

LINUX™ JOURNAL

Since 1994: The Original Magazine of the Linux Community

INTRO TO OSCAD
THE OPEN-SOURCE
CAD TOOL FOR
CIRCUIT DESIGN

WHY MICROSOFT
SHOULD
EMBRACE LINUX

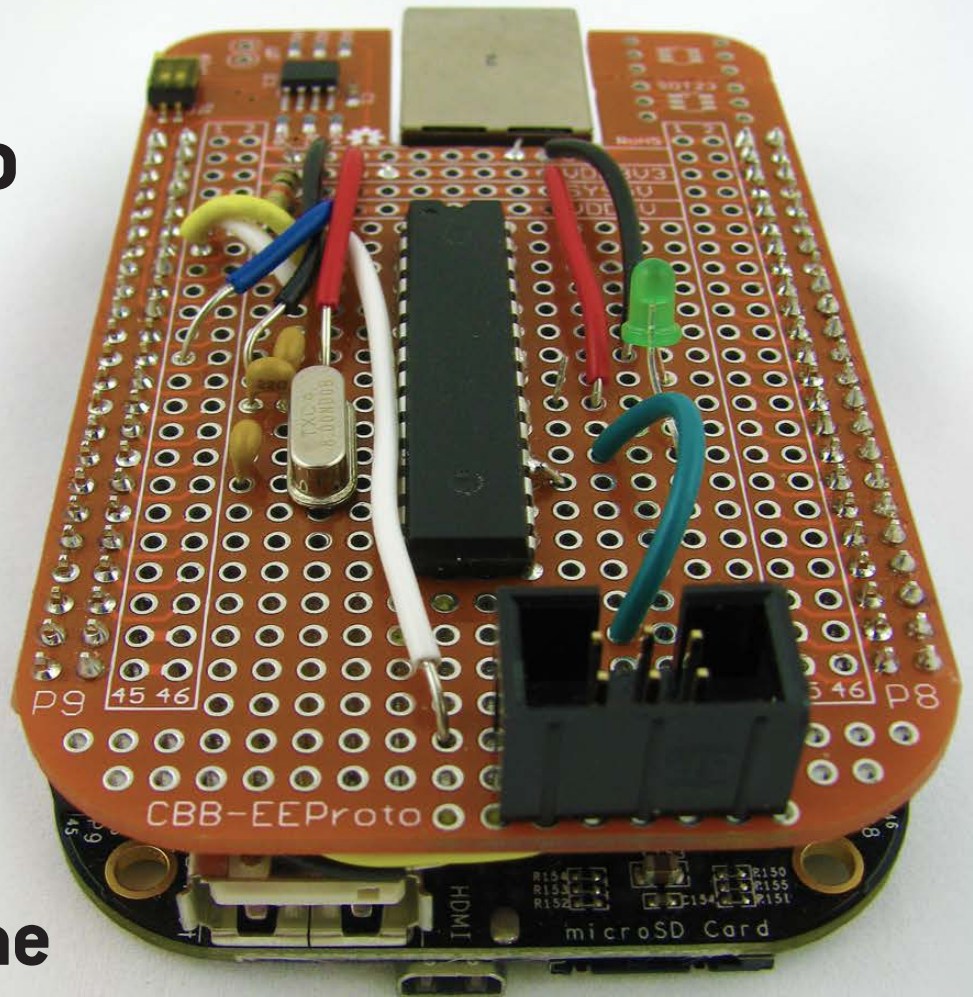
MAY 2014 | ISSUE 241 | www.linuxjournal.com

COOL PROJECTS

Build an
ATmega328p
Programmer
with the
BeagleBone
Black

Reglue:
Bridging the
Digital Divide

Hack the
Parrot A.R. Drone



A LOOK AT
SciPY FOR
SCIENTIFIC
COMPUTING

CREATE
OPEN-SOURCE
FORUMS WITH
DISCOURSE

BASH
PRIMER
FOR
SYSADMINS



WATCH:
ISSUE OVERVIEW





2014 JUNE 13TH - 14TH

TEXAS LINUXFEST

AUSTIN, TEXAS

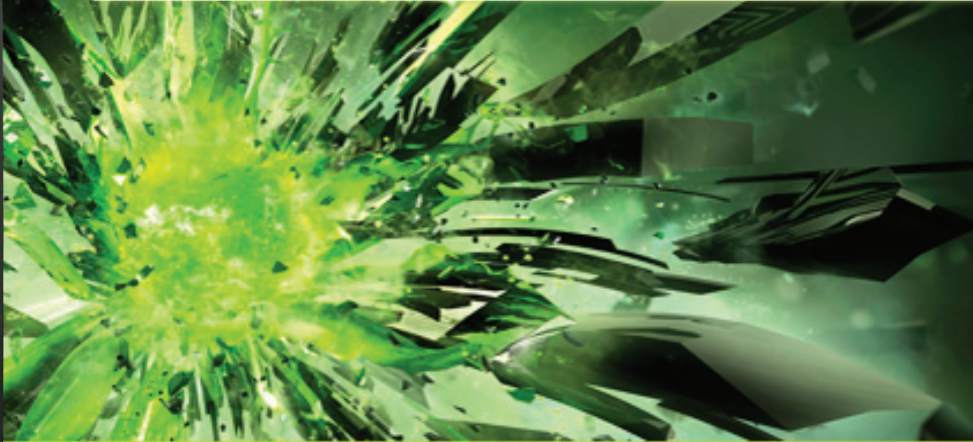
COME JOIN US FOR THE 5TH ANNUAL FESTIVAL.

LEARN FROM TALKS GIVEN BY INDUSTRY EXPERTS

MEET THE MANY VENDORS ON THE EXPO FLOOR

NETWORK WITH THE OPEN SOURCE COMMUNITY

REGISTER NOW AT TEXASLINUXFEST.ORG



Are you considering software-defined storage?



zStax StorCore ZFS Unified Storage from Silicon Mechanics is truly software defined storage.

From modest data storage needs to a multi-tiered production storage environment, the **zStax StorCore** ZFS unified storage appliances have the right mix of performance, capacity, and reliability to fit your needs.

zStax StorCore 64



zStax StorCore 104



May Case Study Feature



Learn how Vault Networks was able to build an enterprise-class cloud solution at a cost-effective price by turning to zStax from Silicon Mechanics.

Talk with an expert today: 866-352-1173
www.siliconmechanics.com/zstax

COOL PROJECTS

FEATURES

58 Hacking the Parrot A.R. Drone

Check out the potential for this semi-autonomous, largely automated quadcopter.

Bill Childers

68 Cross-Breeding the BeagleBone Black with the ATmega328p

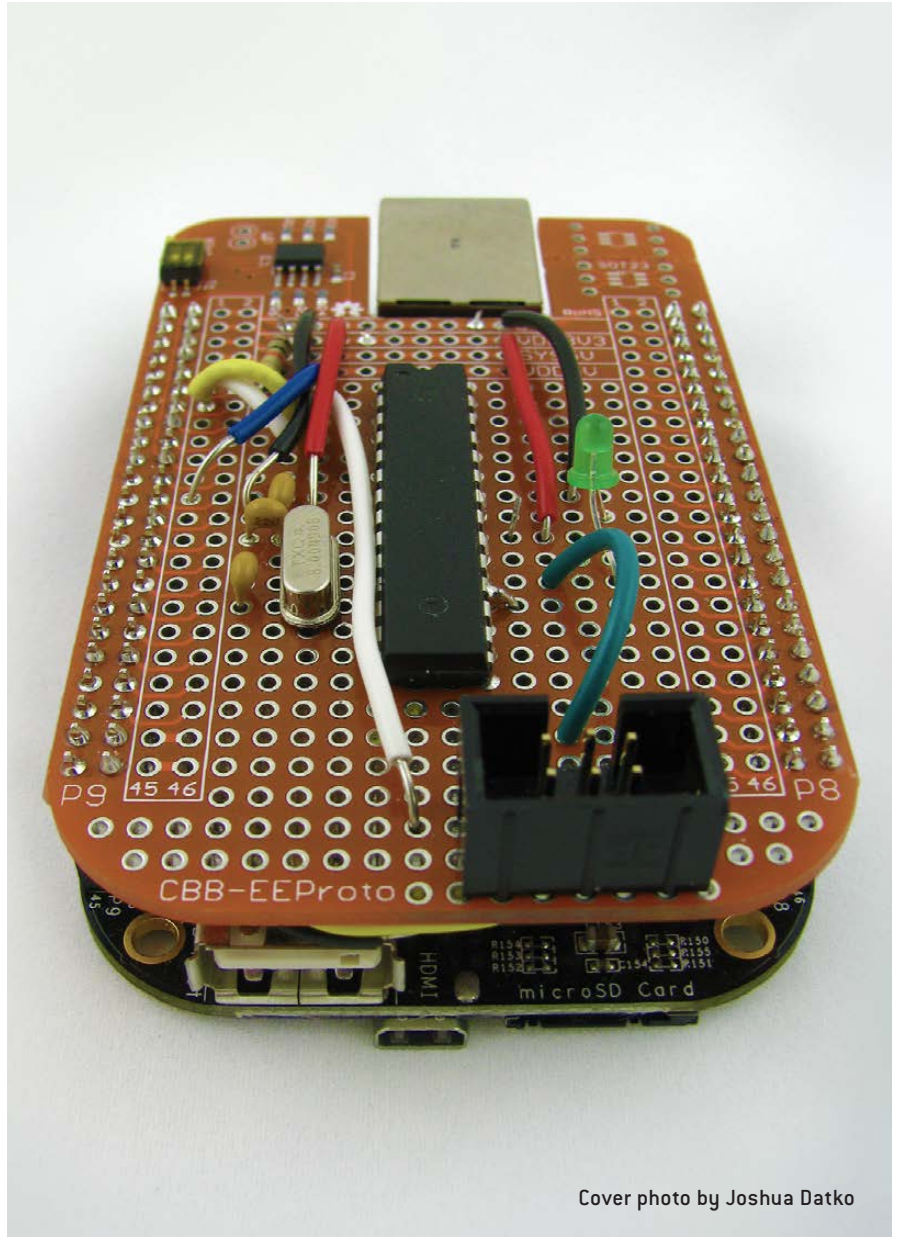
How to build up the hardware from basic components and configure the software to transform the BeagleBone into an ATmega328p programmer.

Joshua Datko

84 Reglue: Opening Up the World to Deserving Kids, One Linux Computer at a Time

Reglue gives free Linux computers to under-privileged children and their families.

Brian Conner



Cover photo by Joshua Datko

ON THE COVER

- Intro to OScad, the Open-Source CAD Tool for Circuit Design, p. 96
- Why Microsoft Should Embrace Linux, p. 114
- Build an ATmega328p Programmer with the BeagleBone Black, p. 68
- Reglue: Bridging the Digital Divide, p. 84
- Hack the Parrot A.R. Drone, p. 58
- A Look at SciPY for Scientific Computing, p. 24
- Create Open-Source Forums with Discourse, p. 30
- Bash Primer for Sysadmins, p. 46

INDEPTH

96 **Oscad: Open-Source Computer-Aided Design Tool**

Introducing Oscad, an open-source CAD tool for circuit design, simulation, analysis and PCB design.

Rakhi R and Kannan M. Moudgalya

COLUMNS

30 **Reuven M. Lerner's At the Forge**

Discourse

36 **Dave Taylor's Work the Shell**

Iterating Turns in *Zombie Dice*

40 **Kyle Rankin's Hack and / Tails above the Rest, Part III**

46 **Shawn Powers' The Open-Source Classroom Hulk Bash!**

114 **Doc Searls' EOF**

A Cool Project for Microsoft: Adopt Linux

IN EVERY ISSUE

8 **Current_Issue.tar.gz**

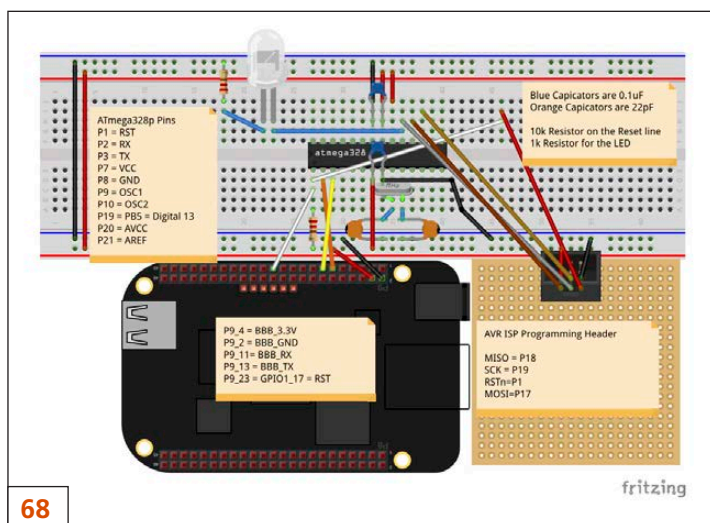
10 **Letters**

18 **UPFRONT**

28 **Editors' Choice**

54 **New Products**

119 **Advertisers Index**



LINUX JOURNAL™

Subscribe to
Linux Journal
Digital Edition
for only
\$2.45 an issue.



ENJOY:

Timely delivery

Off-line reading

Easy navigation

Phrase search
and highlighting

Ability to save, clip
and share articles

Embedded videos

Android & iOS apps,
desktop and
e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan • Adam Monsen

Publisher Carlie Fairchild
publisher@linuxjournal.com

Director of Sales John Grogan
john@linuxjournal.com

Associate Publisher Mark Irgang
mark@linuxjournal.com

Webmistress Katherine Druckman
webmistress@linuxjournal.com

Accountant Candy Beauchamp
acct@linuxjournal.com

**Linux Journal is published by, and is a registered trade name of,
Belltown Media, Inc.**

PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel

Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruiuzenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

Advertising

E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions

E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 980985, Houston, TX 77098 USA

LINUX is a registered trademark of Linus Torvalds.

Attend the Largest Dedicated Android Development Conference in the Universe!

AnDevCon May 27-30, 2014

Sheraton Boston

Get the best real-world Android
developer training anywhere!

- Choose from more than 75 classes and in-depth tutorials
- Network with speakers and other Android developers
- Check out more than 40 exhibiting companies

Take your Android development skills
to the next level!



Find out why you should go
to AnDevCon! Watch the videos
at www.AnDevCon.com

Register Early
and SAVE!



Register Early and Save at www.AnDevCon.com

AnDevCon™ is a trademark of BZ Media LLC. Android™ is a trademark of Google Inc. Google's Android Robot is used under terms of the Creative Commons 3.0 Attribution License.

A BZ Media Event



#AnDevCon





SHAWN POWERS

Cooler Than Minnesota in January

By the time this issue goes to press, the snow in my backyard probably will be only knee deep in the shade. It's the middle of April now, and there's still a six-foot snow drift on my back deck. Needless to say, it's been a long, cold winter.

This issue of *Linux Journal* is cool as well, but in a far more enjoyable way! The first time I was published in *Linux Journal*, it was in a Cool Projects issue, and every year it means a bunch of really nerdy ideas and awesome recipes for weekend projects. This year is no different.

Reuven M. Lerner starts out the issue with an article on Discourse. Web forums certainly aren't anything new, but Discourse is a modern-looking, modern-functioning, discussion forum. Its slick interface

and elaborate underpinnings mean it can be challenging to install, but Reuven walks through the things that make it tick and explains why it's worth the hassle. Dave Taylor also shows that hard work pays off, as he ends his series on the script-based version of *Zombie Dice* he's been building the past few months.

