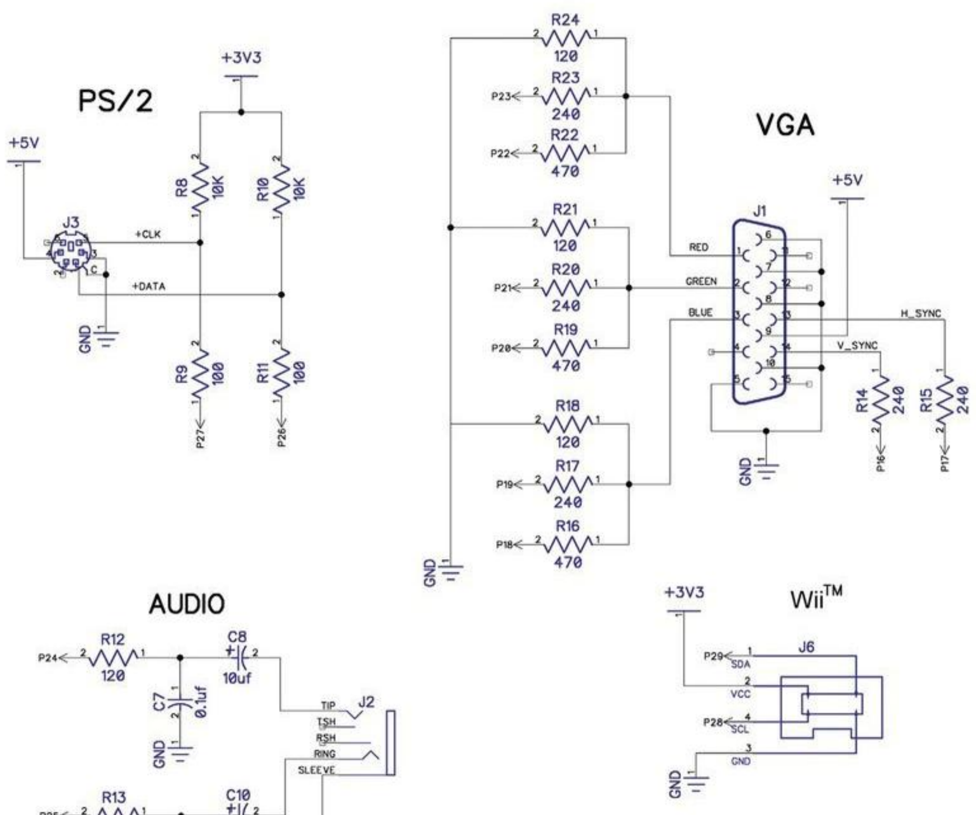


FIGURE 4: Schematic of Amigo peripheral interfaces. Again, circuit designs are standard.

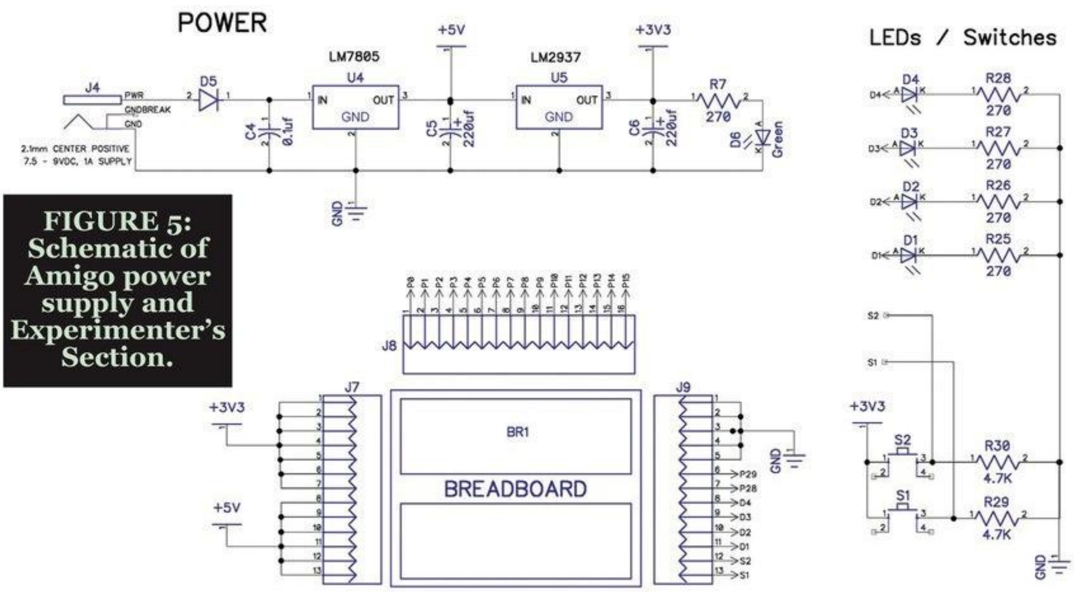


FIGURE 5: Schematic of Amigo power supply and Experimenter's Section.

after picking the subsystems I wanted and assigning Propeller I/O pins to each of them, the design challenge for this project became developing a circuit board layout that tied everything together, and was easy to build and easy to use. I enlisted DipTrace for board design and PCBWay for board fabrication, and can offer both my highest recommendation.

It was my first experience with DipTrace, so I had to climb the learning curve, but the tool suite is full-featured and provided everything I needed. PCBWay was a pleasure to deal with — excellent product quality, fast, and very reasonably priced. For the Amigo, I ended up with a 4" x 4" two-sided board with through-hole components for everything except the SD card socket, which was only available as surface-mount (**Figure 4** and **Figure 5**).

Build it Yourself or Build the Kit

There are several ways that you can create your own Mentor's Friend. The key is a Propeller platform with PS/2, VGA, and SD card capability that can run Color BASIC. If you already have a Propeller board with the needed peripherals or some version of the Propellerpowered Pocket Mini Computer, you should be able to download the Color BASIC Spin source from the article link, adjust the pin assignments to your specific board, then use your Propeller tool to compile the program and load the binary to your EEPROM. However, it may be more fun to use the Amigo kit available from the *Nuts & Volts* webstore — especially if you want to "build a computer from scratch" with a young friend!

The *Nuts & Volts* kit provides everything you need to build your own Mentor's Friend — just add a monitor, keyboard, and power supply, and you're in business. Kit contents include a 4" x 4" through-hole circuit board; a 40-pin DIP Propeller chip with associated crystal, EEPROM, and sockets; a 32K SRAM chip and socket; 3.3 VDC and 5 VDC power supplies; a 2 GB SD card and socket; a reset button; and the connectors and components for a VGA monitor, PS/2 keyboard, left/right audio channel, and Wii Classic Controller. The EEPROM comes pre-loaded with Color BASIC, so there's no need to worry about programming the Propeller.

In addition, the kit provides an onboard Experimenter's Section which consists of a small solderless breadboard, four LEDs, and two tact switches, and headers for

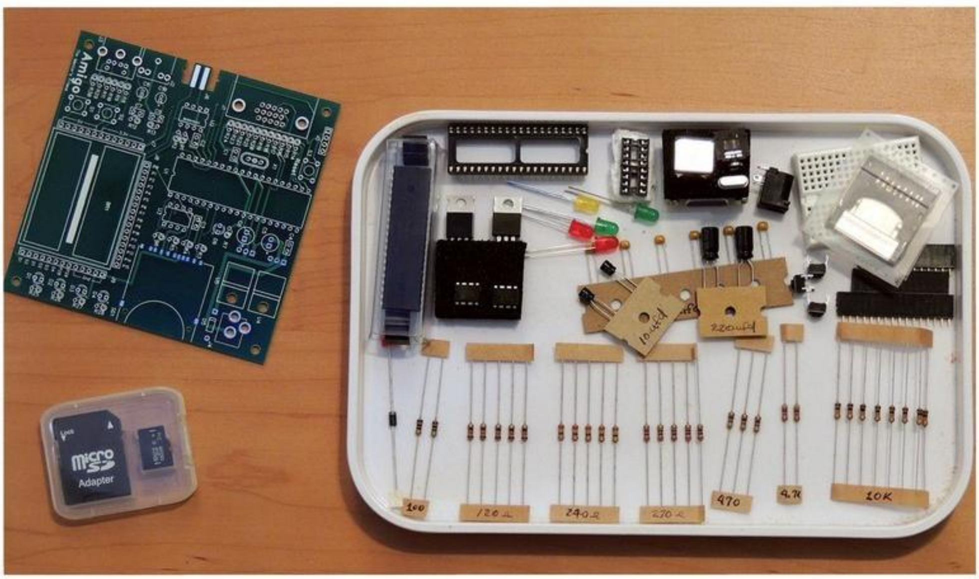


FIGURE 6: The *Nuts & Volts* kit includes everything needed to build your own Amigo, including a pre-programmed EEPROM and a 2 GB SD card with starter files.

access to 18 Propeller I/O pins, the LEDs and switches, and power and ground connections. The breadboard is small, but it's all you need for many computer-controlled hardware experiments. (In fact, it's the same one that's on several Parallax hobbyist boards.) **Figure 6** shows the Amigo kit and what comes in the box.

Since you are reading this magazine, the chances are excellent that you already possess the background and skills to easily build your own Amigo. The assembly instructions are meant to be complete and easy to follow, and they include lessons learned from builders before you. With the exception of the SD card socket, all parts are through-hole — not surface-mount — and the 4" x 4" board size allows plenty of room to work comfortably. I am a slow and methodical builder, but my 15 watt iron, magnifier, and I can easily finish an Amigo kit in a single afternoon, with plenty of time left over for some refreshments after the tools are put away.

Time for a Test Drive

So, once you've built your Amigo, what can you do with it? The first thing you'll notice is that Color BASIC is definitely tiny — only 26 variables, just 4K of program memory, no support for strings, 32-bit integer arithmetic only, no arrays, no explicit DATA statement. However, for an initial mentoring platform, those are all good things, not bad ones. Excluding the various operators, Color BASIC has about 50 commands. (Please see online resources for an overview of most Color BASIC commands. You'll be glad you did!)

It's fun and compact, and easy to learn and enjoy. It also includes everything you need to really understand the fundamentals of computer programming and hardware control, without the continuing disappointment of yet another layer of complexity to learn. With a little guidance, young students can truly "master" the Amigo, and with that mastery comes a self-confidence and "big picture" understanding that will serve them well going forward.

Here is an actual Amigo starter program to give you an idea of how simple Color BASIC is to understand and use. Once you get your kit built and see the READY prompt on your VGA monitor, press <F1> on your keyboard to start the editor, then enter this Color BASIC version of the classic first program:

```
10 REM *** HELLO WORLD! ***
20 COLOR 63,0 REM <~~~ WHITE TEXT, BLACK
                       BACKGROUND
30 CLS        REM <~~~ CLEAR THE SCREEN
40 FOR N=1 TO 100
50 A=RND(34)   REM <~~~ RANDOM COLUMN
60 B=RND(37)   REM <~~~ RANDOM ROW
70 C=RND(63)+1 REM <~~~ RANDOM TEXT COLOR
                        (NOT 0)
80 LOCATE A,B
90 COLOR C,0
100 PRINT "HELLO WORLD!";
110 NEXT N
120 PAUSE 3000 REM <~~~ WAIT 3 SECONDS
130 GOTO 20   REM <~~~ DO IT AGAIN
```

When you've entered the code, press <F1> again to return to Color BASIC and run your program. If all goes well, you should see lots of instances of "HELLO WORLD!" in different colors at random screen locations, followed by a short pause, then repeating. Now, press

History of the Pocket Mini Computer

The Amigo is a direct descendant of the Pocket Mini Computer (PMC): a project by Jeff Ledger that combined the work of several Propeller gurus to offer a series of retro computers running a version of BASIC. A bit of history may be helpful to understand the technical elegance behind this seemingly simple little device.

In 2012, Gadget Gangster offered the first official Pocket Mini Computer created by Ledger, who himself was a longtime Propeller project designer and more recently, the proprietor of propellerpowered.com. (You may have come across Ledger on various computer Internet forums under the nickname of Oldbitcollector. He has been a prolific designer, author, and contributor.)