Kyle Rankin finishes off his series as well. The Tails distribution is a security-minded Linux system that has tons of privacy-enabling bells and whistles. Kyle describes its advanced features, reaching above and beyond what most folks do with it.

My column is also about tools this month—basic shell scripting tools. Dave Taylor provides advanced scripting every month, but for many of us, the basics are still pretty cryptic. If you don't know an IF/THEN conditional statement from a range-numerated FOR loop, this article should be useful.



VIDEO:
Shawn Powers runs through the latest issue.

I'm a little angry with Bill Childers this month, because he has a great article on hacking a Parrot A.R. Drone. I think it just about guarantees I'll purchase a drone in the near future to hack myself. Bill explains that the readily available "toy" drones are incredibly robust machines capable of far more than just scaring cats.

In true "Cool Projects" fashion, Joshua Datko combines two awesome bits of technology into a teeny, tiny, super-powerful device. He combines the surprisingly powerful BeagleBone Black with an ATmega328p microprocessor. The BBB plus Arduino? It sounds like a nerdy version of, "hey, you got peanut butter in my chocolate"—it's a marriage made in heaven.

Rakhi R and Kannan M. Moudgalya describe the OScad program this month. CAD software can be ridiculously expensive if you buy a proprietary option, and often the free and open-source options aren't quite as robust as you'd like. With OScad, however, you can design and test PCBs using 2-D and 3-D interfaces. The software not only helps organize the layout of your boards, but it will simulate the actual operation of the circuits as well! OScad is a cool project itself, but it's also a great tool for *creating* even cooler projects!

Finally, Doc Searls and Brian Conner prove that cool Linux projects aren't always something that can be done over a long weekend. Brian Conner interviews Ken Starks about his Reglue organization. Reglue recycles hardware and uses Linux to provide usable computer systems for education. We all know that Linux is changing the world, but that change only occurs when we have people like Ken wielding it. Doc takes an interesting look at Linux as well, but he posits what might happen if Microsoft were to adopt Linux. I know you're thinking it'll be a Minnesota-cold day in ____ before Microsoft turns to Linux, but crazier things have happened.

This month's focus is always one of my favorites. We also have product announcements, tech tips and more. The cool projects, however, are something that everyone seems to enjoy. They warm my heart, and after a winter like the one we're finally ending, a warm heart is a welcome effect. ■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

letters



Choice of Words

I am a longtime subscriber to *Linux Journal* and generally am very happy with the magazine. Unfortunately, I was very sick in December and early January and only got around to catching up now.

In Shawn Powers' mini-article on Pixlr in the December 2013 issue, I was surprised and disappointed to see his casual use of an epithet used against the disabled. I have always lived with the name of The GIMP because it purports to be an acronym. The use of the word gimp or gimpy is offensive to many who live with physical disabilities and their friends. This is especially true of the manner in which

it was used here to suggest something that is weak or worthless.

However appealing the matching sounds might be, I suggest you resist the temptation much as you have so far managed to do with nigger, jew, fag and retard.

—Keith Nunn

I appreciate the letter, and hope you know I meant no offense. I can see at the end claiming Pixlr wasn't "gimpy" was perhaps a poor choice of words. Thank you for bringing it to my attention. I far prefer to hear about such things than to remain ignorant.—Shawn Powers

Cat Photos Encryption

First of all, I am very grateful to Shawn Powers for his tips and tricks, which always are fun to read and interesting, especially his latest article about encryption, which is very well explained and crystal clear, as understanding how encryption actually works behind the scenes is not trivial. [See Shawn Powers' "Encrypting Your Cat Photos" in the January 2014 issue.]

This helped me to encrypt my USB

sticks successfully. However, I ran into an error regarding the formatting command:

```
mkfs.vfat /dev/mapper/my_crypto_disk -n my_crypto_disk
unable to get drive geometry, using default 255/63
```

I tried again with the option `-F 32` for a fat32 partition type:

```
mkfs.vfat -F 32 /dev/mapper/my_crypto_disk -n my_crypto_disk
unable to get drive geometry, using default 255/63
```

This is not harmful, and the USB drive is encrypted and can be mounted after the passphrase is entered. I just wanted to mention it to see if anyone had an explanation. Maybe it's a bug from my distribution (Kubuntu).

Keep up the good work.

—Patrick Wolf

I haven't seen your exact error, so I'm completely guessing (always dangerous), but I suspect it might be the USB drive itself, or more specifically, its embedded controller. Perhaps the controller itself doesn't give the geometry when queried. I'm curious if it would give the same error if encryption were taken out of the picture. Either way, thanks

for the letter.—Shawn Powers

Ubuntu vs. Fedora—a Reader's Test

In testing Fedora against Ubuntu, trying to be unbiased, I used the full Ubuntu 64-bit on what could be labeled an under-powered machine: a 1GB of RAM Gateway media tower from 2005 with a 512MB NVIDIA EVGA card and a Sound Blaster Audigy 2ZS. In terms of speed and boot up, Fedora is the winner, but in terms of stability and in staying with gaming, the Ubuntu blows it out of the water—no hangups, no slow downs and even when loading the comprehensive OpenSUSE, full KDE, my system still stays with it. As unimpressed as I am, the Internet browsing through Midori was far worse than anything I've seen since Windows 98. In switching to Google Chrome, it hits on usability until it experiences far too many out of memory errors. If this is an LXDE release for Fedora 20, it seems like it's not ready for prime time, and they may want to create long-term support options before they go through with systems that are hurt by less time in working situations. This entire release review was done on Fedora 20, and I'd really like to, well, like

[LETTERS]

Fedora again, having used it since Fedora 13. But all in all, it's not really an operating system as much as it is becoming an exercise in futility.

—**Joseph Ziehm**

I don't have any experience with LXDE on Fedora, but in general, I think the desktop manager isn't quite as mature as its counterparts. Although not quite as efficient (depending on whom you ask), I've been very impressed with recent releases of XFCE. My main operating system currently is Debian with XFCE.—*Shawn Powers*

Mars Needs Older Women

I would like to weigh in on the discussion involving the lack of women in the Linux field (and IT in general). I found a rare contemporary in Rose Dlhopsky's letter in the February 2014 issue. We are both females in our fifties who have been using Linux for a long time.

I installed Linux for the first time in 1999. I didn't know anyone else who used it, so I had to figure everything out myself. Sometimes it worked, sometimes it didn't. I avoided the local LUG because it seemed to be filled with young men who didn't take this old gal too seriously. In fact, there are almost *no* women my age in IT.

I fell in love with Linux and technology. I taught myself Web design and hosted my own Web server for many years. I learned Python and wrote/released a program I'm proud of. I taught computer apps in the public schools and in private institutions. I am proud to be a self-taught geek, but I still feel like an oddball at Linux conventions and gatherings where I see no one who looks like me.

I grew up in a different world. In the 1960s in the US, women were not expected ever to work outside the home. Girls were not allowed in the shop classes, but were required to take Home Economics (cooking and sewing). The only vocational-type class the girls could take was typing, because, you know, they might be secretaries. (Luckily, keyboarding turned out to be a useful skill a few generations later.)

Women were not allowed to have credit in their own names, and they could never buy a car, a house or have a credit card without a man to co-sign. Birth control became available, and for the first time, women could control when and if they got pregnant.

Somewhere while I was growing up, the rules changed. Suddenly, women were expected to work, although girls of my generation had never been prepared for that world. Economics forced most families to require two incomes. Most women gravitated toward traditional fields: secretaries, teachers and nursing.

Somehow, I found and nourished my inner geek, but my community is online where my age and sex doesn't matter, and is usually unknown. But in the real world, it is a lonely place for an older, female geek.

—Michelle Blowers

In Response to Okay, Google

Regarding Shawn Powers' "Okay, Google" piece in the February 2014 Upfront section: yes, *Star Trek IV* was awesome. To this day, I still mimic Scotty picking up the mouse and speaking to it when I find that a PC or Mac's performance is slower than what I feel it should be—especially when assisting clients/customers with computer issues and their patience is wearing thin, it helps to lighten the moment. Using my Nexus 10, I use voice commands occasionally but have found its use limiting or wanting still. Eventually, this will be improved but not for everyday use yet. Not to

mention that talking to my Nexus has had the effect of causing my cube neighbors to have a phone in hand ready to call security.

This brings me to the point that I find voice interaction with a device as archaic, and it tends to disturb others around me (imagine doing that on a plane)—at least as a sole or primary way of interaction. I believe it has its uses, but I would find it annoying and limiting after a while. Our interaction with "people" is more than voice and listening. It also includes visual body language. So, not only do I want to be able to "talk" to my computing device, I want it to "see" my body language and cues, and with that ability, I want to be able to give it commands with hand gestures or even facial expressions (brings a whole new definition to the "cube-dance"). The computing device should not always respond back audibly or visually but also with perhaps some robotic gesture—whatever is an appropriate (based on programming/user preference) response to indicate that the device understood and is carrying out the command.

However, so that we don't turn whole offices into dance clubs, the gestures detected should be relatively small.

[LETTERS]

The voice commands given should be almost whispers, and the visual cues limited to the display devices in the immediate location of the user and the robotic device limited to a certain area. Privacy is of concern here too. This brings up how to detect such small gestures and hear nearly inaudible commands. Today's devices (mics, Kinect and so on) are limited to detecting only loud voices or exaggerated gestures (I do not want to have a *Minority Report*-like special room or office either).

Science-fiction author, Vernor Vinge, in the book *Rainbows End*, has a vision of nano-technology that is embedded in clothing, small hearing-aide devices and contact lenses. The computing device is no larger than our current cell phones. Clothing contains all the "detection" devices to read body language and gestures. Hearing aides hear your voice as well as respond with audio. Contact lenses have embedded nano-technology that polarizes light to make visual responses "part of the user-acute vision" without giving the feeling of looking through a heads-up display, but rather the visual cues become part of what you are looking at. The contact lenses also serve as a

camera, detecting the focused image on the back of the retina. The computing device and nano-technology/devices are able to communicate wirelessly (in a close proximity) in a mesh network fashion.

That sort of technology, while it maybe a ways off, is sounding more plausible each day. I believe Vernor Vinge is on to something here. This type of interaction with computing devices is much more preferable to me. I can ride in a subway and do work or communicate with someone across the country or in another country doing nothing more than tapping my knee or whispering to myself (along with old lady beside me). I also can stare blankly at the floor and see what my next stop is or read the news. I'd like to look out over a landscape and have the name of mountains or even buildings in a cityscape appear or have the name of some specific landmark appear. Or, I'd like the automatic translation of a foreign language that I can hear but the person speaking cannot. And when I am wearing this technology, I do not want to look like Iron Man or RoboCop. In fact, most people should not be able to see or detect that I am wearing/using it. I just

want the technology to be an augmentation to my daily life.

Although a “cranial implant” may sound like a cool way to interface, I find that “invasive” (not to mention that many science-fiction authors have been able to envision its abuses). Of course, all technology has the potential for abuse, but the above nano-technology has fewer risks of being “invasive”, as it uses less power and doesn’t actually penetrate the body. And, it wouldn’t turn the office place into

a dance club or make me look like Iron Man. I am sure this technology could be set up for abuse, but it is more palatable and compatible with everyday life, in my opinion.

—mshives

I do often tease about a cranial implant, and part of me would like some sort of direct interaction, but really my feelings are similar to yours, especially when it comes to speaking out loud to/from devices. In all honesty, I hope sub-vocal communication can be perfected

Powerful: Rhino



- Rhino M4700/M6700**
- Dell Precision M4700/M6700 w/ Core i7 Quad (8 core)
 - 15.6"-17.3" FHD LED w/ X@1920x1080
 - NVidia Quadro K5000M
 - 750 GB - 1 TB hard drive
 - Up to 32 GB RAM (1866 MHz)
 - DVD±RW or Blu-ray
 - 802.11a/b/g/n
 - Starts at \$1375
 - E6230, E6330, E6430, E6530 also available

- High performance NVidia 3-D on an FHD RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop’s features
- One year Linux tech support — phone and email
- Three year manufacturer’s on-site warranty
- Choice of pre-installed Linux distribution:



Tablet: Raven



- Raven X230/X230 Tablet**
- ThinkPad X230/X230 tablet by Lenovo
 - 12.5" HD LED w/ X@1366x768
 - 2.6-2.9 GHz Core i7
 - Up to 16 GB RAM
 - 750 GB hard drive / 180 GB SSD
 - Pen/finger input to screen, rotation
 - Starts at \$1920
 - W530, T430, T530, X1 also available

Rugged: Tarantula



- Tarantula CF-31**
- Panasonic Toughbook CF-31
 - Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
 - 13.1" XGA TouchScreen
 - 2.4-2.8 GHz Core i5
 - Up to 16 GB RAM
 - 320-750 GB hard drive / 512 GB SSD
 - CF-19, CF-52, CF-H2 also available



[LETTERS]

so we can silently interact with a computer system. You played the nerd card with Vernor Vinge, so I'll do the same—the “jewel” Ender used to communicate with Jane in Speaker for the Dead I think was an idealistic evolution of the current Bluetooth earpiece. We really do live in an awesome time, no?—Shawn Powers

Electronic vs. Paper

It strikes me that those people who prefer to read a paper edition of *Linux Journal* could print out their PDF version (two sides, two pages/side) much more cheaply than *Linux Journal* could print and mail it. You could even keep them in a loose-leaf binder.

I myself find reading the PDF on a Xoom tablet quite satisfactory except for the time it takes Xoom to load the pages.

—Eric Beversluis

It is admittedly not as nice to read a printout as opposed to a regular paper magazine. That said, tablets are making digital magazines far more enjoyable to read than they were even a year ago. I also like the potential for interactive features and clickable links that the paper version just doesn't

support. Like I've mentioned before, however, there certainly are options if people want a physical copy, it's just not a service we offer ourselves.—Shawn Powers

Linux Journal iOS App

I've been an *LJ* subscriber for about six years now. I really grew tired of reading all of the negative feedback when *LJ* went digital. At the time, I was downloading the PDF versions and reading them in a PDF reading app on my iPad. Somehow, I just now discovered the iOS app. I primarily use my iPad for reading, and I am quite pleased with the *LJ* iOS app. I'm impressed with the ability to switch to text mode, the options for zooming in and out, and increasing the font size when in text mode. Overall, I believe it gives me an even better experience than reading the PDF version. It also gives me the closest experience to reading it the “old-fashioned way” (on paper) as possible. Keep up the good work!

—Jonathan Calloway

The “app” was a bit rough around the edges at first, but I agree, it's getting nicer. I enjoy the one-click download feature the app provides, as opposed to somehow copying

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Recently, some kernel developers tried to clarify the caveats involved in configuring the **DEBUG_INFO** option in the Linux kernel. Originally, **Borislav Petkov** patched the **KConfig** description of that feature to say that it would cause “huge bloat”. But, **David Rientjes** didn't feel this quite captured the issue. And as **Linus Torvalds** said, some object files would be four times larger with this option enabled. The **fs/built-in.o** file, for example, went from 2.8MB to 11.8MB. So although this didn't affect the size of the runnable **vmlinux** binary file, it did affect the object files, which could cause problems on small systems running tests. Linus said, “I suspect a lot of people are in denial about just how *horrible* the overhead of debug builds are.”

This issue tended to come up when testing the **allmodconfig** build target, because it would enable **DEBUG_INFO** along with everything else. The main reason for compiling the **allmodconfig** build target is just to ensure that the kernel actually will compile, as opposed to running the resulting binary. **Andrew Morton** remarked that he always

disabled **DEBUG_INFO** by hand when testing **allmodconfig** for precisely this reason. Because he didn't plan to run the **vmlinux** binary, there was no need to build in all the debugging data. He suggested clarifying **Borislav's** wording to say that the option would bloat object files on disk and increase build time.

But this didn't fully solve the problem, because the **allmodconfig** build target still enabled **DEBUG_INFO** by default. And, there would be value in letting developers and interested users test the build process in as easy a way as possible, without inflicting bloated slowness on them.

Several folks, including **Ingo Molnar**, tried to design a new **KConfig** option that still would work for testing, but that would avoid the bloat. Eventually, Linus decided to change the **DEBUG_INFO** option so that it would disable the **COMPILE_TEST** option.

That almost failed to be good enough, because **Andi Kleen** pointed out that he actually did run the **vmlinux** binary in some circumstances, and so the massive debugging data wasn't merely bloat to him, it was a real part of his test. But since

his was a very unusual case, Linus told him just to edit the config files by hand if he wanted to re-insert the debugging data.

Recently, **Andy Lutomirski** announced **virtme**, a set of scripts that did the hard work of setting up virtual machines for testing compiled Linux kernel binaries with **virtFS** in **KVM**. He'd gotten fed up with having to do the whole mistake-prone process by hand, so he wrote **virtme** to automate it.

The idea was that **virtme** wouldn't simply boot the kernel into a virtual machine, it also would set up a user environment that was ready to perform tests, download additional software and so on. Andy believed that in time, **virtme** could automate an entire testing regimen and report the results, with the user giving just a single command to start the ball rolling.

Some filesystems have stirred up darker regions of the **VFS** (virtual filesystem), necessitating odd changes. It all started when **Ilya Dryomov** noticed that some recent **ACL** (access control list) patches seemed inconsistent. Specifically, the **Ceph distributed filesystem** didn't work well with the new code. He posted a patch to fix this.

Linus Torvalds, however, in looking over the consistency issues, discovered that some of the **ACL** code relied on passing **inodes** around. This

was a problem, because distributed filesystems didn't necessarily make the same assumptions about things like inodes that single-disk filesystems could, such as assuming an inode wouldn't change suddenly or assuming each inode was unique.

A better approach, he felt, was to pass actual file paths, which didn't suffer from those issues. The problem, as **Christoph Hellwig** pointed out, was that some of the **VFS** code was so deep and dark, it became difficult to construct the data the **ACLs** would need to pass around.

Linus took a look and confirmed the difficulty. He felt that it might not even be worth fixing, if the problem affected only an unusual **Ceph** filesystem case. He suggested that the **Ceph** developers might want to "bite the bullet" and fix the problem on their end.

But, this wasn't good enough. As **Christoph** pointed out, the **Plan9 filesystem** had the same issue, as did **CIFS**. None of these filesystems would be able to use the new **ACL** helper functions without an in-kernel fix for this problem.

So Linus went back into the kernel depths and tried to push the needed file path data as far as he could through the call chain. Ultimately, he did find a way to push the data just far enough to reach where it needed to go. But getting

it any further in, he said, would be much harder. Fortunately, that wouldn't be necessary.

Al Viro felt this was a bit over the top and didn't see the need to pass file path information when inode data would work just as well. But, Linus explained that:

Some network filesystems pass the *path* to the server. Any operation that needs to check something on the server needs the *dentry*, not the inode.

This whole "the inode describes the file" mentality comes from old broken UNIX semantics. It's fundamentally true for local UNIX filesystems, but that's it. It's not true in general.

Sure, many network filesystems then emulate the local UNIX filesystem behavior, so in practice, you get the UNIX semantics quite often. But it really is wrong.

Part of Al's objection was based on the idea that networked filesystems like CIFS couldn't support hard links. But lo and behold, it turned out they really could. This made no sense to Al until he realized this could be accomplished with **Samba** on a UNIX server. But he detested it, along with Linus' entire patch, even while acknowledging that it probably was necessary. He blamed **Andrew Tridgell** for supporting hard links in the first place, but at that point **Jeremy Allison** said, "Actually you have to blame me for that. Tridge always *hated* the UNIX extensions."

—**ZACK BROWN**

They Said It

It's not the hours you put in your work that counts, it's the work you put in the hours.

—**Sam Ewing**

If you don't make mistakes, you're not working on hard enough problems. And that's a big mistake.

—**Frank Wilczek**

No man who ever held the office of president would congratulate a friend on obtaining it.

—**John Adams**

This is why I loved technology: if you used it right, it could give you power and privacy.

—**Cory Doctorow**

For one person who dreams of making fifty thousand pounds, a hundred people dream of being left fifty thousand pounds.

—**A. A. Milne**

Android Candy: Waze

I have a love/hate relationship with Waze. The idea of peer collaboration regarding traffic, combined with the technology to accomplish it on an enormous scale is truly amazing. Yet, every time I've used Waze myself, it's been an exercise in frustration. It has insisted I turn left off a bridge, and then it refused to reroute me when I didn't. On one trip, it had me get off every freeway exit, only to get back on the freeway immediately with the adjacent onramp. That doesn't seem to be the case for everyone, and perhaps it's simply because I live in a fairly rural area, and there aren't many users apart from me in the area.

Waze is a turn-by-turn GPS application. For most people, it works well and gives quick and easy directions to get from point A to point B. It also has a very robust social aspect, which is really what sets it apart. Did you just pass a police officer setting up a speed trap? Click on the Waze app, and it will warn fellow Waze users as they approach. Is there an accident? Tell Waze about it, and it will warn other users and route them around the slowdown.

There is a certain competitive aspect to Waze as well. Who has mapped the most new roads? How many miles have

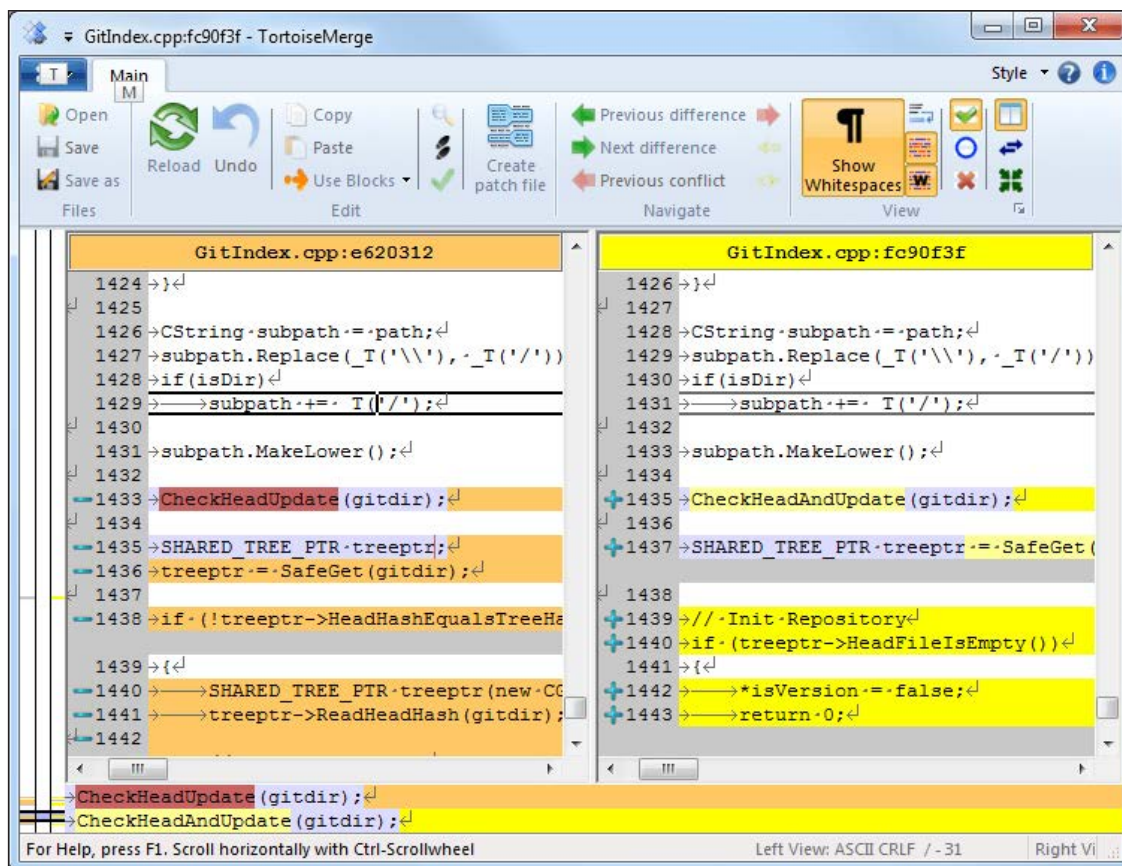


you driven with Waze? From a mapping aspect, the truly amazing part of the entire system is that Waze watches the routes you take and uses that information to guide others. For that reason, Waze prefers you have the app running whenever you're on the road, whether or not you need it for guidance. Your driving adds to the routing algorithms, ideally making things easier for other drivers in the future.

Waze is available at the Google Play store: <https://play.google.com/store/apps/details?id=com.waze>. Check it out for yourself and see if the navigator in your phone is awesome, or if it wants to murder you, like mine does for me.

—**SHAWN POWERS**

Non-Linux FOSS: Git Yer Tortoise On!



completely foreign to the regular point-and-click world they're used to.

Enter TortoiseGit. With a familiar GUI interface to the underlying git system, TortoiseGit can make

Git has become the most popular version-tracking platform around for open-source projects. Whether you're using GitHub, Gitorious, Bitbucket or similar, or even if you're hosting the git repository yourself, accessing the code is something us Linux users take for granted. For Windows users, what seems commonplace to us (typing `git clone` on the command line, for instance) is

Windows-based open-source developers feel right at home. It's open source itself, and it's part of the Tortoise family, which includes TortoiseSVN for Subversion repositories and TortoiseCVS for the Concurrent Versioning System. To check out the whole family of Windows-based Tortoise clients, see the Wikipedia page at <http://en.wikipedia.org/wiki/TortoiseGit>. —**SHAWN POWERS**

Siege Your Servers!

```

top - 13:06:47 up 7 days, 16:04, 2 users, load average: 3.00,
Tasks: 113 total, 10 running, 103 sleeping, 0 stopped, 0 z
%Cpu(s): 94.8 us, 2.6 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 st, 0.0 sr
KiB Mem:  448736 total,  316132 used,  132604 free,  5250 KiB
KiB Swap: 2321512 total,  0 used,  2321512 free,  15114 KiB

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+
 2066 www-data  20   0 41128 6904 1816 R   6.1   1.5   0:00.73
27248 www-data  20   0 41400 8984 3528 S   5.8   2.0   0:07.43
 2032 www-data  20   0 41132 6928 1840 S   5.5   1.5   0:03.69
 2035 www-data  20   0 41128 6936 1840 S   5.5   1.5   0:03.73
 2037 www-data  20   0 41260 6920 1840 S   5.5   1.5   0:03.57
 2070 www-data  20   0 41128 6900 1816 R   5.5   1.5   0:00.63
 2072 www-data  20   0 41128 6900 1820 S   5.5   1.5   0:00.35
27214 www-data  20   0 41400 9252 3652 S   5.5   2.1   0:08.53
 2033 www-data  20   0 41260 6920 1840 S   5.2   1.5   0:03.89
 2069 www-data  20   0 41128 6908 1816 R   5.2   1.5   0:00.63
 2065 www-data  20   0 41128 6900 1816 S   4.8   1.5   0:00.69
 2068 www-data  20   0 41128 6900 1816 R   4.8   1.5   0:00.62
 2071 www-data  20   0 41128 6904 1816 R   4.8   1.5   0:00.61
 2034 www-data  20   0 41260 6920 1840 R   4.5   1.5   0:03.64
 2067 www-data  20   0 41128 6900 1816 R   4.5   1.5   0:00.73
27247 www-data  20   0 41144 7872 2740 R   4.5   1.8   0:06.69
27266 www-data  20   0 41392 9228 3628 S   4.5   2.1   0:06.06

```

My little Raspberry Pi server didn't crash while under siege, but it certainly was taxed!

Setting up Web servers is fairly simple. In fact, it's so simple that once the server is set up, we often don't think about it anymore. It wasn't until I had a very large Web site rollout fail miserably that I started to research a method for load-testing servers before releasing a Web site to production.

There are many, many options for load-testing a Web site. Some are commercial, and some are specific to a particular type of Web server

(there are a few SharePoint-specific load testers, for example), but I struggled to find a simple "simulate a bunch of traffic" method to see how a server would handle load.

As is usually the case, many months after I needed the tool, I stumbled across it. A very simple, yet powerful tool named Siege is available in most

distributions. Developed by Joe Dog Software, Siege does exactly what's on the tin: it lays siege to your Web server. It has lots of options and features, but by simply specifying a Web URL, Siege will launch a ton of generated hits on your server to see how it performs. To try Siege, you can search your software repository, or head over to <http://www.joedog.org/siege-home> to get the program directly from the developer.—**SHAWN POWERS**

SciPY for Scientists

In my last article, I looked at NumPY and some of its uses in numerical simulations. Although NumPY does provide some really robust building blocks, it is a bit lacking in more sophisticated tools. SciPY is one of the many Python modules that build on NumPY's. In fact, SciPY has become sort of the de facto science package in Python programming. If you have a scientific problem you are trying to solve, you could do worse than starting with SciPY. Not only are there more advanced functions and objects available to do linear algebra, but there also are functions and objects to handle calculus, interpolation, signal processing and Fast Fourier Transforms, among others. So many functions are available, they actually are grouped together into sub-packages. In this article, I take a quick look at what sorts of functions are available and how to use them to get some serious work done.

To start, you need to import the main `scipy` module. You would do this with the usual:

```
import scipy
```

This imports the common set of functions and objects used in SciPY. It also imports the most-used parts of NumPY, because they are so fundamental to the work for which SciPY is used. If you need anything else from NumPY, you need to import the NumPY module explicitly. In many cases, that is something you will want to do anyway. All of the extra functions in the individual sub-packages need to be imported explicitly. So, if you want to do some signal processing, you would need to use this:

```
from scipy import signal
```

The simplest package in SciPY probably is the constants sub-package. This package provides a basic set of physical constants that are most used, like pi or Avogadro's number. It also includes a much larger set of constants from the 2010 CODATA database. These physical constants are stored as a tuple of value, unit and uncertainty, and they include items as diverse as the alpha particle mass to the Wien wavelength displacement law constant. The `scipy.misc` sub-package

contains all of those bits and pieces that don't really fit anywhere else. Here, you can find functions like `factorial` (to calculate the factorial of a number) and `imread` (to read an image file into Python).

Linear algebra is one of the heavy uses of computational code. SciPY includes a sub-package called `linalg`, which is a wrapper for the package `linalg` within NumPY. All of the functionality from NumPY is included in `scipy.linalg`, along with several other functions. In the NumPY module, these linear algebra functions may or may not be handled by external libraries, depending on how NumPY was compiled. With SciPY, this is no longer an option. It needs to be compiled with the ATLAS LAPACK and BLAS libraries to handle the actual numerical work in an optimized fashion. There are functions to handle things like finding an inverse, determinant or transpose of a matrix. If you need to solve a system of equations, you can do so with a single function call. If you start with a coefficient matrix, A , and a right-hand side vector, b , you can find the solution vector for your system with:

```
from scipy import linalg
linalg.solve(A,b)
```

In many physics and engineering problems, you need to find eigenvalues and eigenvectors. The `linalg` sub-package provides very fast functions for doing that as well.