The initial PMC combined a Parallax QuickStart board with an expansion board that interfaced a PS/2 keyboard, VGA monitor, Wii controller, audio jack, and SD card. In addition to the open source software objects that supported all those interfaces, the PMC software contained a tinyBASIC interpreter that Ledger and others extended from the original 2006 work of Tomas Rokicki and Mike Green.

The design challenge for Color BASIC was figuring out how to shoehorn all that capability into the eight cogs and 32K memory of the Propeller chip. Several "big brains" in the Propeller user community volunteered their expertise where needed to help make Color BASIC a reality, and in mid-201,2 the first PMC was on the market.

In 2013, Ledger and Propellerpowered released an updated PMC which paired the Parallax QuickStart board with an improved expansion board, adding 32K of SRAM and upgrading the VGA from 48 to 64 colors, with an option for 256. This update also added bus headers to allow direct access to available Propeller I/O pins. In 2014, the PMC was updated again to move to an easier-to-build single board configuration (no longer using the QuickStart board) and to add several features users were asking for. These included more access to I/O pins; onboard breadboard, LEDs, and switches; an interface option to the Raspberry Pi (or other computers); and sockets for port expansion and Flash memory ICs.

The resulting Propeller Experimenter Board (PEB 2014) from **propellerpowered.com** was a superb technology mentoring platform, but was no longer available in late 2014. The Amigo is a simplified version of that PEB 2014 developed by Dane Weston in mid-2015 to provide a BASIC-driven mentoring platform for *Nuts & Volts* readers.

FIGURE 7: Greetings from the Mentor's Friend!

<Esc> to interrupt Color BASIC, and type in SAVE "HELLO.BAS" to save your first program to the SD card. Next, type DIR to view a directory of the files on the SD card. You should see an entry for HELLO.BAS. You can now LOAD "HELLO.BAS" from the SD card, then RUN it to see it work, or LIST it to see the program code (**Figure 7**).

It really is that simple! If you have a youngster watching over your shoulder, press <F1> again to go into the editor and have him or her change WORLD in Line 100 to their name, then press <F1> again. Now, watch those young eyes light up as the Amigo screen fills up with personalized multicolor greetings!

Here's one more example of Amigo simplicity and fun to help get you started. Type NEW at the READY prompt to clear program memory and variables, then press <F1> for the editor and enter this code:

```
10 CLS
20 PRINT "AMIGO SIMPLE I/O": PRINT " "
30 PRINT "CONNECT S1 TO P8"
40 PRINT "CONNECT D1 TO P9"
50 PRINT "PRESS S1, AND D1 SHOULD BLINK"
60 REM ~~~ LOOP TO CHECK FOR BUTTON PUSH ~~~
70 IF INA[8]=1 THEN GOSUB 100 REM -
          '1'=PUSHED
80 GOTO 70
90 REM ~~~ SUBROUTINE TO BLINK LED ONCE ~~~
100 OUTA[9]=1 REM - '1'= LED ON
110 PAUSE 250 REM - WAIT 250MS
130 OUTA[9]=0 REM - '0'= LED OFF
140 PAUSE 250 REM - WAIT 250MS
150 RETURN
```

Press <F1> again to run the code, install two wire jumpers as instructed, then press switch S1. If all goes well, LED D1 should blink while you hold down the button, and stop when you release it. If not, check your jumpers. I know it's easy for me to misalign jumpers on the headers without a magnifier or my readers.

When you're ready to try something different, press <Esc> to return to the READY prompt, and don't forget to save your work with SAVE "BLINK.BAS." **Figure 8** shows my Amigo sending you greetings via a red LED.

This little program uses two jumper wires to demonstrate the Amigo version of the HELLO WORLD program of microcontrollers: the blinking LED. Setup is simple — just two wires — and I hope you think the program code is simple and intuitive.

No separate computer and integrated development environment (IDE), no compiling code and programming chips, and no dedicated microcontroller programming language to learn.

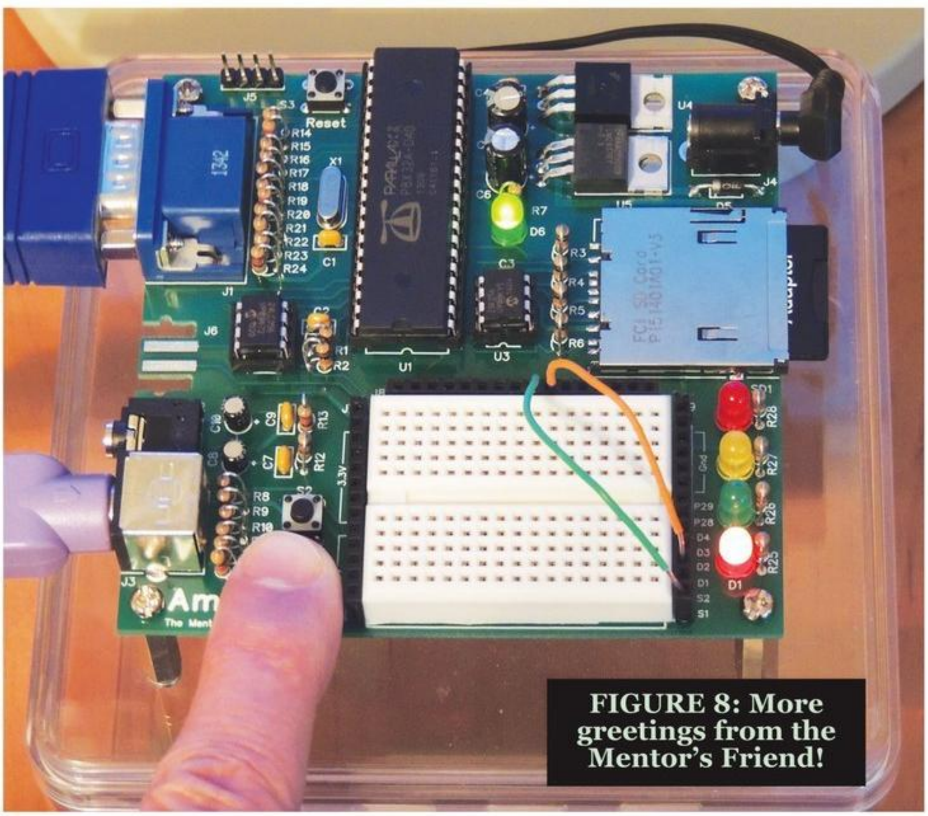


FIGURE 8: More greetings from the Mentor's Friend!

It's absolutely true that the Amigo lacks the richness of a full microcontroller lab (Color BASIC has no native commands for ADC or PWM, for instance), but it also lacks the accompanying complexity and learning curve. It is intended only to introduce youngsters to a few programming and hardware interface fundamentals in a one-on-one setting.

Comparisons to "more robust" learning platforms or languages sort of miss that point. In my opinion, the Amigo gives you plenty to work with to introduce technology in an easy and fun way. Who knows where that might lead?

Next Steps

What else can you do with the Mentor's Friend? In addition to being an excellent mentoring platform, the Amigo is a good choice if you don't need a full-featured programming language and just want to knock around on a simple and fun retro computer. This can include some enjoyable retro programming challenges. Try writing the classic player-versus-computer Battleship game in 4K with no arrays. You can do it, but it will make you work!

The Amigo is also handy to have around if you are a Spin programmer or want to be. (Spin is the purpose-built language from Parallax for programming the Propeller chip.) Editing the Spin source code for Color BASIC (to add your own custom BASIC commands, for instance) is a great way to explore the Propeller and Spin, and the Amigo is a good target platform for custom binaries that you might write.

If you choose to explore this, you will need a Parallax Prop Plug to interface the Amigo to your Spin development environment via USB.

So, that gives you some background on the Mentor's Friend, and some idea of what you can do with it. I think the Amigo is an excellent technology mentoring platform, easy to build, a good value for your dollar, and loads of fun to play with. Starting next month, *Nuts & Volts* will have a series of articles on different Amigo projects, from beginner to advanced. Perhaps you and a young friend will join us on that adventure! **NV**

Dane Weston's book on Color BASIC for the Pocket Mini Computer is available at **www.lulu.com/shop/display-product.ep?pID=4536358**.

ELECTRONET

■ BY L. PAUL VERHAGE

Showing Teachers One Way to Incorporate Near Space

The BalloonSat — or functioning model of satellites lofted by weather balloons into near space — is a fantastic STEM activity for students of all ages. However, if students are going to have this opportunity in the classroom, their teachers first need to know how to design, build, test, and fly BalloonSats. That's why I visited the University of Nebraska Omaha last summer.

Students can record images like this when teachers give them a BalloonSat experience. The data their BalloonSat will collect is astounding when you consider that near space looks and feels like outer space and IS far above the altitude of airplanes.

Nebraska science teachers hard at work measuring and configuring their avionics. Teachers selected their sensor and camera, which then influenced the design and goal of their BalloonSat. It's easier to dream up new ideas and correct mistakes in design while it's still on paper.

To help further the goal of incorporating more near space into STEM-related classes, I have traveled to many schools, summer camps, and teacher workshops where instructors learn and students experience. It's the reason I visited the University of Nebraska Omaha from July 7 -11, 2015. While there, I had a great time and got to help five Nebraska science teachers learn the ropes about near space.

The UNO Aerospace Educator Workshop is an annual summer science class for current and soon-to-be Nebraska teachers. At the three-week workshop, teachers spend one week exploring near space, a second week exploring aviation, and a final week doing projects that include developing an aerospace lesson plan. Participants completing the workshop have a stronger understanding of aerospace in terms of science and engineering, and how to incorporate aerospace into their classrooms. As you can imagine, I was there to help with the near space side.

My partner in crime was Mark Conner N9XTN, a meteorologist working with the US Air Force and a good friend going back to 1998. Mark insures that the near space launch, chase, and recovery will go according to plan. He does this by running flight predictions, predicting weather during the flight, locating launch sites, and rounding up launch and chase help. He also kicks off the High Altitude Balloon week with a presentation on near space.

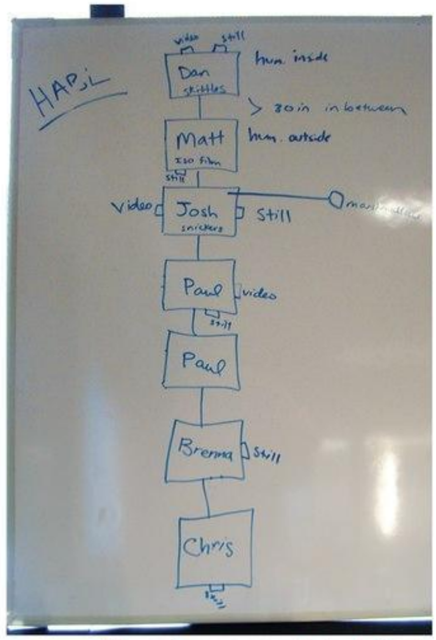
Wednesday

I arrived on campus Wednesday

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/december2015_NearSpace.**

and began my discussion with BalloonSats — the tool these teachers would use to explore near space. This included the BalloonSat materials, procedures, and limitations. In the realm of materials, I gave each teacher a set of avionics that included a flight computer (a BalloonSat Mini), digital camera, and a choice of sensor array (two temperature sensors or a temperature and relative humidity sensor). With their avionics in hand, teachers selected specific mission goals and began the paper design of their BalloonSat and the BalloonSat constellation we would assemble for this mission.

Measure twice, cut once. Metal straight edges and squares are important to laying out cutting lines in Styrofoam. The more carefully students make their lines, the less work they'll do to patch up their airframe.



The class completed their paper design and defined the near space mission by the end of the day. This whiteboard became a record of our plan for building the BalloonSats the next day.