Most people default to using R to do statistics, but you don't have to. SciPY includes a `stats` sub-package that provides many of the functions you will need in the majority of cases. The `describe` function will give you the basic statistical description of a vector of samples. This includes the mean, variance, skew and kurtosis. Once you have some basic statistics, you probably will want to run a t-test to see how well your data matches your model. You can do this with something like:

```
stats.ttest_1samp(x, m)
```

where x is your data and m is your model. This will give you a t-statistic and a p-value. Just as in R, there are many more complicated statistical functions available to you.

A topic near and dear to my heart is solving differential equations. SciPY can help with that task too. The sub-package you need is named `integrate`. There are two sets of functions, one that takes a function object as the input and one that takes a set of fixed samples. You

can do single, double and triple integrations on a function object with the functions `quad`, `dblquad` and `tplquad`. If you have data from some experiment, you integrate it with the trapezoidal rule, Simpson's rule or Romberg Integration. If you are working with ordinary differential equations, some special functions are available. The function `odeint` will solve a set of ordinary differential equations with a given set of initial conditions.

Last, but not least, let's look at the `weave` sub-package. Even though SciPY already is full-featured, it can't cover every eventuality. Although you always can write the code in pure Python for whatever piece is missing, sometimes you need to squeeze every last cycle out of your hardware. In those cases, you probably want to write some optimized C code to do the heavy lifting. Although you could write this and compile it as an external object file, that is far too much work for any self-respecting programmer. Enter the `weave` sub-package.

With `weave`, you can add C code from within your Python program in a number of ways. The most direct is the inline function. With this, you

can write out your C or C++ code, compile it and run it directly within your Python program. All of your Python objects are available within the scope of your inlined code. The contents of any mutable objects are changeable from within your C/C++ code. If you want to return results to your Python program, these are available in a special variable called `"return_val"`. A trivial example, from the SciPY documentation, uses `printf` to show how the inline function works:

```
import weave
a = 1
weave.inline('printf("%d\\n",a);', ['a'])
```

The general form for the inline function is a string containing the code to compile and run, and a list of the Python variables to make available to the C/C++ code. If you have a larger fragment of code you want to inline, you can use triple quotes to define a code block and save it to a variable first. For example, you may have something like:

```
code = """
    for (int i=0; i<a; i++) {
        printf("%d\\n", i);
    }"""
weave.inline(code, ['a'])
```

Another way to speed up your code is to let Python do it for you with the blitz function. In this case, blitz takes some NumPY expression and creates C++ code and compiles it to an external module. The first time you do this, it may take several minutes to generate the code and compile it. Once this is done, the compiled object file is stored to be reused the next time it is called. Now you can see a speedup of 2–10

over just straight Python code. It is also saved after Python closes, so you can reuse it the next time you run your Python code.

Now you have some tools available to do some real scientific computations. In my next article, I'll look at matplotlib, one of the ways available to visualize all of this computational work you have been doing. Until then, get some science done.—**JOEY BERNARD**

LINUX JOURNAL

now available
for the **iPad** and
iPhone at the
App Store.



linuxjournal.com/ios



For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact John Grogan at +1-713-344-1956 x2 or ads@linuxjournal.com.

AutoSSH, for All Your <CONNECTION LOST>

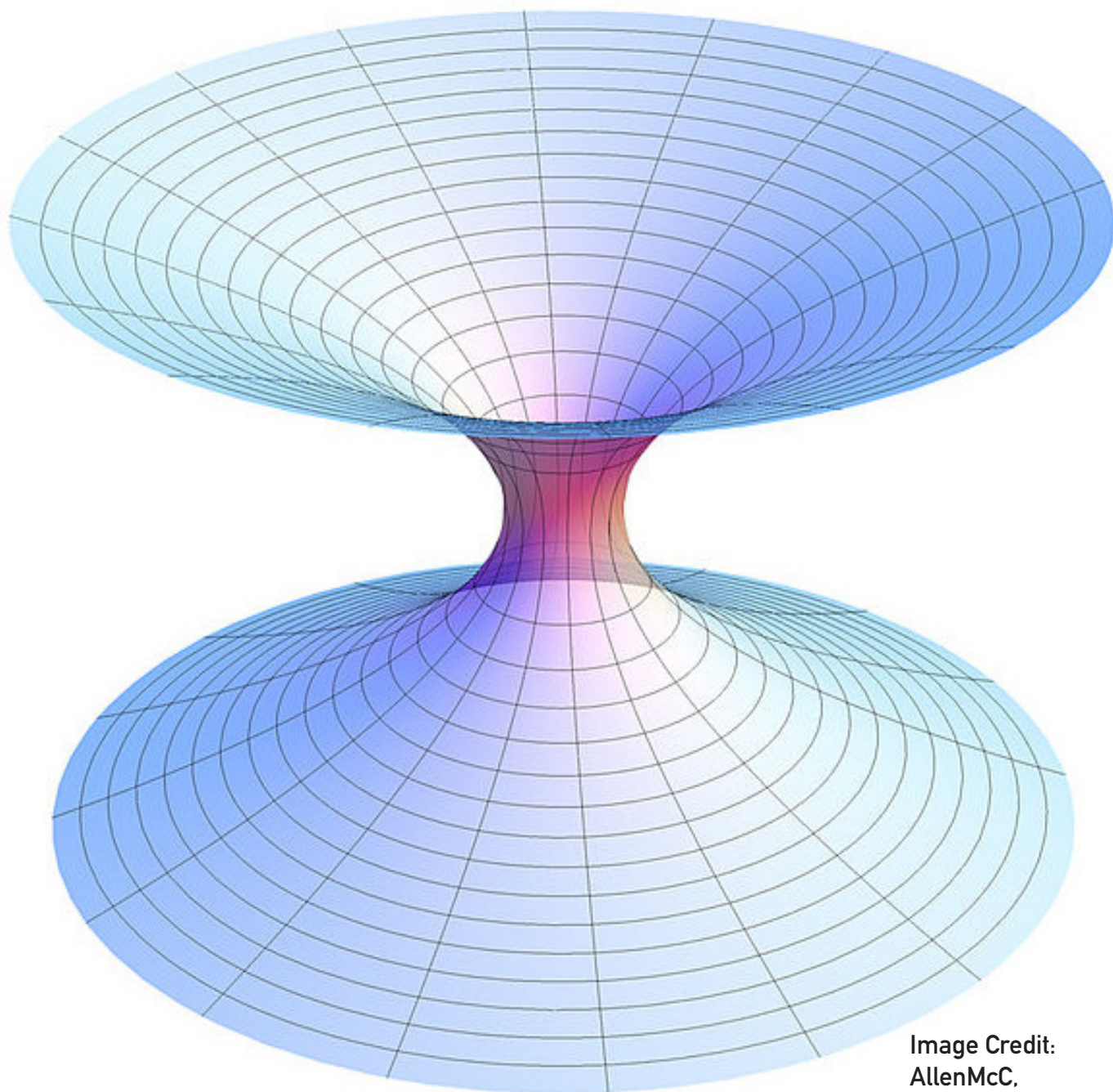


Image Credit:
AllenMcC,
Wikipedia User

I love SSH. I mean, I really, really love SSH. It's by far the most versatile, useful, amazingly powerful tool in my system administration quiver. One of the problems with SSH, however, is that when it dies, it doesn't automatically recover. Don't get me wrong. It's easy to recover with SSH, especially if you've set up public/private keypairs for authentication (I show you how to do that over here: <https://www.youtube.com/watch?v=R65HTJeObkI>). But if the SSH connection dies, it's difficult to reestablish.

In the past, I've done something like enclosing the SSH command in an endless WHILE loop so that if it disconnects, it simply starts over. (I talk about WHILE loops in this month's Open-Source Classroom.) With AutoSSH, however, even if an SSH session is still active, but not actually connected, it will disconnect the zombie session and reconnect a fresh one, without any interaction.

I personally use AutoSSH to keep reverse tunnels active inside a remote data center that is behind a double NAT. Getting into the data center remotely is very difficult, but if I can establish a tunnel from inside the double-NAT'd private network to my local server, getting in and out is a breeze. If that SSH tunnel dies, however, I'm locked out.

In my particular case, the data center is an entire continent away, so driving over isn't an option. With AutoSSH, if something goes wrong, it will keep attempting to reestablish a connection until it succeeds. The program has saved my bacon more than once, and because it's so incredibly useful, AutoSSH takes this month's Editors' Choice award. It's most likely already in your distribution's repositories, but you can check out the Web site at <http://www.harding.motd.ca/autossh>.

—**SHAWN POWERS**

LINUX JOURNAL ARCHIVE DVD



NOW AVAILABLE
www.linuxjournal.com/dvd



REUVEN M.
LERNER

Discourse

Need a forum or examples of modern Web development? Do you want to learn about good application design? Discourse offers all three, in spades.

Back when I started to use the Internet in 1988, there was a simple way to get answers to your technical questions. You would go onto “Netnews”, also known as Usenet, and you would post your question to one of the forums. There were forums, or “newsgroups”, on nearly every possible topic, from programming languages to religions to humor. It was quite amazing, even back then, to be able to get answers to just about any question you had.

But then the Web was invented, and although it took some time, Usenet became a relic of history. Yes, I realize that Usenet still exists to some degree or another, but let’s face it, if you’re looking for answers to questions or for an on-line community, Usenet is unlikely to be your first choice. For better or worse, companies like LinkedIn and Facebook have become large players in this field, even though their group and discussion software leaves much to be desired.

And of course, there are other types of forum software. WordPress, which has become an incredibly powerful and flexible CMS and platform, is blogging software at its heart, meaning that it allows you to post to your blog and accept comments. If you want something a bit more interesting than that, such companies as Disqus and LifeFyre offer SaaS forums, letting you turn any page into (potentially) a social community.

And yet, I probably have implemented forum software five to ten times through the years for various clients, and I have installed forum software about the same number of times. Although forum software often is reasonable, it doesn’t make me go, “wow”. I run an e-mail list for people who live in my city of Modi’in, and it has become quite large, with 2,800 subscribers and 40–50 messages per day; however, we all agree that e-mail is far from a perfect way to communicate. But, e-mail lists still

are far preferable to Web-based forum software, which hasn't ever been exciting or interesting to me.

So, it shouldn't come as a surprise that when Jeff Atwood, cofounder of Stack Exchange and the author of the famous "Coding Horror" blog, announced he was going to make better Web-based forum software, I personally reacted with a shrug. After all, I've used and seen forum software for years. How much different or better could it be?

Well, I can report to you that even though Discourse, Atwood's open-source forum software, is still somewhat unfinished, it already is head and shoulders above other forums I have used, as both a user and an administrator. The fact that it is open source, based on Ruby, PostgreSQL and Ember.js is icing on the cake, making what already was excellent software even better. I've already seen numerous improvements in Discourse since I started to use it, and they just seem to keep coming. For the first time ever, I believe that Web forums can be useful, user-friendly and attractive.

So this month, I look at Discourse—what it is, what it aims to be and what it means regarding not only on-line discussions, but for consumer-facing open-source projects as well.

Installing Discourse

Discourse is an open-source project, hosted at GitHub (see Resources), licensed under the GNU General Public License, version 2. It is backed by Atwood's company, which has the fantastic name of Civilized Discourse Construction Kit, Inc., and it aims to profit through installing and supporting Discourse.

If you're like me, you assume that installing a modern open-source application should be a matter of downloading, installing (perhaps with a separate compilation step first), configuring and then forgetting about it. So, it's a bit of a shock to realize that Discourse is a complex application with many different moving parts that requires some time and attention to install. It's written in Ruby on Rails and Ember.js, and it uses a combination of PostgreSQL (for the main database), Redis and Sidekiq for queuing.

So yes, you can "git clone" the application, and then install the software that way. But, you almost certainly don't want to do that. I did it to get Discourse running on my server, and despite many years of working with these technologies, it still was something of a challenge.

Fortunately, you don't have to do much work. The preferred,

recommended way to install Discourse is to use a Docker container, which means that you're no longer installing an application, but rather an entire, preconfigured virtual machine, with all of the settings in place as the original developers intended. The Discourse instructions describe how to install the software on a Digital Ocean machine, but you can use many different server hosting companies.

Once you have installed Discourse, you need to configure it. And it's when you start to configure Discourse that you see it's a very different animal from most open-source projects, in that it aims to make the life of users extremely easy. The thoughtfulness of the design shines through at every instance. The application needs to be configured by setting a number of Ruby variables within the Discourse application, most of which are documented and described within the configuration file itself.

Once you have configured the application and started it up (and I won't go into details here, because the installation guide describes it well enough), you then will have a working copy of Discourse. Make sure that you have configured the SMTP server correctly, and that you are running the Sidekiq message queue,

so that when you register as a new user, you will receive the e-mailed confirmation request.

Many things in Discourse are designed quite well, starting with the log in system, which allows you to log in via not only a locally defined user name and password, but also via such third-party services as Facebook and GitHub. (You will need to register for an API key for any service you want to use in this way.) Once you have logged in to your newly created account, the real fun begins, setting up the categories into which topics (postings) can be placed. You can do a great deal of customization or very little. Out of the box, Discourse already works quite well.

Configuration of Discourse is done via the administrator menu (using a link in a slightly difficult-to-find location). The configuration menu is huge, letting you use your Web browser to do everything from setting up headers and footers, to entering your Google Analytics token, to setting your API keys for third-party services.

What really blew me away was the fact that when I went to the main administration page, a panel came up telling me what likely was wrong with my installation. For example, it told me I was using a copy of Ruby whose

memory usage was sub-optimal for Discourse. It told me that I had neglected to start up Sidekiq, and it told me how to do that. It even told me that I hadn't changed the logo graphics away from the original defaults.

The design of Discourse, as I mentioned previously, shows that a great deal of attention was spent on small things. When I added a footer to the pages of my copy of Discourse, the in-browser HTML editor automatically added an ending `` tag following the `<a>` tag that I had entered. If I want to update just a single field, rather than the entire set of fields on the update-configuration page, I can do that by clicking on a check box.

It's great to see a Web application that isn't only nice for users, but also nice for administrators. By clicking on a check box, you can back up your database automatically on a regular basis. If you want, you also can enter an Amazon S3 bucket name to which your backup will be delivered. The list of thoughtful features, such as these, goes on and on.

And then, of course, you start to use Discourse for discussions and discover all the little things that they added in order to improve the user experience. For example, you cannot

enter titles or postings that are too short, and Discourse automatically fixes topic titles that are in all caps. If you reference someone with the `@username` syntax, his or her personal interactions menu will glow, indicating that there is something to which he or she should respond. And if the user isn't logged in, or doesn't come to the site within a specified time, he or she will receive e-mail, indicating the reference. This provides a great balance between keeping things on the Web and keeping people involved by e-mailing them and inviting them back.

There are lots of other things, some big and some little, which add to the user experience. One of my favorites, as someone who opens many tabs and sometimes forgets what is in each one, is that Discourse keeps track of my writing even if I use multiple tabs. I don't have to worry that opening the same page in one tab will somehow stomp on the posting I already started in another one. That sort of attention to detail, and an understanding of how people actually use the Web with modern browsers, is quite impressive.

Technical Underpinnings

From a user's perspective, Discourse is piece of software. It is easy for regular users, forum owners and

moderators alike to configure and use it. But Discourse also is a well-written piece of software, designed for maintainability and long-term growth. Looking at the way it was built can help inform your own work, as you try to create increasingly interesting Web applications. That's because Discourse seems to have been built using most of the latest ideas in the Web application world.

For example, although Web applications traditionally did most things on the server, sending complete HTML pages to the user's browser, that is no longer the case. JavaScript is playing an increasingly vital and dominant role in modern Web applications, and Discourse goes all out on this front, using the open-source framework Ember.js, which aims to make it easy to write sophisticated applications in the browser.