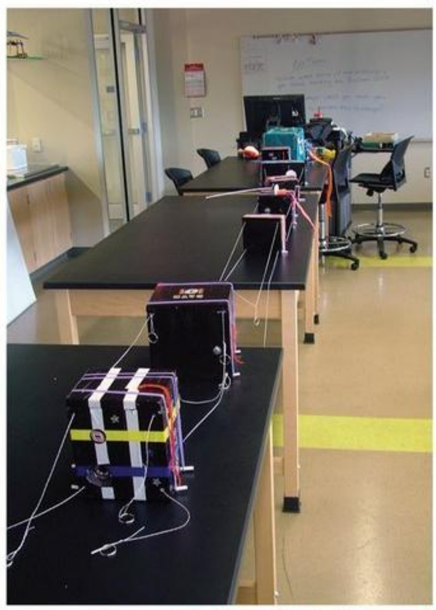
What constraints did these teachers design around? For one, there was a weight limitation: No BalloonSat could be greater than one pound in mass. Second, the BalloonSats needed to include four plastic tubes for tethering to each other and to the rest of the near spacecraft. Another one was the access hatch of each BalloonSat; it had to be placed on the side of the BalloonSat and not on its top or bottom. Finally, each BalloonSat had to secure its hatch using two rubber bands. This way, an errant battery couldn't slip out of the airframe by pushing on the hatch like it was a lever.

Thursday and Friday

Teachers began the construction of their BalloonSats on Thursday and finished them on Friday. Ms. Squires — the instructor of the class and soon-to-be PhD — provided the materials and construction tools like Styrofoam, metal straight edges, squares, Exacto™knives, and hot glue guns. The Styrofoam used to build the airframes is the type of sheeting found at home improvement stores (Menards, in this case) that's pink and 1/2" thick. The crumbly white Styrofoam will work, but is more difficult to build airframes with.

The teachers learned there are several reasons we use Styrofoam to build airframes, including its ability to

A completed BalloonSat and its initial paper design. The white cap is the bottom of a Styrofoam cup. It acts as a sun shield for the BalloonSat's temperature and relative humidity sensors.



Seven BalloonSats (two of them were my experiments) tethered together to the rest of the near spacecraft. Located at the far end are two tracking modules, a recovery parachute, and 18 feet of load line to attach the balloon to the parachute. All together, this near spacecraft weighed about 12 pounds.

insulate from the cold, its low weight, and its ease of machining. Students — even young ones — instinctively understand how to cut and glue it together. That ease allows students to concentrate on the BalloonSat design and its ability to collect data.

After gluing the airframe together, the teachers mounted the avionics inside the airframe. The flight computer was just glued to an interior shelf since it was lightweight. However, the camera was bolted to the airframe with 1/4-20 bolts. Likewise, teachers built a compartment into the airframe to contain the battery. Both the camera and battery are heavy and dense enough to pose a hazard, should they not be adequately restrained during descent. That's because initial descent is a very chaotic event and loose objects get thrown around a lot.

We ran short of time this year, so a comprehensive test of the flight articles was skipped. Otherwise, the teachers would have tested their BalloonSat construction by shaking it to insure nothing was free to bounce around, chilling it to make sure it still operated at low temperatures, and starting it to make sure crews would be able to start it up quickly and without errors at the launch site.

By the end of the day, the class had tethered their BalloonSats to the rest of the near spacecraft, checked out the latest prediction, and made plans to rendezvous at the launch site the next morning. It's pretty awesome to see three days' worth of work come together in a near spacecraft predicted to reach 90,000 feet and collect data in the near space environment.

At an altitude of 3,500 feet, one of the teacher's BalloonSats took this picture of the airport in Atlanta, IA. It's quite a nice airport and the managers have that midwest friendliness.

Five Nebraska teachers at the recovery of their BalloonSats. Next up will be downloading the data and images recorded in their BalloonSats.

Saturday

Everyone met at Treynor High School (Iowa) Saturday morning to begin the balloon filling process. It only takes about an hour to fill the balloon and during this time, teachers prepped their BalloonSats for flight. This included plugging in batteries and starting their cameras. It was an overcast morning and not very promising in regards to seeing the balloon during its flight (we'd still be able to track it over radio).

The flight prediction that morning showed only a minor change to the predicted recovery zone, so most of our driving plan remained unchanged. The largest obstacle facing us was that it had been raining recently and the back country roads and fields promised to be muddy (this is why I carry a pair of tall rain boots in my car when I go on a chase).

After an eventful launch, we made our first stop at Griswold where we could get food and drinks while waiting for the balloon to burst. Fortunately, the clouds opened up enough to let us see the balloon at around 60,000 feet. It was great for the first-time near space explorers to see the balloon as a tiny dot in the sky, and know it was their balloon and its current flight conditions like altitude, speed, and direction.

After the balloon burst at 92,704 feet, we updated our predicted recovery zone and headed out. Remember I said it had rained recently in Iowa? The only difficulty we experienced in this otherwise perfect mission was trying to get a car out of the mud. One of the county roads we drove onto was so bad we had to push the car out while it tried to drive in reverse. (I lost a pair of tennis shoes in the process since my mud boots were not accessible at the time.)

The landing site turned out to be at one end of the taxiway in Atlanta. We arrived at this surprisingly large airport and made a beeline for the operations office. We were able to get permission to enter the airfield in two cars and recover the near spacecraft. Needless to say, we stayed off the runway since our cars were unable to develop sufficient air speed.

I had a job teaching Quadcopters in Idaho on Monday, so I got right back on the road after having lunch with the teachers and chase crew in Council Bluffs. I want to thank Ms. Squires for sending me the data and images the BalloonSat recorded. It looks like the BalloonSats were in large part a total success.

Results?

It's through workshops like this that I hope to introduce teachers to BalloonSats, so they feel comfortable with introducing them into their curriculum. Not only do they need to see how to build a BalloonSat and get its data, they need to see how BalloonSats can fit into science standards like the Next Generation Science Standards, or NGSS.

NGSS is trying to take science education away from just teaching facts, history, and the interconnectedness of science to integrating science with engineering. BalloonSats fit into that view of science education because students need to design an engineered structure to collect science data in a lethal environment that can't otherwise be reached.

In some ways, a BalloonSat is a student's avatar, collecting the data they'd like to collect, but can't. Isn't that what exploring deep space is all about?

Onwards and Upwards,
Your near space guide **NV**

A BalloonSat Curriculum

Here are some of the topics a BalloonSat class will cover:

- What is near space?
- Conditions in near space
- How to reach near space
- Factors affecting near space flights
- Predicting near space flights
- Limitations and rules
- Avionics (depending on time available, this will include basic electronics and programming)
- Sensors
- Airframe design
- Airframe construction (tools and techniques)
- BalloonSat testing

Completing a BalloonSat curriculum can take over a week of full class time if it involves assembling and programming the flight computer. Some schools use an entire semester for their BalloonSat when they run it as an afterschool club. On the other hand, when the avionics is preassembled and programmed, teachers and students can get a BalloonSat ready for flight in two days. A class then needs an extra day to download and analyze their data.

Perhaps you're a teacher or know a teacher who might be interested in near space exploration with BalloonSats.

If so, please feel free to contact me at nearsys@gmail.com. I'd be happy to work out a way where I could help kick-start your near space program.

■ BY CHUCK HELLEBUYCK

Using 3D printers for practical projects on your workbench.

Fabrikator Mini – The 3D printer for Your Bench Top

I've shown you many tools that you can 3D print for your workshop if you have a 3D printer, but what if you don't have one? Well, I'm going to show you the tool you should get that will fit right next to your soldering iron: the Fabrikator Mini.

■ FIGURE 1. Fabrikator Mini.

The Fabrikator Mini is an open source miniature 3D printer that does a fantastic job of creating 3D printed objects. It's small enough to fit on just about any bench, but large enough to print many useful gadgets.

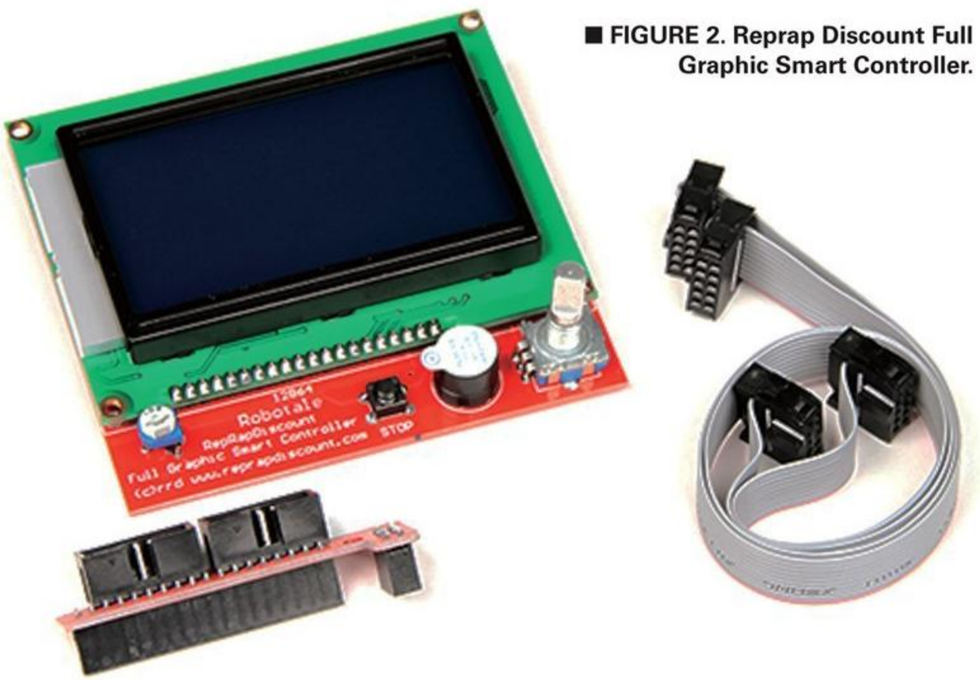
The Fabrikator Mini is small enough to fit in a carrying case and travel with you if you need 3D printing on the road. Parker Leung originally designed it as the Tiny Boy 3D printer, intended as an easy to assemble kit for students to get introduced to 3D printing. Because the design is open source even for commercial sales, the Fabrikator Mini became an easy to get, fully assembled version of the Tiny Boy we can all get our hands on.

The Tiny Boy has a rather small build area when compared to current 3D printers, but is on par with the original Makerbot Cupcake and Thing-o-matic 3D printers that helped launch the home 3D printer revolution. At 80 mm x 80 mm x 80 mm for the build area, you might wonder how useful it could be. Many prints will fit in that area, and lots of designs can be easily broken into sections that can be printed on the Fabrikator Mini and then glued/attached together.

The price is very attractive at under $220. That's a bargain for a fully assembled 3D printer ready to use out of the box. As mentioned, it relies on open source software such as Repetier and can print at a 0.1 mm to 0.4 mm layer height, though 0.2 mm is the most reliable fine print resolution.

To get the best prints out of it, it's recommended in the setup manual to run the print head at 11 mm/second. This is rather slow, but produces outstanding print quality. It's also designed to print with PLA plastic, so a heated bed is not included. That didn't stop me, however.

I added some Buildtak material to the build area — which helps with making prints stick — and printed with ABS. I had great results using a raft as the first layer. A raft is just a layer of plastic put down first to make everything stick better to the unheated bed. This is the way the



■ FIGURE 2. Reprap Discount Full Graphic Smart Controller.

Post comments on this article and find any associated files and/or downloads at
www.nutsvolts.com/index.php?/magazine/article/december2015_Practical3DPrinting.

original Cupcake printer worked when ABS was the only choice back then.

What I have found is so many of the prints I want to make for my bench — including custom tool holders or knobs for potentiometers — can all be printed on the Fabrikator Mini without having to break up the design into sections. The speed also isn't an issue because it just churns away quietly while I solder together a board or two or write some code.

After a while, the noises coming from the hot end moving back and forth to extrude the melted plastic forms a sort of calming "music" that is easy to listen to. Plus, since most prints are small, very little plastic is used making it seem less expensive to operate.

LCD and SD Card Control

One disadvantage to the Fabrikator Mini design is the need to control it from the USB port of your computer. The Repetier software sends the G-codes to the printer through that USB connection. I wanted a way to let it run independent of the computer, and I found that in an open source LCD module with SD card support. The board is called the Reprap Discount Full Graphic Smart Controller.

It can be purchased from a lot of different places on the Internet. What makes it special is the ability to completely control the Fabrikator Mini from a multi-turn potentiometer with a switch underneath. A few turns and a click, and you have a print loaded and running from the inserted SD card. All you have to do is use your computer to create the G-code file from the Repetier software, save it to the SD card, and you have "computer free" 3D printing on your bench.

The Smart Controller requires a firmware upgrade to the motherboard of the Fabrikator Mini. The open source nature of the machine makes this as easy as programming any electronics module. I can step a user through it on my YouTube Channel. There are many other places to get the details, as well.

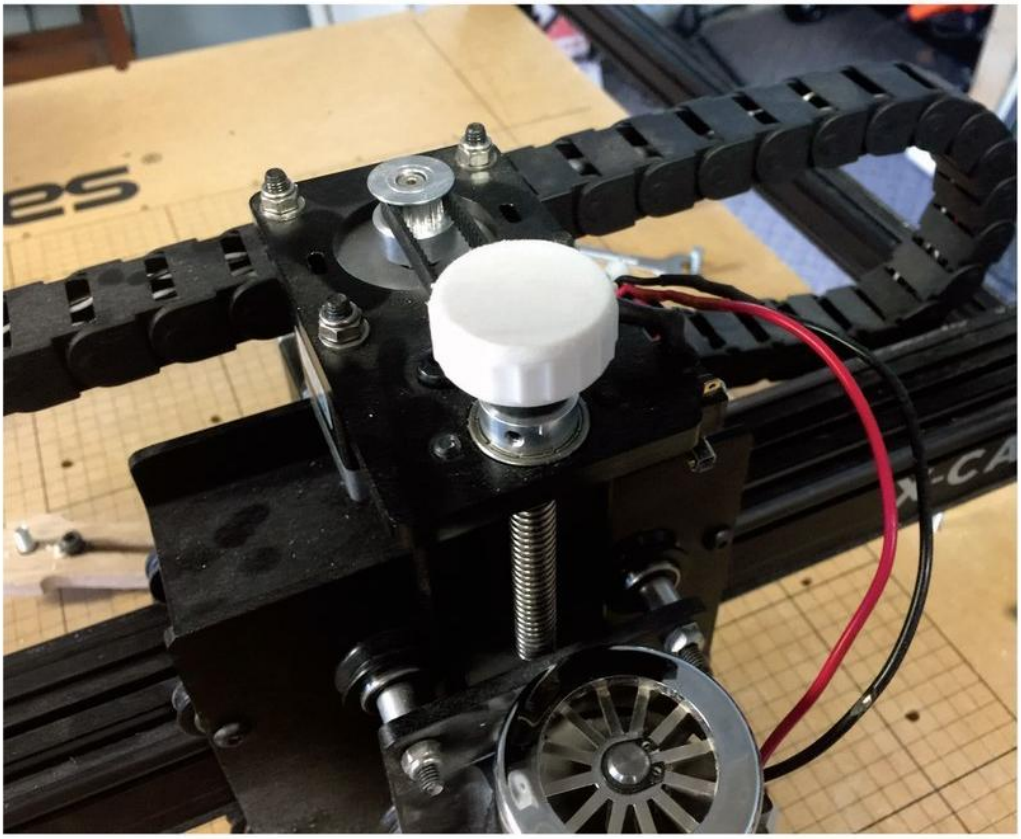
To make the printer more attractive and easier to use with the Smart Controller, I designed a base for the Fabrikator to sit on and a stand for the LCD that locks into the base, then downloaded a case design from Thingiverse that was created by user, Wersy to house the LCD module. I also created a spool holder that conveniently holds a spool I took from an empty da Vinci 3D printer cartridge. This gave me a complete stand-alone 3D printer that I can keep on my bench or have it travel with me.

I printed the LCD case, stand, and base on my larger 3D printer, but with some help from Fabrikator Forum member, Roger_1994, I now have the designs broken into sections that are small enough to print on the Fabrikator Mini; they lock together with puzzle piece ends. I put the files on my webpage dedicated to the Fabrikator Mini at **www.elproducts.com**. This way, I can give back to others the way I've been helped.

3D Prints

I have an X-Carve CNC machine that I'm using to learn how to carve out wood cases and circuit boards. One of the additions I added was a Z axis knob that allowed me to manually adjust the home position of the cutting bit. I found a design from YouTube creator I Like To Make Stuff and printed it in flexible plastic on the Fabrikator Mini. This again isn't the recommended type of plastic for this

■ **FIGURE 3. X-Carve CNC Z axis knob.**



machine, but it handled it fine. The knob presses over a nut and gives me a knob with a good grip to change the Z axis position.

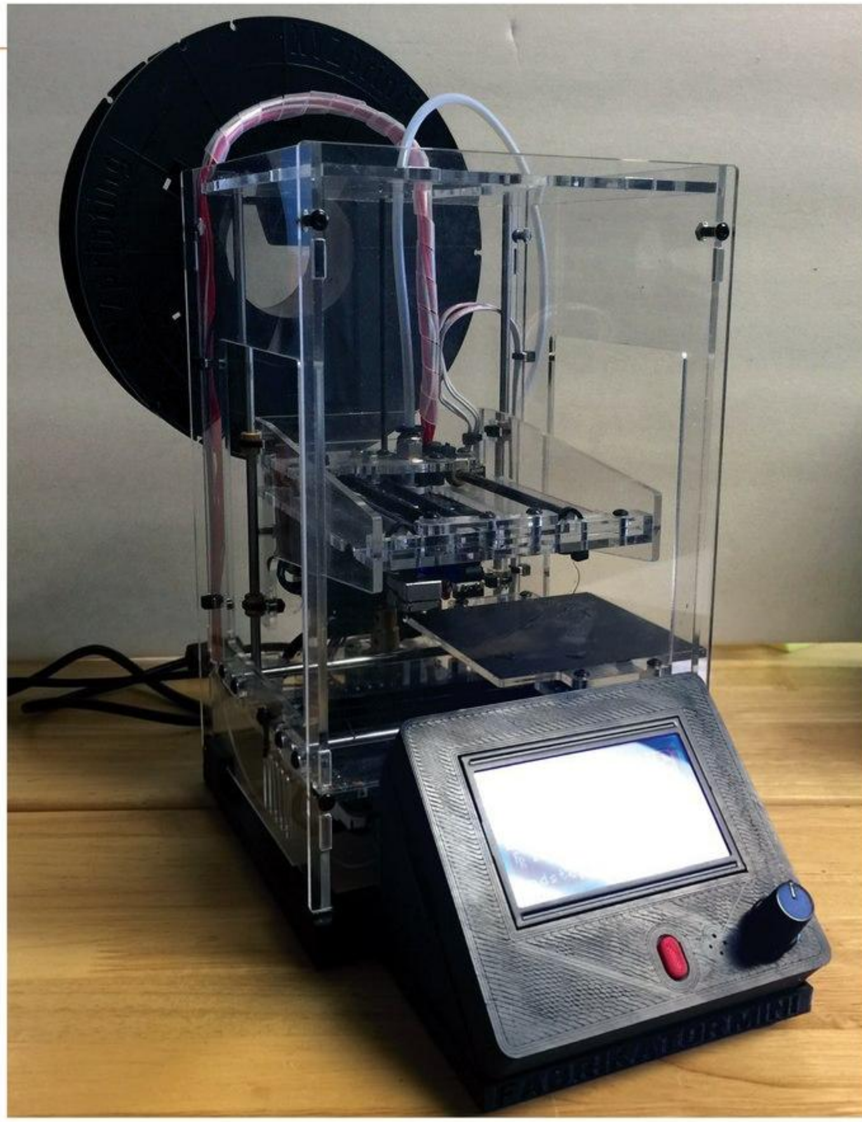
Replacement Parts

One of the more frustrating parts of any 3D printer is when something breaks. If it's a custom design, then you either have to get a replacement part from the manufacturer or hope someone has designed a replacement that you can print. In the case of a hot end or motherboard or stepper motor, these aren't parts you can just 3D print. Fortunately, the Fabrikator Mini is built from parts you can easily get online.

The extruder is an E3D V6 style, though I believe it is a clone of the original design. The motherboard is an MCSBASE. The steppers — which rarely fail — can also be found online. So, this is an easy to use, easy to fix, low cost 3D printer designed for kids but useful for everybody.

So, if you were still in the thinking stage of getting a 3D printer for your electronics lab, then I say don't wait any longer. Get a Fabrikator Mini and get started printing all those tool holders and knobs and custom cases you have in mind. If you need to design them, check out the free online Tinkercad software that I often use in my YouTube videos.

Yes, 3D printing has gotten easier over the past few years. Now it's gotten better, cheaper, and smaller. **NV**



■ **FIGURE 5. My completely modified Fabrikator Mini.**

■ **FIGURE 4. E3D V6 hot end.**

Resources

Check out my website and blog:
www.elproducts.com

My YouTube channel:
www.youtube.com/elproducts

Tinkercad design:
LCD case: **www.thingiverse.com/thing:87250**
X-Carve knob: **www.thingiverse.com/thing:1020243**

Fabrikator Mini Website with base and stand:
www.elproducts.com/fabrikator-mini.html

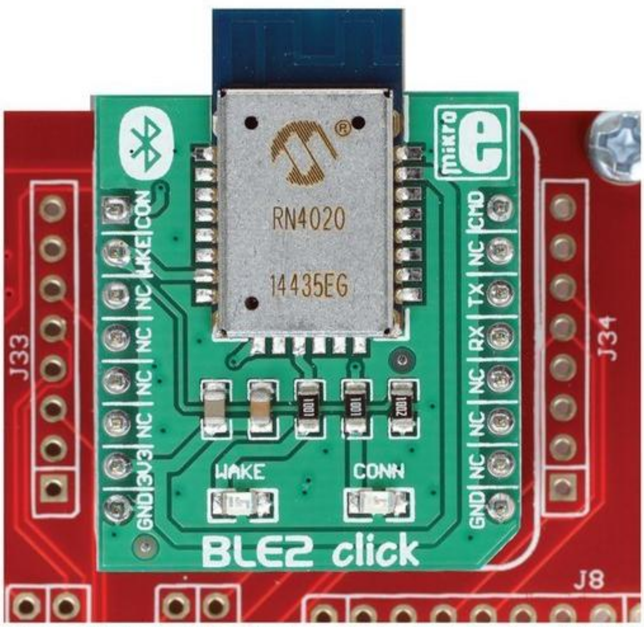
My 3D designs:
www.thingiverse.com/elproducts/designs

Tinkercad:
www.tinkercad.com

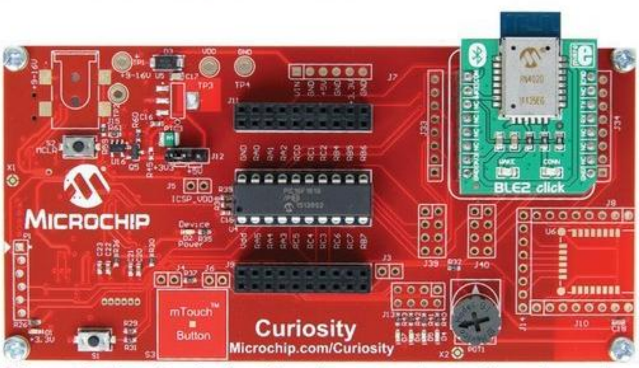
■ BY FRED EADY

Automagic with the MPLAB Code Configurator

Do you recall the scene from *Star Trek IV: The Voyage Home* in which Scotty is formulating transparent aluminum to house a couple of whales on the "borrowed" Klingon Battle Cruiser? After requesting access to a computer, Scotty is directed to a PC equipped with a mouse and keyboard. Perplexed, Scotty tries to "talk" to the PC. When the computer doesn't respond, Bones hands Scotty the mouse. Scotty then attempts to use the mouse as a microphone. The PC still doesn't "answer." The owner of the aluminum products business tells Scotty to just use the keyboard. So, Scotty huffs, "Keyboard? How quaint."



■ Photo 1. This is MikroElektronika's version of an RN4020 module carrier complete with LEDs.



■ Photo 2. The Curiosity development board's onboard PIC programmer is populated on the opposite side of the printed circuit board. I've plugged in the RN4020 *click* board that you see mounted to the upper right.