Thus, in nearly every case in Discourse, requesting a URL doesn't result in a complete page of HTML being sent to your browser. Rather, you'll often get a page of HTML without any content. A second, separate request, initiated by Ember, then will request the data that should fill in that page. So, a page refresh often will result in an HTML request followed by a JSON request. If you click on a button or link, however, you're

likely not to cause a full page refresh, but rather an Ember action, which will result in an Ajax call to the server, followed by JavaScript-driven changes to the DOM. But as far as the browser is concerned, you have remained on the same page the whole time.

Interviews with Atwood indicate how much he cares about performance, and that is pervasive throughout the site. Although Ruby on Rails has long allowed you to turn an object into JSON with a simple "render json" call, it turns out (and is documented in the Discourse source code) that this is 20% slower than invoking the JSON conversion method on your own. Most applications likely never have noticed this problem, but Discourse, in trying to be speedy, has identified, benchmarked, improved and then documented such issues. In addition, administrators of a Discourse forum automatically have it run with rack-mini-profiler, a Ruby-compatible version of a famous JavaScript plugin that measures the loading speed of the page and of the various Ajax calls executed from within it.

Conclusion

If you need a Web-based forum or are thinking of starting an on-line community, I would argue that you should use Discourse. Despite the



DAVE TAYLOR

Iterating Turns in *Zombie Dice*

You know you're dying to learn how to finish up the *Zombie Dice* game. Dave shows how in his latest shell script programming column.

Back again so soon? Okay, we can make this work. After all, I'm busy trying to avoid gunshots while harvesting as many brains as I can manage—in the *game*, don't get worried. We might work a lot here at *Linux Journal*, but we haven't become the undead as of yet. Then again, we do have a predilection for nighttime activities.

More seriously, I've been building a computer version of the dice-rolling game *Zombie Dice* in my past few articles, and I'm ready to take all the individual components and wrap them up in a turn-based mechanism that lets you actually play the game.

You'll recall that *Zombie Dice* (<http://www.sjgames.com>) has you rolling three dice at a time, getting either brains (good—accumulate 13 and you win), gunshots (bad—get

three and you're dead) and runners (neutral—they force you to roll that same die again next round). Dice are unevenly divided into green, yellow and red, and each has a different ratio of good-to-bad outcome possibilities on the die faces.

Here's where I am at this point:

```
sh ./zdice.sh
    rolled red die: runner
    rolled yellow die: brain
    rolled yellow die: gunshot
1 brains and 1 gunshots.
```

Now, you can see that if I stop, I'm alive and have one brain out of the 13 I need to win. If I continue, I have one gunshot and can survive one more, but if I get shot twice in the next round, I'm dead. The first die is red (the toughest), and it's a runner, so

To model this turn-by-turn gameplay, you need a number of global variables to keep track of your scores per round and in the overall game.

that's not very good.

What I haven't talked about yet is the gameplay. The way *Zombie Dice* works in real life is that it's a two-or-more-player game, and you keep rolling until you either get killed from gunshots, which summarily ends the turn (but you retain any brains you've accumulated from previous turns), get spooked from the gunshots you've had inflicted (in which case you add the brains you've accumulated in this round to your cumulative total) or hit 13 brains, in which case, you win.

Once your turn is over, you pass the dice to the next player. The game continues, turn by turn, until someone gets 13 brains and wins. The trick, of course, is to balance risk aversion (you don't want to lose the brains you've already won) with risk taking (you can handle another gunshot, go for it) in the desire to win before the other player does.

To model this turn-by-turn gameplay, you need a number of global variables to keep track of your scores per round and in the overall game.

Let's start with per round. I code that as follows, starting by showing your per-round accumulated score, then testing to see if you've died from gunshot wounds, and if you haven't, asking if you want to risk another round:

```
show_score
if [ $? -ne 0 ] ; then
    echo "BOOM. You died. But you did get to roll \
        $totalrolls times and eat $brains brains."
    exit 0
fi

/bin/echo -n "You now have $brains brains. Stop here? (y/n) "

read answer
if [ $answer = "y" ] ; then
    echo "You survived. Now it's my turn..."
    exit 0
fi

totalrolls=$(( $totalrolls + 1 ))
```

Most of this is straightforward, but you might be curious about the use of `/bin/echo` rather than just a plain `echo` statement. This is a quirk of Bash and one that I've never really

understood. The version of `echo` that's built in to the Bash shell doesn't know what the `-n` flag means, so if you want to have a prompt that leaves the cursor on the end of the line, here's what ends up happening:

```
-n You now have 2 brains. Stop here? (y/n)
_
```

Use `/bin/echo` instead, however, and it works as you'd hope:

```
You now have 2 brains. Stop here? (y/n) _
```

Yeah, it's a bit daft, but easily addressed with the `/bin/` prefix. Now the code properly accumulates the score, and here's how it plays:

```
$ sh ./zdice.sh

    rolled red die: gunshot
    rolled red die: brain
    rolled yellow die: gunshot

1 brains and 2 gunshots.
You now have 1 brains. Stop here? (y/n) n

    rolled green die: runner
    rolled green die: brain
    rolled green die: runner

2 brains and 2 gunshots.
You now have 2 brains. Stop here? (y/n) n

    dice 1 was a runner, rerolling the same color die again:
    rolled green die: gunshot
    rolled green die: brain

    dice 3 was a runner, rerolling the same color die again:
```

```
    rolled green die: brain
4 brains and 3 gunshots.
BOOM. You died. But you did get to roll 3 times and eat 4 brains.
```

How did I get that to work? Here's the entire main routine, so you can see how everything fits together (the functions I've presented in previous columns if you want to check them out):

```
totalrolls=1
diceroll[1]=0;diceroll[2]=0;diceroll[3]=0
while /usr/bin/true ; do
    for rollcount in 1 2 3
    do
        if [ ${diceroll[$rollcount]} -eq 0 ] ; then
            pick_color
        else
            echo "    dice $rollcount was a runner, \
                rerolling the same color die again:"
            color=${diceroll[$rollcount]}
            diceroll[$rollcount]=0 # reset it for next roll
        fi

        roll_die $color

        echo "    rolled ${colorname[$color]} die: ${nameof[$roll]}"

        add_score

        if [ $roll -eq $RUNNER ] ; then
            diceroll[$rollcount]=$color;
        fi
    done
```




KYLE RANKIN

Tails above the Rest, Part III

Now that you know how to use Tails, let's dig in to some of the more interesting advanced features.

In my first two columns in this series, I gave an overview of Tails, including how to get the distribution securely, and once you have it, how to use some of the basic tools. In this final column, I cover some of the more advanced features of Tails, such as some of its log in options, its suite of encryption tools and the persistent disk.

Superuser and Windows Camouflage

By default, Tails operates with superuser privileges disabled. You don't need superuser privileges to use most of Tails, as those privileges come in handy only if you want to install extra software, modify any local hard drives on the system or do anything else that requires root privileges. Tails disables superuser privileges so an attacker also cannot perform

superuser functions that might threaten the security of your system. That said, if you intend on using Tails routinely as your desktop, you may find you want to install extra software on a persistent disk.

To enable the superuser account, at the initial login window, click the Yes button under More options, and then click the Forward button at the bottom of that window. In the new window, enter the administrator password in the Password and Verify Password text boxes, and then click Login. You also may have noticed a check box in this window to enable Windows Camouflage. This option changes the default desktop theme to look like a default Windows XP install. The idea here is that if you are using Tails in a public place (like on an Internet café, library or hotel

As you might imagine, a security- and anonymity-focused distribution like Tails provides a number of encryption tools.

computer), at a glance, your desktop probably will blend in with the rest.

Encryption Tools

As you might imagine, a security- and anonymity-focused distribution like Tails provides a number of encryption tools. These include more general-purpose tools like GNOME disk manager, which you can use to format new encrypted volumes and the ability to mount encrypted volumes that show up in the Places menu at the top of the desktop. In addition to general-purpose tools, Tails also includes an OpenPGP applet that sits in the notification area (that area of the panel at the top right-hand section of the desktop along with the clock, sound and network applets). The OpenPGP applet has a clipboard icon by default, and you can think of it much like a secured clipboard in the sense that it lets you copy and paste plain text into it and then encrypt or sign it.

The simplest way to encrypt text is via a passphrase, since you don't have to create or import a GPG keypair into your Tails system (made even more

difficult if you don't take advantage of a persistent disk). To encrypt with a passphrase, type the text that you want to encrypt into a local text editor (don't type it into a Web browser window as there is a possibility for JavaScript attacks to access what you type). Select the text, then right-click on the clipboard icon and select Copy. Next, click on the clipboard icon and select Encrypt Clipboard with Passphrase. You will be presented with a passphrase dialog box where you can enter the passphrase you want to use, and once the text is encrypted, the clipboard icon will change to display a lock. This means that your desktop clipboard now contains encrypted text, and you can paste it in any other application, like a Web e-mail application, by right-clicking in that input box and selecting Paste.

If you have copied your GPG keys to this Tails session, you also can use the same tool to encrypt text with your keys. Once you copy the text to the applet, just click on the applet and select Sign/Encrypt Clipboard with Public Keys. You then will be prompted to select the keys of any recipients you

want to be able to decrypt the message. Once you finish with this wizard, you can paste the encrypted text like with the above passphrase option.

You also can use the same applet to decrypt text that has been encrypted with a passphrase. To do this, select the complete encrypted section, including the `-----BEGIN PGP MESSAGE-----` at the beginning and the `-----END PGP MESSAGE-----` at the end. Then, right-click on the OpenPGP applet and select Copy. The icon should change to a lock if the text is encrypted or a red seal if it is only signed. Then, click on the applet and select Decrypt/Verify Clipboard. If the message is encrypted with a passphrase, you should see an Enter passphrase dialog box. Otherwise, if the message used public-key cryptography and you have your keypair on this installation of Tails, you may be prompted for the passphrase to unlock your secret key. If your passphrase or key is able to decrypt the message successfully, you will get a GnuPG results window along with the decrypted text.

Persistent Disk

Tails goes to great lengths to preserve your anonymity by intentionally not persisting any of your data. That said, if you use Tails routinely, you

might find it useful if at least some of your settings stayed around between reboots. In particular, you may want to save account settings in the e-mail or Pidgin clients, or you may want to have your GPG keys persist so you don't have to copy them each session you come across an encrypted e-mail you need to open. Or, you may just have some documents you'd like to work on for more than one session. Whatever the reason, Tails includes a persistent disk option you can use to create an encrypted disk alongside Tails to store this kind of data.

Before you create a persistent volume, there are a few warnings to keep in mind. The first is that Tails goes to great lengths to pick secure programs and to give the programs it installs secure configuration. With persistent volumes, you have the potential to change a configuration or add new browser plugins or packages that may not be as secure or may reveal who you are. When you choose what levels of persistence to enable, it's always best to err on the side of only the features you need. It's also important to note that although the volume is encrypted, no steps are taken to hide that the volume exists. If someone recovers your Tails disk, he or she could see that the persistent volume is there and convince you to

reveal your passphrase.

To create a persistent volume, click Applications→Tails→Configure persistent storage to launch the persistent volume wizard. The persistent volume will be created on the same device you are using for Tails, and the wizard will prompt you for the passphrase to use to encrypt the volume. Once the volume is created, you will need to restart Tails to enable the persistent disk.

Once you reboot, the initial login screen will detect that you have a persistent volume and provide a button labeled “Use persistence?” that you can click to use the persistent volume for this session. You then will be prompted for your passphrase. Once you are at your desktop, the persistent volume will show up as a disk under Places→Home Folder labeled Persistent. You then can drag or save any files to the disk that you want to persist across reboots much like any other directory.

The real power of the persistent volume is in Tails’ ability to store certain configurations or files to it automatically. Click Application→Tails→Configure persistent storage again, and this time, you will see a number of persistent volume features that you can enable:

- Personal Data: allows you to save personal files in a folder that appears under the Places menu.
- GnuPG: persists any GPG keys or settings.
- SSH Client: all of your SSH keys and configuration files.
- Pidgin: Pidgin accounts and settings, including OTR encryption keys.
- Claws Mail: settings for the Claws e-mail program.
- GNOME Keyring: GNOME’s key management software.
- Network Connections: wireless passphrases and other network settings.
- APT Packages: any packages you install on the live system can persist across reboots if you click this option.
- APT Lists: any software repository lists that you download when you perform an apt-get update.
- Browser Bookmarks: pretty self-explanatory.
- Printers: printer configuration settings.

DRUPALCON
AUSTIN

★ ★ ★ ★ ★
2014

JUNE
2-6
AUSTIN2014.DRUPAL.ORG



SHAWN POWERS

Hulk Bash!

Not a programmer? No worries, Bash scripting doesn't have to be rocket science.

I'm not a programmer. Anyone who has read my "code" through the years in my columns would agree. That doesn't mean I don't have a constant need to depend on scripts to help automate my job. Let's face it, system administrators don't have enough arms on their bodies or minutes in a day to accomplish all the various things that need to be done. Any sysadmins worth their salt know enough about scripting to make sure they don't do the same task over and over. If you need to do something more than once, you should be using a script. So in this article, I want to give a quick primer, along with a little bit of insight regarding Bash scripting for Linux.

I'm going to assume you're like me and don't have a programming background. If you're a programmer, you're probably writing your own programs. My paltry scripting abilities won't do you much good, and you'll probably e-mail me about how inefficient I'm being. (Many will

anyway, and that's fine, but be sure to read the next section.)

What Hack-Job Scripting Isn't

There's a place for efficient, well planned programming. There's even a place for well thought-out scripting (Dave Taylor, for example, will give you far more insight on the proper way to script). If you're in the server room as a system administrator, however, sometimes you just need to write a five-line script to automate a task that would take you an hour on your own. If it takes you six hours to write a "proper" script that is designed to save you five hours... well, you fail 6th-grade math.

Error handling is another very important aspect of any programming. Again, however, if you're just making a script to create a JSON configuration file, thinking too much about potential errors is just silly. Look at the config file when you're done, and if it's wrong, fix the script. In my early years as a system administrator, I was afraid to

use a scripting language, because I wasn't a "programmer", and I figured I'd do something wrong. I was right, I *did* do things wrong. I still do. But so do programmers, and there's no better way to learn than by doing. So let's look at some of the basic things a Bash script can do and then experiment. The worst you'll do is mess something up, but then you get to fix it, which is often more fun than breaking it in the first place!

A Few Basic Things I Didn't Know, and a Few I Did

When I first started scripting, I did some really, really dumb things. They worked, so I don't regret them at all, but if I'd known then what I know now, I might have gotten a little more sleep in my 20s. Here's a very partial list of useful things to know when using Bash scripts.

The Backtick On the Bash command line, and therefore in Bash scripts, when you enclose a command in backticks, Bash replaces the stuff inside the backticks with the results. For example, say you have a text file "file.txt" that has a single line of text in it, reading "This is cool." On the command line, if you type:

```
echo "cat file.txt"
```

The command line will return simply:

```
cat file.txt
```

But, if instead you enter:

```
echo "`cat file.txt`"
```

You'll get the following:

```
This is cool.
```

What's happened is that the Bash shell takes the output of `cat file.txt` and uses that in the `echo` statement, because it was encased in backticks. An embarrassing truth is that for a long, long time that's the only way I knew to pass information to a Bash script. If I wrote a script that needed input, I'd save the input into a text file, and then encase `cat thatfile.txt` in backticks. It worked, but I really wish I'd learned earlier about command-line arguments.