If Microchip keeps it up, pretty soon we'll all be "talking" to PIC microcontrollers. This month's subject matter is centered on Microchip's new Curiosity development board. As we put the Curiosity board through its paces, you will quickly discern that the real magic lies in the automated software development that has been mixed into MPLAB X.

Curiosity Automated the Cat

The Curiosity development board is not designed around a single PIC. If your application calls for an eight-pin PIC, you can choose from the 12F752, the 12F151, or the 12F1572. Need something bigger? Take your choice from the 16F1455, the 16F1507, the 16F1619, or the 16F1708. I've only listed a few of the PIC possibilities. There are currently 46 supported 12F and 16F PIC types, including a couple of 18F devices.

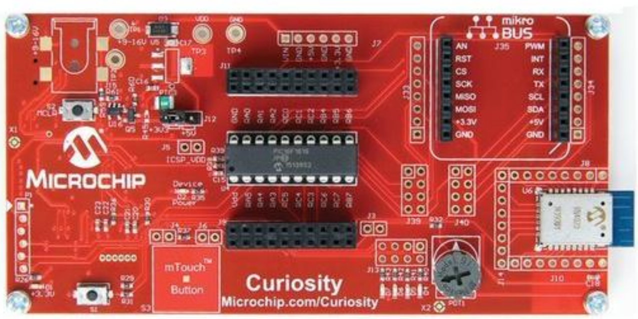
Once you choose the PIC variant you want to work with, you won't have to go very far to find the tools you will need to program and debug it. Forget having to chase down a PIC programmer. The Curiosity development board has its own. An onboard PIC24FJ256GB106 is programmed to act as the Curiosity board's resident programmer/debugger. Power is not a problem, either. The Curiosity dev board can be powered from its USB portal, an external nine volt DC source, or an external variable DC source. If you want to use the nine volt DC source, be prepared to do a little bit of component mounting and soldering.

Like all well-designed development boards, the Curiosity board sports a user-accessible pot, LEDs, and a pushbutton switch. There's even an mTouch Button pad laid into the board's fiberglass. In the spirit of IoT (Internet of Things), the Curiosity board also includes a set of pads designed to support a Microchip RN4020 Bluetooth Smart radio module. If you're not up to soldering down a raw RN4020, you can simply "click" in a MikroElektronika RN4020-equipped *click* board. You're not limited to just the RN4020 version depicted in **Photo 1**. You can plug in any of the multitude of MikroElektronika *click* boards.

The Curiosity dev board equipped with an RN4020 *click* you see in **Photo 2** has a number of unpopulated dual-row jumper block header pins. You have the option of populating them if you wish to use physical shorting jumper blocks. However, there happens to be a zero ohm SMT jumper soldered across the jumper block header pads on the opposite side of the Curiosity board. Instead of populating physical jumper blocks, you can simply keep or remove the zero ohm SMD jumpers. **Photo 3** is a solder-side shot of the onboard programmer/debugger. **Photo 3** also reveals a couple of the zero ohm SMD parts represented as R34 and R46.

The Curiosity board requires the target PIC's LVP bit to be set if you wish to use the board's onboard PIC programmer. For those times that you'll need to set a previously programmed PIC's LVP (Low Voltage Program) bit, the Curiosity is equipped with a standard PICkit3 ICSP interface. If the target PIC is new and unprogrammed, you can set the LVP bit in the configuration bits. As you'll see later, setting up a new PIC is a walk in the park.

■ Photo 3. This is essentially an LVP version of the PICkit3. MPLABX 3.10 recognizes the Curiosity board programmer/debugger circuit as a PKOB. Note the SMD jumpers at R34 and R46.



■ Photo 4. I've opted to solder in a raw RN4020 Bluetooth Smart radio module instead of plugging in an RN4020 *click*. The RN4020 is a breeze to mount and solder manually.

Now the Cat Can Code in C

The Curiosity board hardware is just sitting there waiting for some code to execute. Let's add one more piece of hardware before we unleash the C coding cat that is caged within MPLAB X. Behold **Photo 4** and the newly added raw RN4020 Bluetooth Smart radio module. There's a method to this madness. We're going to use the MPLAB Code Configurator to automate the process of building the base code for our new semi-permanent RN4020 peripheral.

The goal is to bring the RN4020 to life. I have a couple of Curiosity dev boards, so, we'll hang the RN4020 *click* on the second board. If all works as designed, we only have to write the code once as both Curiosity boards will drive their respective RN4020 Smart radio modules in a similar manner. The only difference will be configuring one board as a Central (Master) and the other board as a Peripheral (Slave).

Preflight Checks

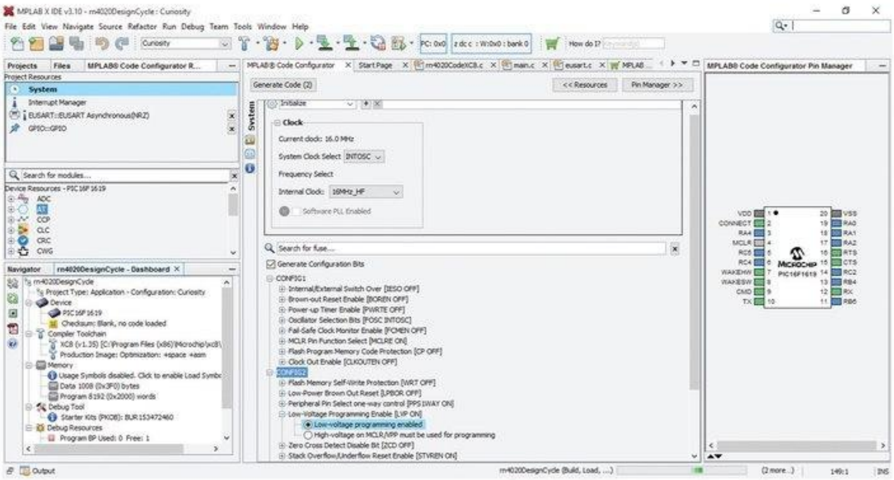
The Curiosity comes loaded with a PIC16F1619, which features a simple demo app. All we have to do is attach the Curiosity to a PC via USB. Once that's done, we should see the Device Power and +3.3 VDC LEDs illuminate. If the PIC is running correctly, pressing pushbutton S1 will illuminate LED D4. Rotating the pot should change the intensity of LED D7. If all that checks out, we can start our taxi on the Bluetooth Smart runway.

Let's stick with the PIC16F1619 and create a new MPLAB X project called *rn4020DesignCycle*. Instead of our trusty PICkit3, we'll use the Curiosity board's onboard programmer/debugger. We will use version 3.01 of MPLAB X as it will recognize the Curiosity's onboard programmer/debugger hardware.

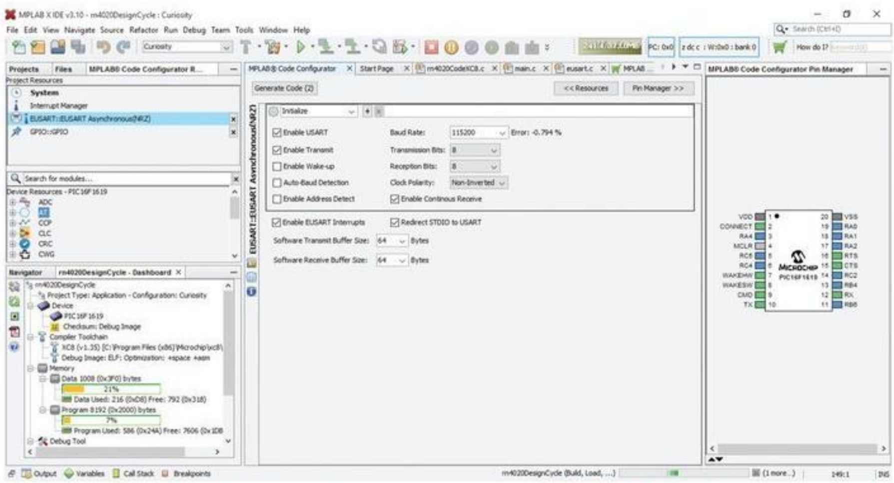
We must also configure MPLAB X to utilize the Microchip XC8 C compiler and the MPLAB Code Configurator. The MPLAB Code Configurator produces C code that we can weave into our RN4020 firmware application. There is no substitute for reading and understanding datasheets. However, the Configurator saves us a bunch of time as we don't have to worry about which bits in which register must be set or cleared to activate a particular PIC subsystem.

For instance, by utilizing the Code Configurator, UART baud rates and timer intervals are automatically calculated. MPLAB Code Configurator even takes care of peripherals

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/december2015_DesignCycle**.



■ Screenshot 1. This is a capture of the MPLAB Code Configurator System configuration screen. All of the PIC16F1619 configuration fuses can be configured from this panel (configuratively speaking).



■ Screenshot 2. All of the pain of setting up an EUSART has been removed by this window. We need only tell the MPLAB Code Configurator what we want the EUSART to do and how fast it should run. To utilize the XC8 C compiler's *printf* function, we must check the Redirect STDIO to USART check box.

that require the use of PPS (Peripheral Pin Select).

We can generate code with the Code Configurator until the cows come home. If the code we're producing doesn't match our RN4020 hardware, it is totally useless. So, let's draw up our plan schematically and set up our hardware using the Configurator.

System Setup

Schematic 1 tells us that pins RB5 and RB7 are receive and transmit pins, respectively. Let's use that information to initialize the PIC16F1619's EUSART. However, before we set up the EUSART, we need to take care of the system setup first. We will run at 16 MHz using the PIC's internal oscillator. The PIC's configuration fuses are set using the Code Configurator. The fuse panel is shown in **Screenshot 1**. What you don't see in **Screenshot 1** is my personal selection of some configuration fuse particulars. The only fuse that is critical to be configured is the LVP Enable, which should be turned ON.

EUSART Setup

Now that we have established a CPU clock frequency, we can move on and configure the PIC's EUSART. The RN4020 operates at a baud rate of 115200 bps. So, as you can see in **Screenshot 2**, we simply relate our baud rate desires to the MPLAB Code Configurator. While we're at it, we'll lay the ground work for the EUSART interrupt mechanism. In **Screenshot 2**, I've also enabled EUSART interrupts and set up a pair of 64-byte receive and transmit buffers. Being able to write to the EUSART using *printf* statements is a good thing. So, to enable the *printf* functionality, I've checked the Redirect STDIO to UART check box.

GPIO Setup

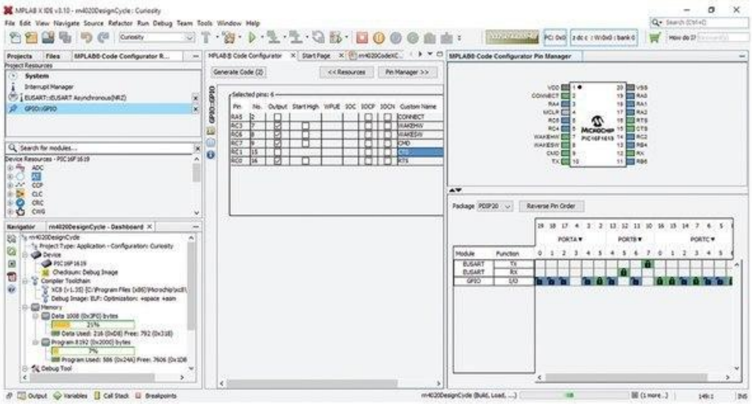
In my opinion, this is the coolest feature of the Configurator. When creating new project code, I spend LOTS of time manually defining I/O pins and creating their aliases. As you can see in **Screenshot 3**, the Code Configurator has assisted in identifying the EUSART pins, as well as all of the RN4020 interface I/O pins.

The Curiosity dev board has a trace running from the PIC's RC0 pin to the RN4020's WAKEHW pin. If you don't want to cut that trace on your Curiosity board, you can assign the WAKEHW pin to RC0 and forego using the optional RTS pin. I cut the WAKEHW trace on the board and wired in the RTS signal.

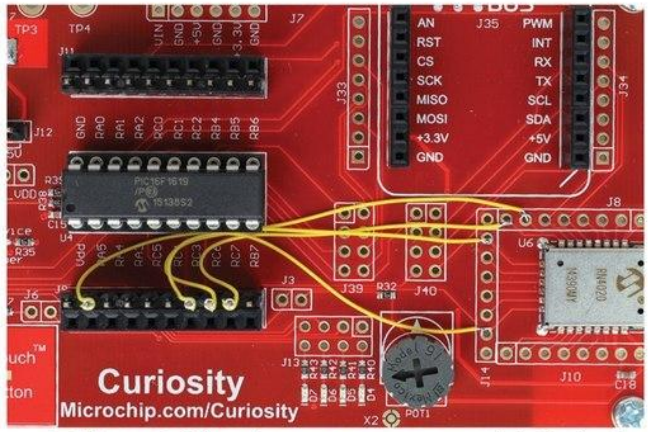
Wiring In the RN4020

The RN4020's TX and RX pins are already connected correctly by existing traces on the Curiosity. The TX and RX lines are shared with the *click* mikro BUS socket. So, you can't load the socket and use the onboard RN4020 at the same time. If you want to use the onboard RN4020 and a *click* module simultaneously, you'll have to isolate the onboard RN4020 by cutting the traces that are common to the *click* mikro BUS socket.

I like wire wrap. If you use it carefully, you can save



■ Screenshot 3. This panel enables you to completely set up your GPIO configuration. Everything we need to configure and identify the PIC's I/O pins is here.



■ Photo 5. When I want to repurpose this Curiosity development board, all I have to do is pull the male headers and untack the connections on the RN4020 pads. Note the absence of the RTS and CTS connections.

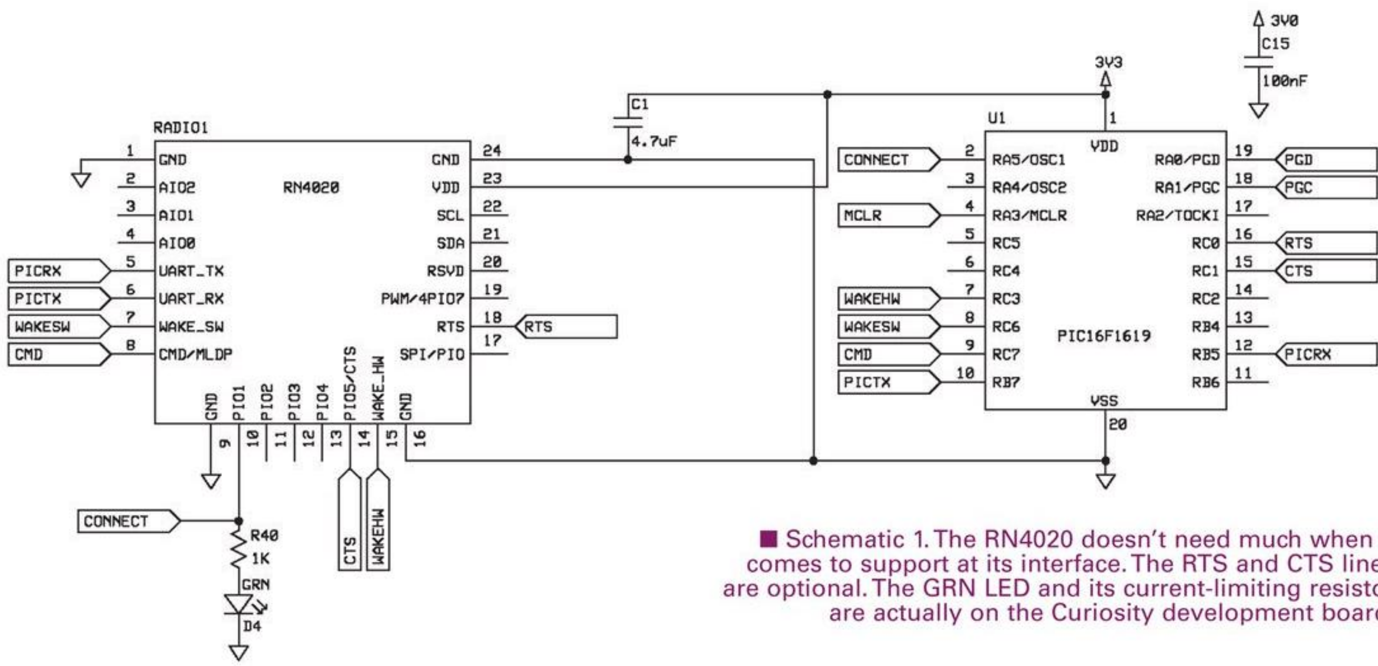
yourself, your circuit boards, and your electronics from destructive soldering and unsoldering operations. Wire wrap techniques also allow very easy "redos" and changes. Take a look at **Photo 5**. I've inserted a pair of standard 0.1" headers into the PIC's female GPIO headers. The exposed male header posts are used as wire wrap points. The other ends of the wire wrap wires are tacked to the RN4020 header pads per **Schematic 1**.

Does It Work

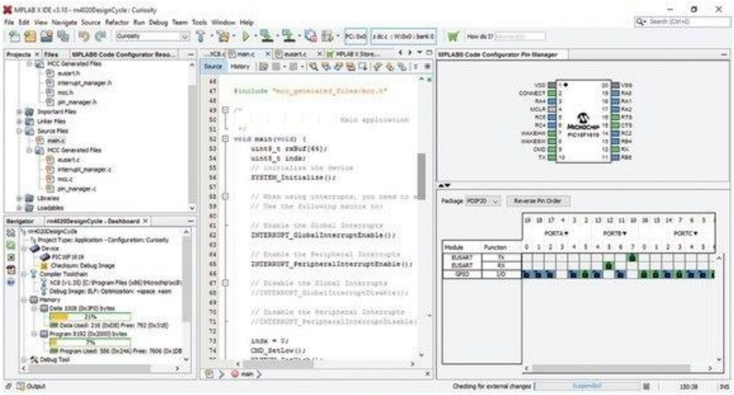
Don't know yet. Let's click the Code Configurator's Generate Code button and see what transpires. We should at a minimum get some configuration fuse statements, some EUSART setup, macro, and interrupt code, and some GPIO macros.

Screenshot 4 is promising. Let's delve into the Configurator-generated *mcc.c* file:

```
// CONFIG1
#pragma config IESO = OFF
   // Internal/External Switch Over->Internal
   // External Switch Over mode is disabled
#pragma config BOREN = OFF
   // Brown-out Reset Enable->Brown-out Reset
   // disabled
#pragma config PWRTE = OFF
   // Power-up Timer Enable->PWRT disabled
#pragma config FOSC = INTOSC
   // Oscillator Selection Bits->INTOSC
   // oscillator: I/O function on CLKIN pin
#pragma config FCMEN = OFF
   // Fail-Safe Clock Monitor Enable->Fail-Safe
   // Clock Monitor is disabled
#pragma config MCLRE = ON
```



■ Schematic 1. The RN4020 doesn't need much when it comes to support at its interface. The RTS and CTS lines are optional. The GRN LED and its current-limiting resistor are actually on the Curiosity development board.



■ Screenshot 4. Looking good. There are a number of MPLAB Code Configurator-generated files that were automatically placed into our project.

```
    // MCLR Pin Function Select->MCLR/VPP pin
    // function is MCLR
#pragma config CP = OFF
    // Flash Program Memory Code Protection-
    // >Program memory code protection is
    // disabled
#pragma config CLKOUTEN = OFF
    // Clock Out Enable->CLKOUT function is
    // disabled. I/O or oscillator function on
    // the CLKOUT pin
```

I've only partially listed the entire set of CONFIG fuse entries here. There are also CONFIG2 and CONFIG3 fuse statements that reflect the remaining configuration bit entries that were made in the **Screenshot 1** fuse panel. I'll bet if I look hard enough, I'll find some clock settings information. For instance, the *mcc.h* file also contains:

```
#define _XTAL_FREQ  16000000
```

The EUSART is fully covered. The Code Configurator generated EUSART setup code based on the entries that were made in **Screenshot 2**. Here are the Configurator-generated EUSART RX/TX buffer definitions:

```
static uint8_t eusartTxHead = 0;
static uint8_t eusartTxTail = 0;
static uint8_t eusartTxBuffer[EUSART_TX_
 BUFFER_SIZE];
volatile uint8_t eusartTxBufferRemaining;

static uint8_t eusartRxHead = 0;
static uint8_t eusartRxTail = 0;
static uint8_t eusartRxBuffer[EUSART_RX_
 BUFFER_SIZE];
volatile uint8_t eusartRxCount;
```

The MPLAB Code Configurator also generated a complete EUSART interrupt handler and didn't skimp on the GPIO code, either. The Configurator automatically generated the EUSART TX and RX pin PPS pin assignment code:

```
bool state = GIE;
GIE = 0;
PPSLOCK = 0x55;
PPSLOCK = 0xAA;
PPSLOCKbits.PPSLOCKED = 0x00; // unlock PPS

RXPPS = 0x0D; // RB5->EUSART:RX
RB7PPS = 0x12; // RB7->EUSART:TX
PPSLOCK = 0x55;
PPSLOCK = 0xAA;
PPSLOCKbits.PPSLOCKED = 0x01; // lock PPS
GIE = state;
```

There's nothing you can't do to the PIC's WAKESW I/O pin thanks to the Configurator:

```
 // get/set WAKESW aliases
 #define WAKESW_TRIS              TRISC6
 #define WAKESW_LAT               LATC6
 #define WAKESW_PORT              RC6
 #define WAKESW_ANS               ANSC6
 #define WAKESW_SetHigh()      do { LATC6 = 1; }
                               while(0)
#define WAKESW_SetLow()        do { LATC6 = 0; }
                               while(0)
#define WAKESW_Toggle()        do { LATC6 =
                               ~LATC6; } while(0)
#define WAKESW_GetValue()      RC6
#define WAKESW_SetDigitalInput()
        do { TRISC6 = 1; } while(0)
#define WAKESW_SetDigitalOutput()
        do { TRISC6 = 0; } while(0)

#define WAKESW_SetAnalogMode()
        do { ANSC6 = 1; } while(0)
#define WAKESW_SetDigitalMode()
        do { ANSC6 = 0; } while(0)
```

The WAKESW pin manipulation code is duplicated across all of the GPIO pins that we defined in **Screenshot 3**. I think you get it. The MPLAB Code Configurator is awesome. However, to get the RN4020 online, we'll have to write a bit of code. First, let's bring the RN4020 to life:

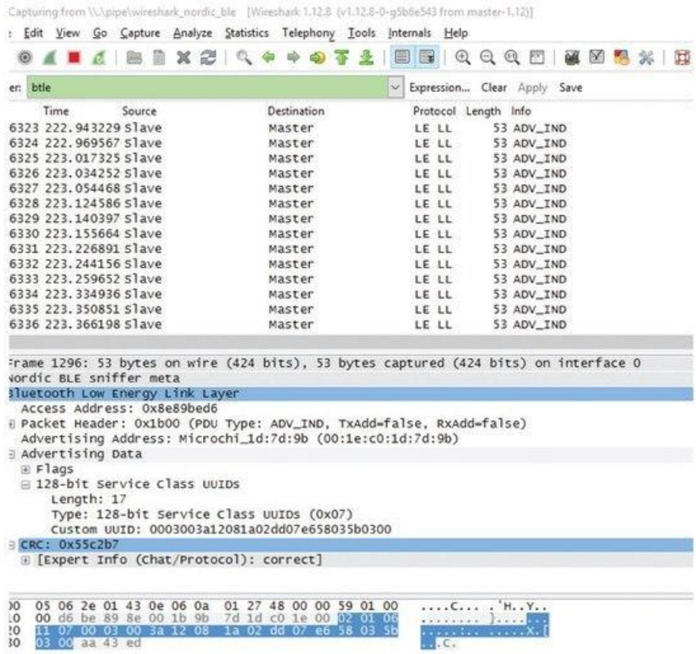
```
indx = 0;
CMD_SetLow();
WAKEHW_SetHigh();
WAKESW_SetLow();
__delay_ms(100);
WAKESW_SetHigh();
__delay_ms(2000);
```

If things are good following the RN4020 CPR, we should receive a CMD message from the RN4020:

```
indx = 0;
if(EUSART_DataReady)
{
     do{
          rxBuf[indx++] = EUSART_Read();
     }while(EUSART_DataReady);
}
```

When the CMD message arrives, we will configure the Curiosity dev board as a peripheral and enable MLDP:

```
if(rxBuf[0] == 'C' && \
   rxBuf[1] == 'M' && \
   rxBuf[2] == 'D')
{
```



■ Screenshot 5. This is a Wireshark capture of our Curiosity development board's RN4020 advertisement.