Command-Line Arguments If you need user input, Bash scripts allow for this by taking input from the command line when the script is launched. You can reference those variables inside the script, making things much, much simpler. Here's an example snippet:

```
#!/bin/bash
# This is my script, named coolscript
echo "My name is $1, you killed my father, prepare to $2."
```

There are ways to write to a file directly inside Bash, but it's far more convenient to have the Bash file dump its results to the screen, and once you have the script tweaked, redirect the output to a file.

If you launch the script by typing:

```
chmod +x coolscript (NOTE: This makes the script executable,  
it only needs to be done once)  
./coolscript "Inago Montoya" "Die"
```

The script will take your two arguments and substitute them in the script. So the output will be:

```
My name is Inago Montoya, you killed my father, prepare to die.
```

You can have any number of arguments, and the \$1, \$2, \$3 pattern will follow. If your arguments are strings, like my example, note that encasing your arguments in quotes will allow for spaces. Without the quotes, the output of the script would be:

```
My name is Inago, you killed my father, prepare to Montoya.
```

You wouldn't get any errors, but the word "die" would be stored in the \$3 variable and just not used in the script. Command-line arguments

are something I use all the time. It's a great way to get information into the script. If you want an interactive experience, you can use the read command, but I usually just use command-line arguments because it saves time.

Redirecting Output Because system administrator scripting is often a quick hack to solve a problem, redirecting output to a file is fairly common. There are ways to write to a file directly inside Bash, but it's far more convenient to have the Bash file dump its results to the screen, and once you have the script tweaked, redirect the output to a file. The process is simple, but the ability is ridiculously useful. The following command (which could be in a script):

```
echo "This is cool stuff"
```

will immediately respond by displaying "This is cool stuff" on your screen. Generally, you'll have a more complex script that will display many things

on the screen (like a repetitive JSON config file or something), but it still will just print it to the screen. If you want to save the output to a file, you either can copy it and paste it (which I did at first), or you can redirect the output to a file. To do that, change the command to:

```
echo "This is cool stuff" > coolfile.txt
```

You won't get anything printed on the screen, but you also shouldn't get any errors. The cool part is that you will now have a new file called "coolfile.txt", which contains a single line of text. I'm sure you can guess what text that is! One disadvantage of the > redirector is that it writes over whatever file you specify. So if you repeat the command, you'll end up with a brand-new file, named exactly the same thing, with a single line of text. Thankfully, if you use two greater than signs (>>), it will append to the end of the file as opposed to overwriting it. So if you type:

```
echo "This is line one" > oneliner.txt  
echo "This is line two" >> oneliner.txt
```

The file "oneliner.txt" actually will contain two lines of text. Try

playing around with redirecting text. What happens if you try to use a double greater than symbol when a file doesn't exist? Will it error out? Will it create a file? Give it a try and see if you can figure out the way redirection works.

Conditionals: Getting Iffy with It

One of the most common uses for a script is to do some "thing" based on whether or not some other "thing" is true. The construct for accomplishing such a thing is to use an IF/THEN conditional statement. The format works like this:

```
#!/bin/bash  
# An example of an IF/THEN statement  
if [ true ]  
then  
echo "The condition is true!"  
echo "I love true conditions..."  
else  
echo "Uh oh, the condition is false"  
fi
```

A quick walk-through should be clear. If whatever is inside the square brackets evaluates as true, the portion of the script after "then" is executed. If it evaluates as false, the part after the "else" is executed. The else portion of the statement is optional. If the else portion isn't there, the if

Just as useful, if not more useful than a conditional statement in a Bash script, is the loop.

statement just doesn't do anything if the conditional statement is false. It's important to end the entire statement with "fi", which tells Bash that the list of things to do is over. The difficult part is often figuring out what to put inside the [] brackets. The "conditional statement" can get fairly complex, but there are a few common examples:

```
if [ -e /tmp/filename.txt ]
```

This translates to "if the file /tmp/filename.txt exists, then this is true", so the if statement would execute whatever is in the then part of the script.

```
if [ -d /tmp/thing ]
```

This translates to "if /tmp/thing exists, *and* it's a directory, then this is true", so if there is a file at /tmp/thing rather than a folder, the statement will evaluate as false. In that case, the else part of the script will execute, or if there's no else part, the script will just move past the fi statement doing nothing at all. There are a bunch of things that can live in the conditional

brackets. If you want to see a huge list of possibilities, check out http://www.tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_01.html.

Usually, the majority of solutions can be met with creative uses of if/then/else, especially if you nest if/then statements inside other if/then statements. The logic can become fairly complex. There is the case command, which is ideal in some scenarios. Rather than having two options (true/false, if/then), case allows for a list of options. case statements are a little more complex, but just as logical. If you'd like to learn more about case, check out the in-depth guide at http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_07_03.html.

Getting Loopy

Just as useful, if not more useful than a conditional statement in a Bash script, is the loop. I find two types of loops particularly useful: the FOR loop and the WHILE loop. Let's start with the WHILE loop, because it works much like the if/then statement above. Here's an example of a WHILE loop, which I'll dissect next:

```
#!/bin/bash
# A simple WHILE loop
COUNT=0
while [ $COUNT -ne 10 ]
do
    echo "The counter is $COUNT"
    let COUNT=COUNT+1
done
```

There's a few new concepts here, but they're fairly straightforward. First, you set a variable named `COUNT` to zero. Then you set up the while conditional statement. In this case, it's a comparison, comparing the value of the `COUNT` variable to the number 10. The `-ne` means "not equals", so in English, the conditional statement reads, "While the variable named `COUNT` doesn't equal 10, repeat the following." Everything between the `do` and `done` will loop over and over until the conditional statement evaluates as false. As you can probably guess, it's very easy to make an infinite loop with a `WHILE` loop.

Once the loop begins (with the `do` statement), the script echos "The counter is 0", then the variable `COUNT` is incremented by 1, and the loop starts over because `COUNT` is still not equal to 10. Eventually, `COUNT` does equal 10, so the loop stops and moves past

the `done` statement. In the script above, the last thing to print on the screen will be "The counter is 9", because after that is printed, the `COUNT` variable is incremented to 10, and the loop doesn't run again. What would happen if you changed the incrementor to `let COUNT=COUNT+3`? (Answer: the loop would never end and would count by 3 until you got tired and pressed `Ctrl-C` to end the script.)

The conditional statement works just like the `if` conditional statement, and the link above will give you lots of conditional tests to use inside the `[]` brackets. Many are easy to guess, like `-eq` is "equals", `-lt` is "less than" and so on. It's important to know that the conditional statement is evaluated *before* the loop is run, so if the statement starts as false, the stuff between the `do` and `done` never will execute.

The FOR Loop

The last construct I'm going to cover here is the `FOR` loop. It's the hardest loop to wrap your brain around, but it's also one of the most useful. If it seems too complicated or confusing, I urge you to keep playing around with it until it makes sense. Really, `FOR` loops are incredibly useful. Here's a couple simple `FOR` loops that do the

same thing:

```
#!/bin/bash
# Simple FOR loop example
for x in 1 2 3 4 5
do
    echo "Loop number $x"
done
```

```
#!/bin/bash
# Simple FOR loop example with range
for x in {1..5}
do
    echo "Loop number $x"
done
```

Basically, the FOR loops above will print:

```
Loop number 1
Loop number 2
Loop number 3
Loop number 4
Loop number 5
```

What the loop actually does is take the “set” of items from the second half of the FOR statement (so 1 2 3 4 5, or {1..5}) and runs the loop as many times as there are items. Every time the loop runs, it assigns the particular item in the set to the variable in the first part of the FOR statement (so the variable \$x in this case). The examples above make it fairly easy to see what is happening, but it can become really complicated, so understanding

the basics is key. I’m finishing this article with another code snippet. See if you can figure out what it’s going to do, and I’ll go over the results.

First, say you have three text files in a folder /tmp/folder/ by themselves:

- file1.txt: contains the text “This is file 1” on a single line.
- file2.txt: contains “This is file 2” on a single line.
- file3.txt: contains two lines of text, “This is line 1” and “This is line 2”.

Next, create the Bash script that will deal with the files in a FOR loop:

```
#!/bin/bash
# A script that manipulates files with a FOR loop
for x in `ls /tmp/folder/`
do
    echo "I am the file named: $x"
    cat /tmp/folder/$x
    echo " "
done
```

This returns:

```
I am the file named: file1.txt
This is file 1
```

```
I am the file named: file2.txt
This is file 2
```


Moxa's DS-P510A-8PoE Series Switches

Keeping critical devices running in the harshest of environments—and thereby protecting the bottom line—is the role of the Moxa's new EDS-P510A-8PoE Series rugged switches. Targeted at industrial manufacturing and outdoor networks, these Gigabit managed PoE+ Ethernet switches are equipped with eight /100BaseT(X), 802.3af (PoE) and 802.3at (PoE+)-compliant Ethernet ports, and two combo Gigabit Ethernet ports for uplinks for transmitting data up to 120km with high EMI immunity. The devices also feature 3KV LAN surge protection and a wide temperature range (−40°C to +75°C) that ensures the highest reliability of PoE systems while delivering network speeds of up to 1,000Mbps to eliminate bottlenecks. PoE-compliant network devices receive a maximum 30 watts of power per PoE+ port in standard mode, and high power output of up to 36 watts for greater compatibility with power-hungry devices.

<http://www.moxa.com>



Mobile Edge's UrgentPower DX 5200



Today's mobile lifestyle requires power on the go. Purveying the tools to make that lifestyle possible is Mobile Edge, maker of the new UrgentPower DX 5200 mobile power solution for laptops, tablets and smartphones. The UrgentPower DX 5200—offering USB, micro USB, mini USB and an Apple 30-pin connector—is designed to deliver immediate backup power via a high-capacity 5200mAh battery. The battery can provide up to 20 hours of additional talk time for smartphones, up to five hours of additional power for most tablets and more than 500 hours of run time for MP3 players and e-readers. An LED battery-level indicator and LED flashlight also are featured. Built-in safety features include over-voltage/under-voltage protection along with over-current/over-temperature and short-circuit protection.

<http://www.mobileedge.com>



Avago Technologies' 100G QSFP28 SR4 and 100G CFP4 LR4 Optical Transceivers

The latest expansion to Avago Technologies' portfolio of 100G optical transceiver module solutions are the Avago 100G QSFP28 SR4 and 100G CFP4 LR4. The two innovations, designed for modern data-center and enterprise networking applications, cover the full spectrum of link distance requirements for 100G interconnects, says Avago. The 100G QSFP28 SR4 solution, designed for 100GbE short-range data-center interconnects, is compliant with IEEE 802.3bm 100GBASE-SR4 and CAUI-4 specifications and enables seamless transition from 40G to 100G using the QSFP form factor. Meanwhile, the 100G CFP4 LR4, designed for 100GbE long-range data communications, is compliant with IEEE 802.3 Clause 88 for 100GBASE-LR4 media, Clause 83E for CAUI-4 electrical interface, OTN OTU4 and the CFP4 MSA for MDIO functionality. The solution leverages best-in-class CyOptics single-mode laser technology. <http://www.avagotech.com>

2ndQuadrant's Barman

Professional PostgreSQL specialist

2ndQuadrant is calling the updated Barman 1.3 PostgreSQL Backup and Recovery Manager

"more robust" and "exciting". Barman,

an open-source administration tool, allows

businesses of any size to perform continuous remote backups of multiple servers, providing significant time and cost savings in the event of a crash. A key feature is the network compression option, which allows for a reduction in the amount of traffic between the PostgreSQL server and the backup server. For instance, when two servers are in different data centers, variable costs from network traffic can be reduced. Barman 1.3 features a new code infrastructure in terms of output, sub-processes, remote commands, filesystem, events ("hooks") and metadata management. In addition, administrators can now force a rebuild of the xlog.db file (WAL archive).

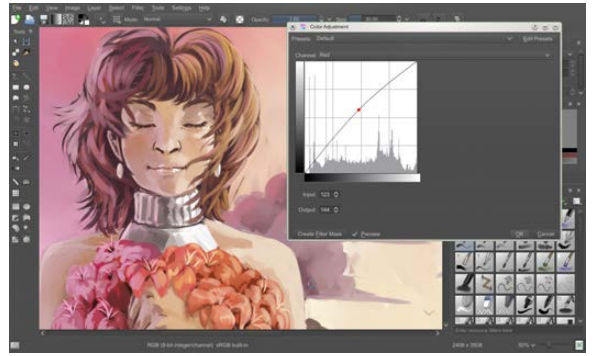
<http://www.2ndquadrant.com> and <http://www.pgbarman.org>



Barman
Backup and recovery
manager for PostgreSQL

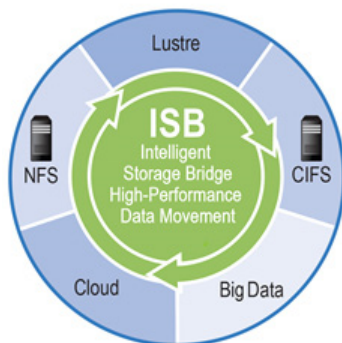
Krita

Although Krita is free, a graphic artist recently described the open-source digital painting studio as “a production beast”. Recently updated to version 2.8, Krita is an advanced digital painting application, used by illustrators, comic-book artists, concept artists, matte painters and in VFX and game studios worldwide. Although version 2.8 offers significant advancements on both Linux and Windows, the release marks a major milestone on the Windows side: the platform is henceforth fully supported. Of the many new features and improvements, highlights include a drastically improved drawing experience thanks to a complete rewrite of the graphics tablet support and an improved OpenGL canvas. The new High Quality Filtering option uses shaders instead of hardware to perform display zooming, giving clean, clear, crisp and accurate line art, also at lower zoom levels. Krita is developed by the Krita team and KO GmbH.



<http://www.krita.org> and <http://www.kritastudio.com>

Terascale's Intelligent Storage Bridge



The problem that Terascale's Intelligent Storage Bridge (ISB) tackles is the vast amounts of unstructured “stranded data” found in today's HPC infrastructures. These infrastructures are typically not optimized for efficient and daily movement of data, which strictly limits valuable analysis opportunities. Terascale says that ISB is the first solution of its kind to solve this “stranded data” challenge by automating the transfer of large data sets between high-performance fast scratch storage and a

cost-effective enterprise storage system. The latest version of the ISB now includes vendor-agnostic support of Lustre solutions, allowing organizations to bring together a wider range of HPC and enterprise storage solutions. Other new aspects of the ISB include programming interfaces that enable integration and management by third-party orchestration solutions or home-grown scripts and applications, as well as passive CIFS and NFS gateway capabilities, which allow users of workstation technologies lacking Lustre support to access and update fast scratch data through native filesystem technologies.

<http://www.terascal.com>



Open-E Data Storage Software

Open-E hews to the philosophy that you should be saving big money by leveraging commodity servers for centralized data storage in high availability, big data, cloud storage, virtualization and business continuity environments. This philosophy is alive and well in the updated Open-E Data Storage Software (DSS) V7, a Linux-based enterprise storage application used for building and managing centralized data storage servers—NAS and SAN. Open-E claims that the Open-E DSS V7 can significantly increase the performance of replicated environments, ensuring the safety of customers' businesses in case of any disaster, and brings superior performance, extensibility and reliability. The company adds that customers can achieve superior performance of up to 10X in random writes of mirrored volumes, allowing data to be copied in real time and every change immediately mirrored from the primary server to the secondary storage server. Storage system performance also will be evident in environments with multicore processors and can be optimized further with integrated and improved hardware performance-tuning parameters.