■ Photo 6. The Sniffer is based on the Nordic Semiconductor nRF51822.

```
        indx = 0;
        printf("SR,32000000\r\n");
          //set as Peripheral and enable MLDP
        __delay_ms(500);
        if(EUSART_DataReady)
        {
             do{
                  rxBuf[indx++] = EUSART_Read();
             }while(EUSART_DataReady);
        }
   }
```

The RN4020 will respond with AOK if the peripheral/MLDP command was successfully issued. Upon receipt of the AOK message, we will perform an RN4020 reboot:

```
   if(rxBuf[0] == 'A' && \
      rxBuf[1] == 'O' && \
      rxBuf[2] == 'K')
   {
        indx = 0;
        printf("R,1\r\n"); //reboot command
        __delay_ms(1000);

        if(EUSART_DataReady)
        {
             do{
                  rxBuf[indx++] = EUSART_Read();
             }while(EUSART_DataReady);
        }
   }
   if(rxBuf[0] == 'R' && \
      rxBuf[1] == 'e' && \
      rxBuf[2] == 'b' && \
      rxBuf[3] == 'o' && \
      rxBuf[4] == 'o' && \
      rxBuf[5] == 't')
   {
        indx = 0;
        __delay_ms(1000);
        if(EUSART_DataReady)
        {

             do{
                  rxBuf[indx++] = EUSART_Read();
             }while(EUSART_DataReady);
        }

   }
    if(rxBuf[0] == 'C' && \
       rxBuf[1] == 'M' && \
       rxBuf[2] == 'D')
    {
         // RN4020  is ONLINE at this point
    }
```

The comment "// RN4020 is ONLINE at this point" says it all.

But, Does It Work?

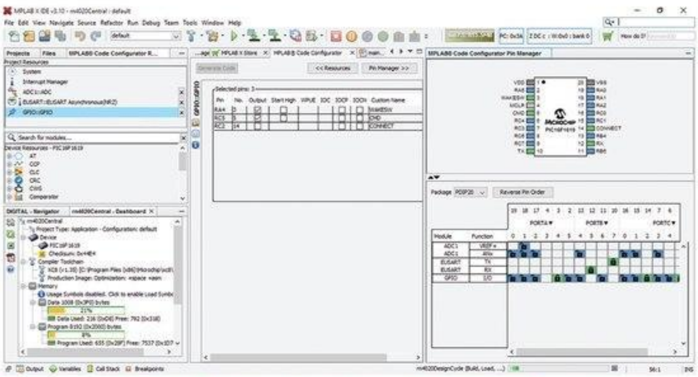
I happen to have a Bluetooth Smart Sniffer on the EDTP bench. The Bluetooth Smart Sniffer hardware is under the lights in **Photo 6**. The Sniffer passes packets to Wireshark. If our RN4020 driver worked, the RN4020 should be "advertising" and Wireshark should be picking that up. It works. Take a look at **Screenshot 5**. The proof of concept is revealed in the Wireshark capture's Advertising Address field. The actual data is highlighted in the hex dump at the bottom of **Screenshot 5**.

Generating a Central

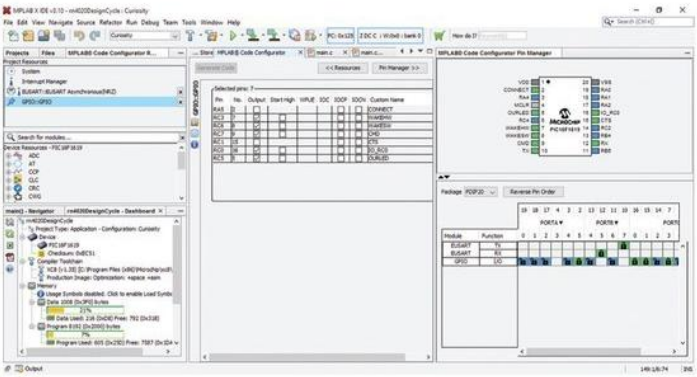
Now that we have a Peripheral (Slave), we can use most of the peripheral code to generate a Central (Master). The first order of business is to create a new project. I called it *rn4020Central*. I used the Code Configurator to generate the code for our new Central project. The newly generated code is identical to our original *rn4020DesignCycle* peripheral project with the exception of the GPIO setup. The new project GPIO setup is shown in **Screenshot 6**.

We will need to configure the RN4020 *click* module as a Central. To do this, we simply change the role value in our original peripheral code:

```
   if(rxBuf[0] == 'C' && \
      rxBuf[1] == 'M' && \
      rxBuf[2] == 'D')
   {
        indx = 0;
        printf("SR,92000000\r\n");
          //set as Central and enable MLDP
```



■ Screenshot 6. This is a familiar shot. Note the reduced number of GPIO pins required to support the RN4020 *click*. These are the only pins that the *click* brings out to its headers.



■ Screenshot 8. I simply added the RC5 pin as OURLED and regenerated the GPIO code. Macros for the new GPIO pin were seamlessly added to the original code.

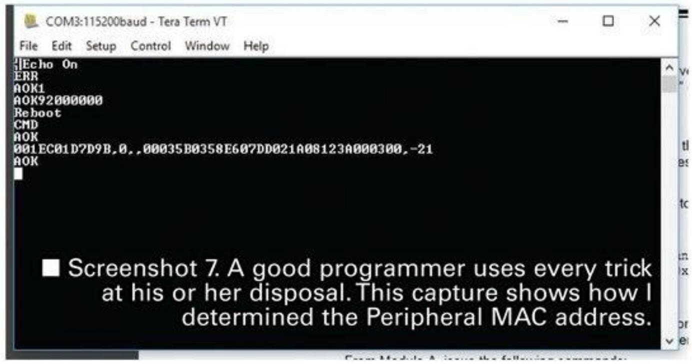
```
    __delay_ms(500);
    if(EUSART_DataReady)
    {
        do{
            rxBuf[indx++] = EUSART_Read();
        }while(EUSART_DataReady);
    }
}
```

Once the Central comes up, it will need to connect to our original peripheral. We're going to cheat a little and use an RN4020 PICtail to obtain the peripheral's public MAC address. The process is captured in **Screenshot 7**.

We could have written scan code to get this address, but since we're only concerned with a peer-to-peer connection, I decided to go the cheap and easy way. Here's the Master connection code that was added to the original peripheral code:

```
if(rxBuf[0] == 'C' && \
   rxBuf[1] == 'M' && \
   rxBuf[2] == 'D')
{
    printf("E,0,001EC01D7D9B\r\n");
     //connect to Peripheral
    __delay_ms(100);
    CMD_SetHigh();     //enter MLDP mode
}
```

Once connected, we can exchange data using the



■ Screenshot 7. A good programmer uses every trick at his or her disposal. This capture shows how I determined the Peripheral MAC address.

MLDP mode, which allows us to communicate with ASCII characters. Let's send a tilde character (~) once per second:

```
    do{
        __delay_ms(1000);
        printf("~");

  }while(1);
```

We didn't code the Peripheral to process the tilde. So, we can go back to the peripheral project and add the code. We can also use one of the Curiosity board's LEDs to indicate that the tilde character is getting processed. A quick look at the Curiosity schematic (in the Curiosity User's Guide) tells us that we can access LED D7 via RC5 of the PIC16F1619. I added the RC5 pin using the Code Configurator in **Screenshot 8**. Here's the tilde handler code:

```
    indx = 0;
    do{
     if(EUSART_DataReady)
     {
         do{
             rxBuf[indx] = EUSART_Read();
             if(rxBuf[indx] == '~')
             {
                 OURLED_Toggle();
//blink  LED D7 on Curiosity Development Board
             }
         }while(EUSART_DataReady);
     }
  }while(1);
```

By the way, when the Peripheral establishes a connection, LED D4 will illuminate on the Peripheral Curiosity development board. The Master indicates a connection by the CONN LED on the *click* module.

Still Curious?

You've got to try this on your own. I'll supply the project files we used via the article link. In the meantime, you can now add Curiosity to your design cycle. **NV**

Curiosity Development Board
XC8 C Compiler
RN4020 Bluetooth Smart Radio Module
RN4020 *click* Module
Microchip
www.microchip.com

www.mikroe.com/click

BLE Sniffer
Adafruit
www.adafruit.com

CLASSIFIEDS

THE OWNERSHIP, MANAGEMENT, AND CIRCULATION STATEMENT OF NUTS & VOLTS MAGAZINE, Publication Number: 1528-9885 is published monthly. Subscription price is $26.95. 7. The complete mailing address of known office of Publication is T&L Publications, Inc., 430 Princeland Ct., Corona, Riverside County, CA 92879-1300. Contact Person: Larry Lemieux. Telephone: (951) 371-8497. 8. Complete Mailing address of Headquarters or General Business Office of Publisher is T&L Publications, Inc., 430 Princeland Ct, Corona, CA 92879. 9. The names and addresses of the Publisher, and Associate Publisher are: Publisher, Larry Lemieux, 430 Princeland Ct., Corona, CA. 92879; Associate Publisher, Robin Lemieux, 430 Princeland Ct., Corona, CA 92879. 10. The names and addresses of stockholders holding one percent or more of the total amount of stock are: John Lemieux, 430 Princeland Ct., Corona, CA 92879; Lawrence Lemieux, 430 Princeland Ct., Corona, CA 92879; Audrey Lemieux, 430 Princeland Ct., Corona, CA 92879. 11. Known Bondholders, Mortgagees, and other security holders: None. 12. Tax Status: Has not changed during preceding 12 months. 13. Publication Title: Nuts and Volts 14. Issue Date for Circulation Data: October 2014-September 2015. 15. The average number of copies of each issue during the proceeding twelve months is: A) Total number of copies printed (net press run); 29,996 B) Paid/Requested Circulation (1) Mailed Outside County subscriptions: 20,832 (2) Mailed In-County subscriptions: 0 (3) Paid Distribution Outside the Mail including Sales through dealers and carriers, street vendor, and counter sales and other paid distribution outside USPS: 6021 (4) Paid Distribution by other classes of mail through the USPS: 0; C) Total Paid Distribution: 24,456; D) Free or Nominal Rate Distribution by mail and outside the mail (1) Free or Nominal Rate Outside-County Copies: 0 (2) Free or Nominal Rate In-County Copies: 0 (3) Free or Nominal Rate Copies Mailed at other classes through the USPS: 0 (4) Free or Nominal Rate Distribution Outside the mail: 1,634; E) Total Free or Nominal Rate Distribution: 1,634; F) Total Print Distribution: 26,091; G) Copies not distributed: 3,906 H) Total: 29,997; 16. Paid electronic copies: (A) 2,396; (B) Paid Print + Electronic copies: 26,852; (C) Total Distribution both print and electronic: 28,487; Percent paid circulation: 94.26%. Actual number of copies of the single issue published nearest the filing date is September 2015; A) Total number of copies printed (net press run) 28,984; B) Paid/Requested Circulation (1) Mailed Outside County subscriptions: 20,353 (2) Mailed In-County subscriptions: 0 (3) Paid Distribution Outside the Mail including Sales through dealers and carriers, street vendor, and counter sales and other paid distribution outside USPS: 5,801 (4) Paid Distribution by other classes of mail through the USPS: 0; C) Total Paid Distribution: 23,861; D) Free or Nominal Rate Distribution by mail and outside the mail (1) Free or Nominal Rate Outside-County Copies: 0 (2) Free or Nominal Rate In-County Copies: 0 (3) Free or Nominal Rate Copies Mailed at other classes through the USPS: 0 (4) Free or Nominal Rate Distribution Outside the mail: 1,375; E) Total Free or Nominal Rate Distribution: 1,375; F) Total Print Distribution: 25,236; G) Copies not distributed: 3,748; H) Total: 28,984; 16. Paid electronic copies: (A) 2,293; (B) Paid Print + Electronic copies: 26,154; (C) Total Distribution both print and electronic: 27,529; Percent paid circulation: 95.00%. I certify that these statements are correct and complete. Lawrence Lemieux, Publisher – 11/02/2015.

NEW PRODUCTS *Continued from page 21*

output frequencies from 9 kHz to 3.0 GHz. The instruments provide maximum output power up to +20 dBm and low SSB phase noise of -105 dBc/Hz, amplitude accuracy of ± 0.5 dB, and frequency resolution of 0.01 Hz at any frequency. An oven-controlled crystal oscillator time base provides < 5 ppb temperature stability and < 30 ppb/year aging stability.

The DSG800 RF signal generators provide conventional sweep functions (step, list, logarithmic, and linear), as well as analog modulation functions including amplitude modulation (AM), frequency modulation (FM), phase modulation (ΦM), and pulse modulation.

In addition, the DSG800 offers an optional pulse train generation capability for those customers needing to translate serial data onto an RF Link.

There are two models available:

DSG815 (9 kHz to 1.5 GHz) $1,999
DSG830 (9 kHz to 3 GHz) $3,599

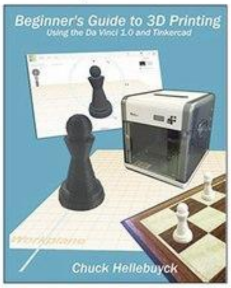
For more information, contact:
RIGOL Technologies, Inc.
www.Rigolna.com

For complete product details, visit our webstore!!

The Nuts & Volts WEBSTORE

GREAT FOR DIYers!

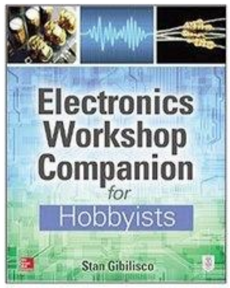
Beginner's Guide to 3D Printing by Chuck Hellebuyck



This book was written to answer the questions most beginners need answered. It covers many of the popular 3D printer choices and then uses the under $500 Da Vinci 1.0 from XYZprinting to show you how easy it is to get started. I take you through using Tinkercad software for creating your own custom designs. I go further and show you how to take a simple design and send it off to a professional 3D printer. This book was designed for anyone just getting started.

$19.95

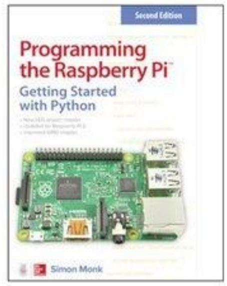
Electronics Workshop Companion for Hobbyists by Stan Gibilisco



In this practical guide, electronics expert Stan Gibilisco shows you, step by step, how to set up a home workshop so you can invent, design, build, test, and repair electronic circuits and gadgets. Electronics Workshop Companion for Hobbyists provides tips for constructing your workbench and stocking it with the tools, components, and test equipment you'll need. Clear illustrations and interesting do-it-yourself experiments are included throughout this hands-on resource.

$25.00

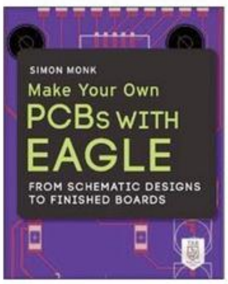
Programming the Raspberry Pi, Second Edition: Getting Started with Python by Jack Simon



This practical book has been revised to fully cover the new Raspberry Pi 2, including upgrades to the Raspbian operating system. Discover how to configure hardware and software, write Python scripts, create user-friendly GUIs, and control external electronics. DIY projects include a hangman game, RGB LED controller, digital clock, and RasPiRobot complete with an ultrasonic rangefinder

$15.00

Make Your Own PCBs with EAGLE by Eric Kleinert

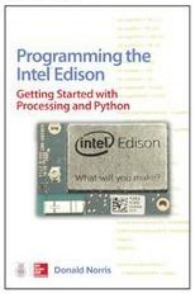


Featuring detailed illustrations and step-by-step instructions, *Make Your Own PCBs with EAGLE* leads you through the process of designing a schematic and transforming it into a PCB layout. You'll then move on to fabrication via the generation of standard Gerber files for submission to a PCB manufacturing service. This practical guide offers an accessible, logical way to learn EAGLE and start producing PCBs as quickly as possible.

$30.00

Programming the Intel Edison: Getting Started with Processing and Python by Donald Norris

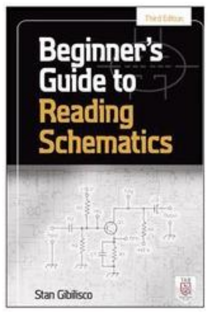
Learn To Easily Create Robotic, IoT, and Wearable Electronic Gadgets!



Discover how to set up components, connect your PC or Mac, build Python applications, and use USB, WiFi, and Bluetooth connections. Start-to-finish example projects include a motor controller, home temperature system, robotic car, and wearable hospital alert sensor..

$20.00

Beginner's Guide to Reading Schematics, 3E by Stan Gibilisco



Navigate the roadmaps of simple electronic circuits and complex systems with help from an experienced engineer. With all-new art and demo circuits you can build, this hands-on, illustrated guide explains how to understand and create high-precision electronics diagrams. Find out how to identify parts and connections, decipher element ratings, and apply diagram-based information in your own projects.

$25.00

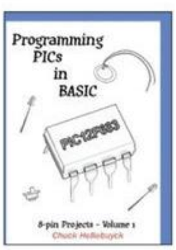
How to Diagnose and Fix Everything Electronic by Michael Jay Geier

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 gadgetry to cherished analog products of yesteryear.

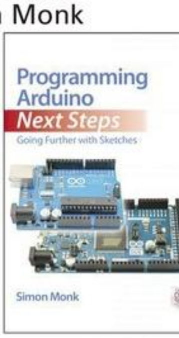
$24.95

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**

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



In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Also shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download. **$20.00**

>>> QUESTIONS

Reset Timer

A router and modem at a remote location periodically lock up, requiring a reset by unplugging to restart. Is there a simple circuit I could use to drop a relay out for about 30 seconds, every 24 hours? The relay contacts would be wired to drop out the power circuit to the devices. That way, when it does lock up, at least it would be reset again within a day.

#12151 **John C**
Chicago, IL

Motor Brake

I'm looking for a schematic for an electronic brake for a one phase 120 VAC motor. I had a commercial one, but it failed. I discovered it was potted with no way to tell what was inside.

#12152 **George Gray**
via email

LiPo Smart Charger

I'm looking for a smart charger for LiPo cells that can run from a solar panel. I've heard that the combination is incompatible because of fluctuations in output from the solar panel. Is this true? If so, is a workaround a larger panel?

#12153 **Michael Gage**
Weiner, AR

>>> ANSWERS

[#4153 - April 2015]
Over-Current for PWM Circuit

I have a Marlin P. Jones DC motor speed controller (Part 31566MD, 6-24 volts, 20 amps max). I need to add an over-current circuit to it.

I've found the 31566 schematic. The simplest solution that I've used in similar circuits is to use a ZXCT1009; this is a three-terminal current sensor. You put a 100 mR shunt in the positive line, attach the ZXCT1009 across it, and a current proportional to the motor current (about 1/1000 actually) flows out of the third pin to ground. Simply attach a resistor, say 1K ohm to ground, and you can read the motor current off this.

The most important advantage of this chip is the sense resistor can be grounded at the Arduino, so you won't be measuring any errors due to voltage drop in the ground wiring. The datasheet shows how to use it, and provides calculations on how to use PCB traces as current shunts.

Bob Turner
Salamander Bay, Australia

[#6151 - June 2015]
Transistor as an STDT Switch

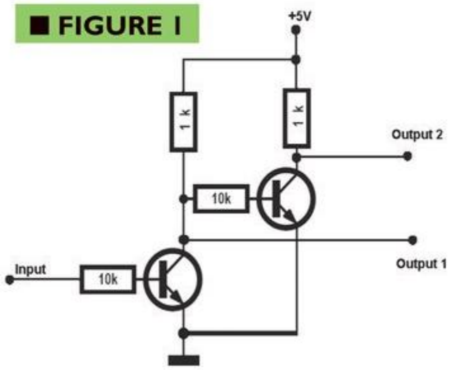
In a recent issue of NV, Roger Secura wrote the article "How to Use a Transistor as an SPST Switch." My question is how can an STDT switch be made using transistors, FETs, or other non-mechanical components? If it is possible, please post a circuit.

It's pretty simple. You just need another switch that's connected to the output of the first switch. These switches are commonly called inverters, meaning that the output is inverted from the input. This means that for a high level input, the output is a low level, and vice versa. So, all we need to create a bi-level output from a single input is a second inverter connected to the output of the first inverter. Bipolar transistors and FETs act pretty much the same way; the difference being that bipolar transistors are current-driven, whereas FETs are voltage-driven.

If you need more theory on transistor circuits, I refer you to a series of articles by Ray Marston in *Nuts&Volts* titled "Bipolar Transistor Cookbook." This is an eight-part series that *N&V* has graciously made available online at **www.nutsvolts.com/magazine/article/bipolar_transistor_cookbook_part_1**. Be sure to have plenty of reading time available, because it contains a train-load of information. **Figure 1** is a circuit depicting how to connect two transistors as an SPDT switch.

David Mason
Hazel Green, AL

FIGURE 1



[#10153 - October 2015]
Gold vs. Tin

I need to buy IC sockets in bulk for an upcoming project, and I'm debating whether the added cost is worth it to upgrade from tin to gold contacts. Am I paying for longevity or simply slightly lower contact resistance when I spend double or triple for a gold IC socket?

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

Gold plated contacts are highly resistant to corrosion. If you are designing equipment that is going to potentially operate in humid environments or where reliability is paramount, then using sockets with gold plated contacts can be a good investment. For most applications, though, conventional machined pin sockets provide excellent performance and the added cost of gold plating will not buy you much.

Most consumer equipment uses inexpensive stamped pin sockets and even those are normally adequate.

#1 **James Sweet**
via email

Depends on the environment that the project will be operating in. Gold plated sockets are usually specified for high reliability; that is, in mission-critical or life-critical applications. Tin plated contacts don't like lots of high frequency vibration, such as near heavy industrial equipment. It also doesn't tolerate humid environments. Gold plating is great for use in humid environments, and is better than tin in high vibration environments.

There's not a lot of difference in contact resistance, and you won't gain anything with gold. If your project will be used in a humid or corrosive environment, then I suggest a conformal coating be applied over the entire circuit board. Gold plating comes with its own unique problems; the main one being separation of the gold plating from the plated surface causing a failed connection, or worse, an intermittent connection.

I've used tin plated sockets for many years, and never had a problem that could be traced to the sockets. My money is on the tin plated sockets.

#2 **David Mason**
Hazel Green, AL

I would get the less expensive tin sockets, unless you will have the project in a corrosive environment (and you would also have to deal with possible wire corrosion). In using many sockets over the years, I have not had any problems with tin. Get this "springy" side-contacting socket at **www.digikey.com/product-detail/en/A08-LC-TT/AE9986-ND/821740** rather than the machined pin type of socket. I have had some machined pin sockets make bad contact with the IC.

#3 **Steven Barnicki**
Milwaukee, WI

■LOOK FOR ■SEARCH FOR ■FIND
Find your favorite advertisers here!

ADvertiser INDEX


littleBits
THE ULTIMATE INVENTOR'S TOOLBOX.
For the kids who bust things open to see how they work. The explorers. Inventors. Experimenters. With the littleBits Gizmos & Gadgets Kit, your mischief is your masterpiece. See more at
LITTLEBITS.CC/NUTSANDVOLTS
GIZMOS & GADGETS KIT
littleBits
GIZMOS & GADGETS
POWER

COMING TO CES 2016

If you thought our original AIR for WICED BLE module and *Atmosphere*, online development platform were amazing – just imagine the fun you'll have with our:

- extended family of AIR for WICED modules
- totally modular, hardware development tools
- *Atmosphere* 2.0 platform
- and all-new *Stratosphere* cloud platform

Register now to be part of the official launch!

AIR for WICED

Anaren ATMOSPHERE

Anaren STRATOSPHERE

Anaren®

What'll we think of next?®