<http://www.open-e.com>

Jordan Schwartz's *The Art of LEGO Design* (No Starch Press)

No Starch Press' tagline could not be more directly targeted at us: "the finest in geek entertainment". What geek would not revel in one of the publisher's latest titles, Jordan Schwartz's *The Art of LEGO Design*, a book that seeks to inspire the inner-geek artist in each of us. In the book, former LEGO Group designer Jordan Schwartz explores LEGO as an artistic medium, revealing rarely known and creative ways to build impressive models with the system. From effective composition to intricate texture design, Schwartz shares a variety of creative insights into crafting both realistic and stylized models. Readers can learn tricks like how to turn rubber pieces into an octopus, use light bricks to build a roaring LEGO fire and create an owl face out of minifigure capes. Full color images of inspiring models stand alongside interviews with talented builders, empowering readers to build their own, custom LEGO masterpieces.



<http://www.nostarch.com>

Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o Linux Journal, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

Hacking the Parrot A.R. Drone



**Hack a toy into your own flying
robot overlord with the A.R. Drone!**

BILL CHILDERS

Drones. They're all over the news lately, and it seems that you can't get away from them. From their use in Afghanistan and other far-away places, to monitoring farmers and crops domestically, they're everywhere. Their robotic nature and all-seeing camera eyes lead one toward visions of a Skynet-like army of mechanical flying hunter-killers chasing after humans and

terminating them one by one—or maybe that's just me, and I've watched too many science-fiction movies.

However, there's a less death-inducing version of a drone that's available to us, the non-military nerds, and it's available today. It's the Parrot A.R. Drone, a semi-autonomous, largely automated quadcopter. I'd call it a toy, but it's as close to a toy as a



Figure 1. The A.R. Drone, Flying

smartphone or a tablet. This “toy” has some pretty wild potential, and it’s masked and presented as a cute, fun plaything.

So What Is an A.R Drone, Anyway?

The A.R. Drone is, at its heart, a \$299 radio-controlled quadcopter. Its four rotors provide stability and ease of flight without the pilot having to worry about offsetting the torque of a single rotor or dealing with the complexities of flying a conventional tail-rotor helicopter design. Although there

are a few toys that do this on the market today, the A.R. Drone justifies its cost by taking this a little further—it is controlled by a smartphone (or tablet), and uses an onboard computer running Linux to do a heavy amount of in-flight stabilization and computation to make the drone simple to fly for even the most novice of radio-control pilots.

The A.R. Drone has a bunch of onboard instrumentation. There’s two cameras, one in the nose and one in the belly, and the



Figure 2. AR Drone Screenshot

Table 1. A.R. Drone Hardware Breakdown

HARDWARE	VERSION 1.0	VERSION 2.0
CPU	500MHz ARM	800MHz ARM
RAM	128MB	1GB
Front Cam	VGA	720p
Ground Cam	QVGA	QVGA
USB	Diag only	Onboard, can hook a Flash drive to it
Wi-Fi	802.11 b/g	802.11 a/b/g/n
Ultrasonic Altimeter	Yes	Yes
Pressure Sensor	N/A	Yes

cameras feed video back to the smartphone app for first-person perspective while flying. It's also got an onboard ultrasonic altimeter and an accelerometer to measure tilt and acceleration. Using these instruments in concert with the smartphone app allows the A.R. Drone to do things for you that are usually really hard, such as station-keeping while hovering with the wind blowing. The pilot doesn't have to compensate for the wind, as the A.R. Drone does all the hard work. The smartphone app has very intuitive controls, as well. To take off, you simply tap the "Take Off" button at the bottom of the screen, and the drone will lift off and hover in front of you, approximately two feet off the ground. From there, you just tap and hold on the left side of the screen to engage the accelerometer controls, and tilt your smartphone in the direction you want the drone to go. On the

right side of the screen, you just tap and hold, then slide up and down to control altitude, or slide left and right to spin the drone around. If you lift both fingers from the phone's screen, the drone will stop all maneuvers and station-keep, hovering in place.

Onboard, the A.R. Drone really is a flying computer, coupled to a Wi-Fi access point. There also are two versions of the A.R. Drone: version 1.0 and 2.0. I was able to pick up the 1.0 version for \$100 on-line as a refurbished item, but if you can afford it, the 2.0 version brings much-needed improvements, such as a 720p video camera in the nose and a faster CPU. Table 1 shows the hardware breakdown.

As you can see, there's a definite advantage to the A.R. Drone 2.0, but the 1.0 is still a capable unit. All my investigations apply to the 1.0, but should apply to the 2.0 version as well.

Enough of This, Let's Get Hacking!

Since the smartphone app relies on the fact that you pair your phone or tablet with the access point inside the A.R. Drone, I figured that'd be the ideal place to start poking around at the little guy. I pointed my laptop at the access point that the drone broadcasts (ardrone_27515 in my case) and got an IP address on the "network". I then pointed the "nmap" tool at the IP address of the drone, which was 192.168.1.1:

```
bash-3.2$ nmap -O 192.168.1.1

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-07 10:39 PST
Nmap scan report for 192.168.1.1
Host is up (0.0019s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
MAC Address: 00:26:7E:62:03:8A (Parrot SA)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

OS detection performed. Please report any incorrect
  -> results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.06 seconds
```

Well, look at that. The A.R. Drone

has an FTP server and a Telnet server, with both running and open. That Telnet server is just too inviting of a target, so let's just give that a try:

```
bash-3.2$ telnet 192.168.1.1
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^J'.

BusyBox v1.14.0 (2012-08-20 14:37:54 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

```
#
```

That's nifty. It looks like the A.R. Drone's Telnet server drops you right to a root shell running BusyBox, without any kind of authentication at all. It's not the best for security, but it's great for just hacking around.

Let's run a couple commands and see what's going on on this little guy. First, I'll check the CPU and see if it is what I think it is (a little ARM processor):

```
# cat /proc/cpuinfo
Processor       : ARM926EJ-S rev 5 (v5l)
BogoMIPS       : 233.47
Features        : swp half thumb fastmult edsp java
CPU implementer : 0x41
CPU architecture: 5TEJ
```

```

CPU variant      : 0x0
CPU part        : 0x926
CPU revision    : 5
Cache type     : write-back
Cache clean    : cp15 c7 ops
Cache lockdown  : format C
Cache format   : Harvard
I size         : 32768
I assoc        : 4
I line length  : 32
I sets         : 256
D size         : 16384
D assoc        : 4
D line length  : 32
D sets         : 128

```

```

Hardware        : Mykonos Parrot platform
Revision        : 0904
Serial          : 000000000000000000

```

There's a little ARM CPU there. Let's check RAM and see how much the A.R. Drone has available once it's all running and has its services alive:

```

# free

```

	total	used	free	shared	buffers
Mem:	126068	23644	102424	0	0
Swap:	0	0	0		
Total:	126068	23644	102424		

It looks like the A.R. Drone has a very large portion of its memory available (102MB out of 128MB).

That might be handy if you ever decide to hack the drone further and run your own software on the drone. For now though, let's just continue with investigating the drone itself, and try to get a computer to control the drone instead of the smartphone application. Here are some more basics about the A.R. Drone's running Linux distribution:

```

# uname -a
Linux myhost 2.6.27.47-parrot #1 PREEMPT Mon Aug 20
14:46:29 CEST 2012 armv5tejl GNU/Linux

```

```

# df -h

```

Filesystem	Size	Used	Available	Use%	Mounted on
ubi1:system	12.0M	7.0M	4.4M	61%	/
tmp	61.6M	32.0K	61.5M	0%	/tmp
dev	61.6M	0	61.6M	0%	/dev
ubi0:factory	4.8M	44.0K	4.5M	1%	/factory
ubi2:update	13.2M	160.0K	12.3M	1%	/update
ubi2:data	67.5M	1.3M	62.7M	2%	/data

There's a lot more stuff to investigate on the A.R. Drone's filesystem and operating system, but in the interest of time, I'll cut to the chase. All the really interesting bits of the software—the stuff that makes it fly—is in /bin, and there are a few processes listening on weird ports like 5554, 5555 and 5556. It's also running a DHCP

server (the udhcpd daemon), so that the smartphone app can get an IP address once the phone or tablet associates with the Wi-Fi network that the drone creates.

It's the processes running on those ports 5554, 5555 and 5556 that I'm going to continue to explore. If you can feed the right kind of data to the A.R. Drone on these ports, you can replace the

smartphone app with your own program, running on a computer. That'd give you the ability to begin to make the drone truly autonomous. Exploring how to make the drone fly is beyond the scope of this article, as it'd involve reverse-engineering the A.R. Drone protocol and writing your own software. However, there's a lot of smart people out there on the

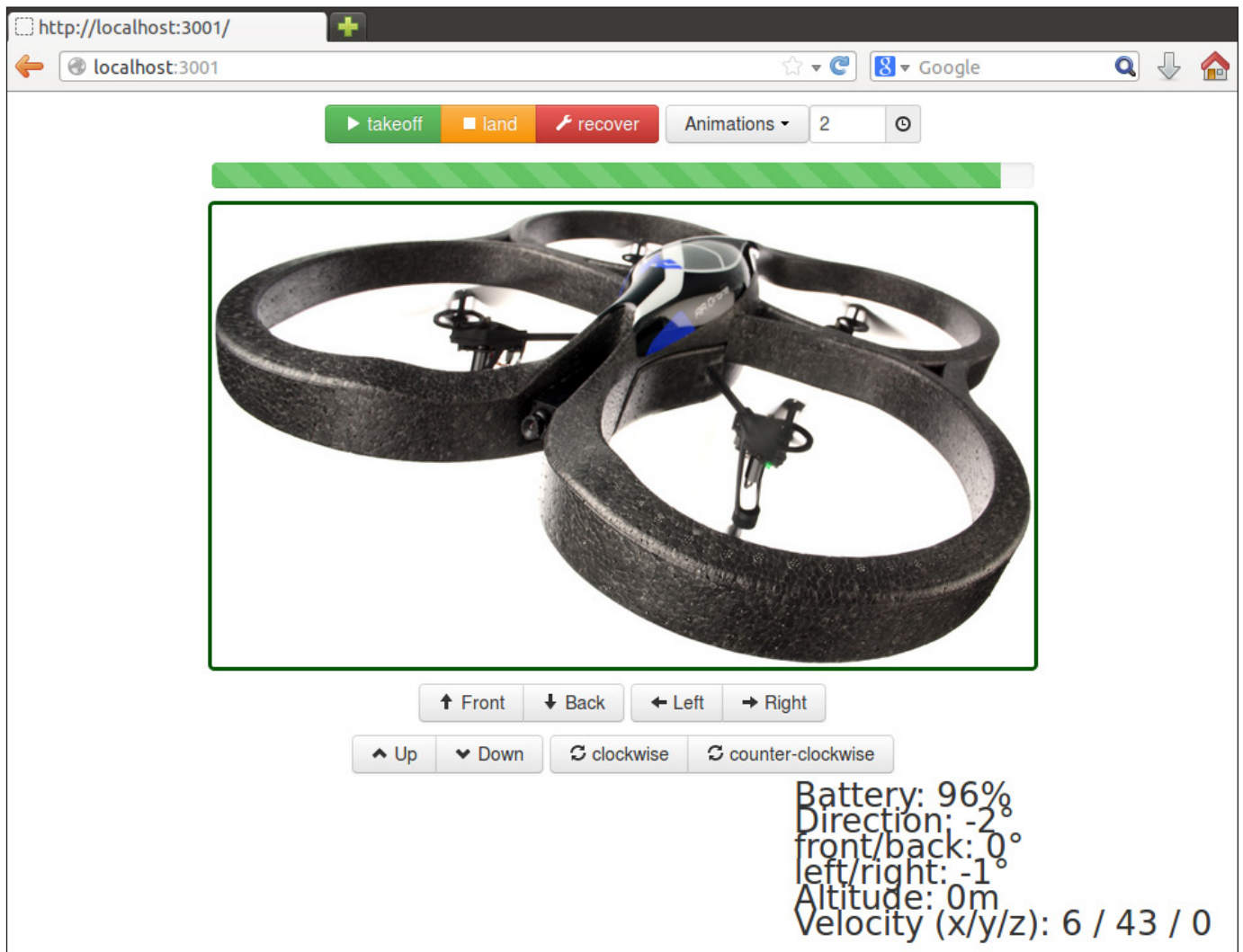


Figure 3. A Screenshot of drone-browser

Internet, and they've done that for you. Let's start off by trying out "drone-browser" (see Resources), a rudimentary Web-based UI for the A.R. Drone.

Using Open-Source software with the A.R. Drone

drone-browser is written in node.js, as is a lot of the other software available for the A.R. Drone. To get it going, you need a node interpreter and a copy of the software from the project's git repository. See the sidebar "Getting drone-browser Running" for more information. Once you have drone-browser going, you'll get a browser-based GUI, as shown in Figure 3.

Once you have drone-browser

running, you can control the A.R. Drone via keyboard commands or button-presses in the Web browser window. Be warned, however, that the controls are not nearly as refined as the smartphone application, and there is no "auto-stop" command. When you command the drone to fly forward, it will—until it receives a stop command or runs into an obstacle. Don't do what I did and attempt this in the house—I nearly ran the drone into my wife as she sat at her desk.

So, drone-browser is a good first step to start playing with computer control, but it's not a way to control your A.R. Drone programmatically. In order to enable

Getting drone-browser Running

Getting drone-browser fired up on your computer is pretty quick, all things considered. These instructions are for Ubuntu, but other distributions are similar.

1) Install the node.js interpreter:

```
sudo apt-get install node
```

2) Clone the project's git repository:

```
git clone https://github.com/functino/drone-browser.git
```

3) Associate your computer to the drone's Wi-Fi network:

4) Run the code:

```
node ./server.js
```

5) Connect your browser to the node server by pointing it to `http://localhost:3001`.

That's it!

full Terminator-style hunter-killer mode, you'll need another piece of software—something that will let you write a program on the computer and translate it into the instructions the drone can accept. This exists already, and it's called `node-ar-drone` (see Resources for more information).

`node-ar-drone` lets you write relatively straightforward JavaScript code that can be interpreted into instructions to fly the drone. Installing `node-ar-drone` is straightforward, as described in the sidebar. As per `node-ar-drone`'s documentation, the fastest way to control the drone is with an interactive program. Begin by creating a file called `repl.js` that contains the following instructions (example borrowed from the

project documentation):

```
var arDrone = require('ar-drone');
var client = arDrone.createClient();
client.createRepl();
```

Then, just run your code, and issue the commands at the command line:

```
node ./repl.js
// Make the drone takeoff
drone> takeoff()
true
// Wait for the drone to takeoff
drone> clockwise(0.5)
0.5
// Let the drone spin for a while
drone> land()
true
// Wait for the drone to land
```

node-ar-drone

The `node-ar-drone` software is exactly what you need to get the drone under full computer control.

1) Install the `node.js` interpreter (if you haven't already):

```
sudo apt-get install node
```

2) Install the `node-ar-drone` libraries:

```
npm install git://github.com/felixge/node-ar-drone.git
```

3) Write your `node.js` program to control the drone.

4) Associate your computer to the drone's Wi-Fi network.

5) Run your program:

```
node ./myprogram.js
```

Then stand back as the drone starts running through the paces you told it to!

Cross-Breeding the BeagleBone Black with the ATmega328p

Build your own ATmega328p programmer with the BeagleBone Black. This tutorial starts from raw components and shows you how to assemble the hardware, manipulate the kernel's device tree files and write the software for a complete soup-to-nuts project.

JOSHUA DATKO

The Raspberry Pi is the inexpensive, embedded Linux computer that comes to mind for most *Linux Journal* readers. Last year, the Pi swept three Readers' Choice Awards: Linux Product of the Year, Best Other Linux-Based Gadget and Best New Open-Source Project. However, I was in the proud 3.3% of readers who voted for the BeagleBone. Although the Pi may have the popularity, the BeagleBone Black (BBB) has a faster processor, 2GB of onboard eMMC storage and more than 65 different Input/Output (I/O) pins. These features make the BBB my embedded development platform of choice. However, for some truly low-level projects, it is more appropriate to use an embedded microprocessor, such as that provided by the familiar Arduino platform.

In this project, I describe how to combine the ATmega328p, the microprocessor on the Arduino UNO, with the BBB. Specifically, I detail the steps needed to use the BBB as a programmer for the ATmega328p so that you can upload sketches built with the Arduino IDE directly to the chip!

There are a few open-hardware projects that incorporate this technique. For example, one of the most successful BBB projects, OpenROV, uses a similar

configuration. OpenROV is an open-source (software and hardware) underwater exploration robot. On the robot, the ATmega328p controls the servos for movement. There is another BBB cape that uses an ATmega328p: the CryptoCape. The CryptoCape is a collaborative project between SparkFun Electronics and Cryptotronix, my open-hardware company. The cape also contains various other security Integrated Circuits (ICs), like a Trusted Platform Module (TPM), an encrypted Electrically Erasable Programmable Read-Only Memory (EEPROM), a Real-Time Clock with an attached holder for a coin-cell battery and ICs that perform the Elliptic Curve Digital Signal Algorithm (ECDSA). I've included an ATmega328p on the CryptoCape for users to upload their own crypto-library to the microcontroller or to use the microcontroller to interface with other hardware.

The soon-to-be-released Arduino TRE combines the Atmel ATmega32u4 with the Texas Instrument Sitara ARM Cortex-A8. The Arduino TRE is a collaborative effort between BeagleBoard.org and Arduino, and early looks indicate that it has four USB ports, HDMI, audio In/Out, Ethernet and power options for a 5V jack and USB power. Until the TRE is

available, you can make your own BBB and ATmega cross-breed using the approach outlined here.

In this configuration, think of the BBB as a front end for the ATmega328p. Imagine a configuration where the ATmega328p is controlling an electromechanical device, but the firmware needs to be updated. With an attached BBB, the BBB can run a Web server, accept the new firmware and update the ATmega328p without a programming cable.

For completeness, there is another

software support is not as mature as the Arduino project, and for custom applications, one needs to program in assembly for the PRU instruction set. However, if execution speed is critical for your application, it may be worth your time to learn about the PRU.

When combining the ATmega328p and the BBB, you must consider the operating voltages of both devices. The Arduino UNO, which is built around the Atmel ATmega328p, operates at 5V, while the expansion headers on the BBB operate at

In this configuration, think of the BBB as a front end for the ATmega328p.

option for those looking for a low-level hardware interface on the BBB. The BBB contains two independent Programmable Real-Time Units (PRUs). The two PRUs on the BBB run at 200MHz and have shared memory with the main processor. The PRUs run at much higher speeds than AVRs and are better suited for high-speed applications like real-time motor control or video encoding/decoding. Although there is an example library provided by BeagleBoard.org, the PRUs are more complicated and have a slightly higher learning curve. The

3.3V. To combine the BBB and the ATmega328p, you must decide whether to operate all components at 3.3V or use logic-level converters and supply power to the ATmega328p at 5V. If operating at 3.3V, you also must reduce the crystal frequency for the ATmega328p from 16MHz to 8MHz. For this tutorial, I've chosen to keep all voltages at 3.3V, because the circuitry is simpler, and I can afford the reduced processor speed.

This tutorial has three parts. First, I describe the hardware components and skills required to

build the prototype. Second, I detail the software required to power, program and communicate with the microprocessor. And finally, I show how to upload sketches onto the ATmega328p.

The Hardware

This project does require soldering. However, I have chosen to use only through-hole components, which generally are easier to solder. If you are new to soldering, the SparkFun Web site has some great tutorials (see Resources).

For this project, you need the following components:

- Soldering equipment (iron, solder).
- (1) ATmega328p in DIP package.
- (2) 22 pF capacitors (for the crystal).
- (2) .1 uF capacitors (for decoupling).
- (1) 8MHz crystal.
- Hookup wire.
- (1) 6-pin shrouded connector.
- (1) Protoboard.
- Male breakaway 0.1 inch pins.

- AVR Programmer (AVR Pocket Programmer).
- (Optional) AVR breakout board.
- (1) LED.

I've made a public SparkFun wish list that lists all of these components (see Resources).

There are two ways to build the hardware for this project: the easy way is to use a solderless breadboard to assemble the components, and the harder way is to build a prototype cape. I generally try to get projects working on a solderless breadboard first, before I start soldering components onto a protoboard, but sometimes a breadboard introduces challenges with poor connections. Refer to the Fritzing diagram for details of the breadboard

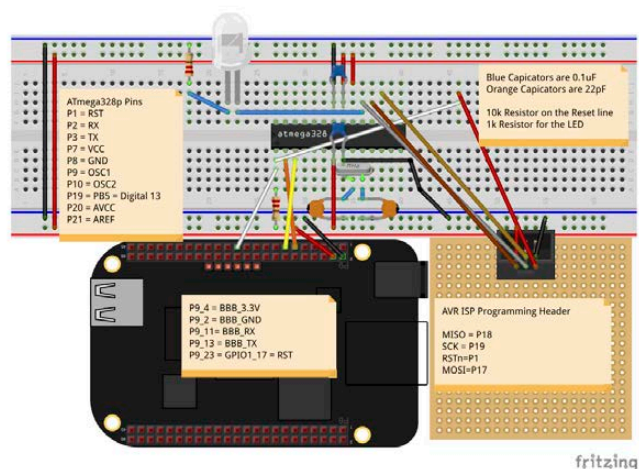


Figure 1. Schematic for the ATmega328p-BeagleBone Combination

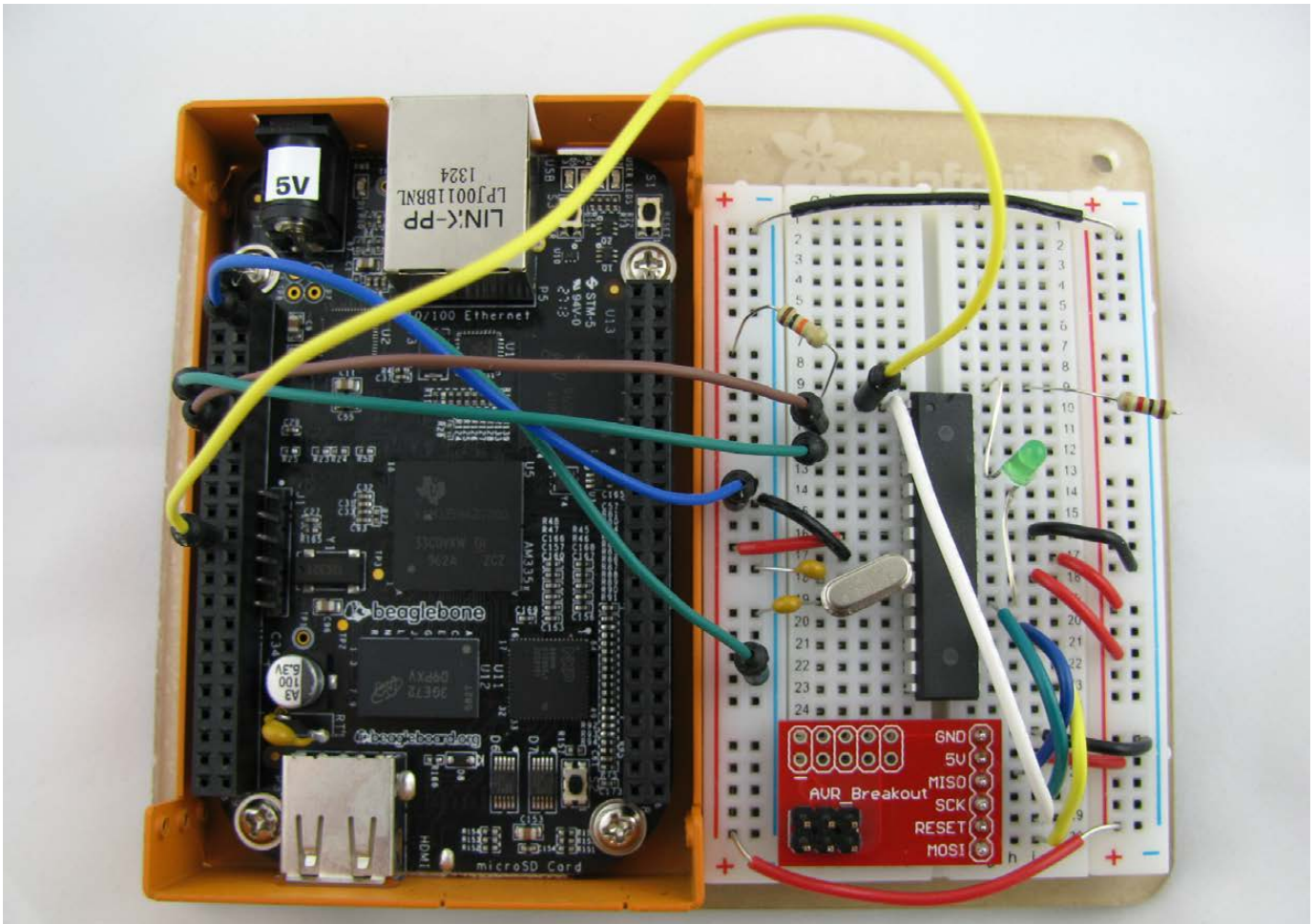


Figure 2. BBB with an ATmega328p on a Solderless Breadboard (photo by Josh Datko)

implementation (Figure 1).

If your ATmega328p already has a bootloader, you can skip the ISP programming header, but it's very handy to flash a bootloader quickly or sketch on the microprocessor if there is a problem. However, you should not connect the ISP programmer to the board while connected to the BeagleBone. The programmer is most likely using 5V, and the VCC line is tied to the main power supply from the BBB, which is 3.3V. You could add

logic-level converters to make this safer, but it adds some complexity.

The benefit of using a protoboard is that you will have a much more stable prototype, both electrically and mechanically. However, it requires more of a time investment and significantly more soldering. The protoboard also requires some 0.1" male headers to attach into the BBB's expansion headers. You could get away with soldering just the pins that are used in this project, but I

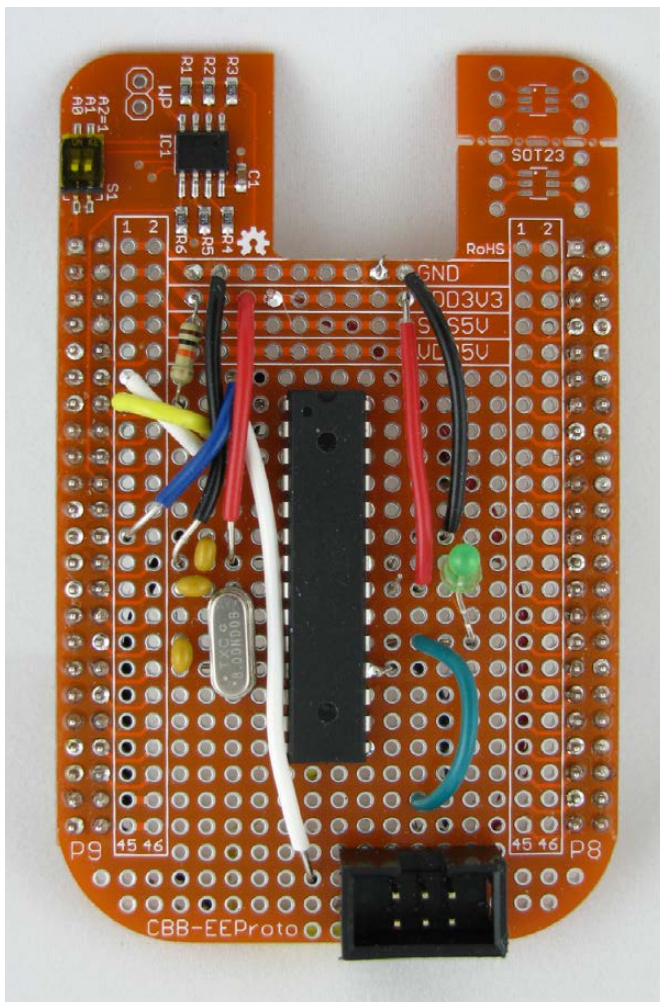


Figure 3. ATmega328p on the Logic Supply BeagleBone Proto Cape with EEPROM— Front (photo by Josh Datko)

recommend attaching the entire 2x23 male headers.

The Software

Flashing a Bootloader There are several ways to load a bootloader, including purchasing an ATmega328p with a pre-installed bootloader. The method I chose was to use the Arduino IDE and the AVR Pocket

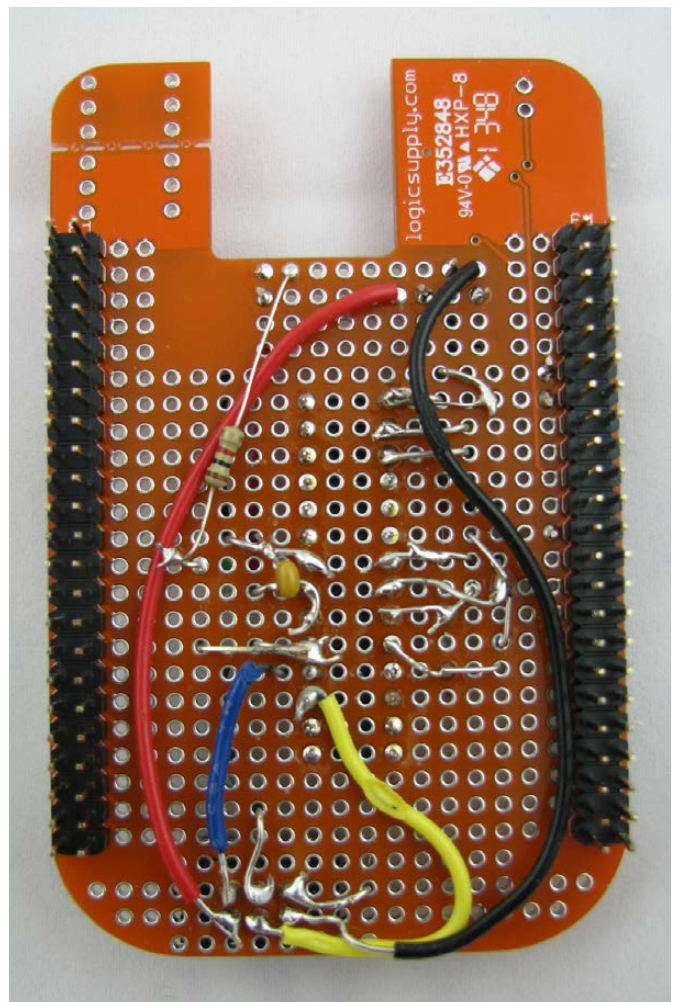


Figure 4. ATmega328p on the Logic Supply BeagleBone Proto Cape with EEPROM— Back (photo by Josh Datko)

Programmer. To load the bootloader, connect the six-pin ISP cable to your ISP header and the USB end to your host computer. Select "Power Target" on the Pocket Programmer. In the Arduino IDE, select a board type of "Arduino Pro or Pro Mini (3.3V, 8MHz) w/ ATmega328". Under Programmer, select "USBtinyISP". Then select "Burn Bootloader". After a few minutes, you

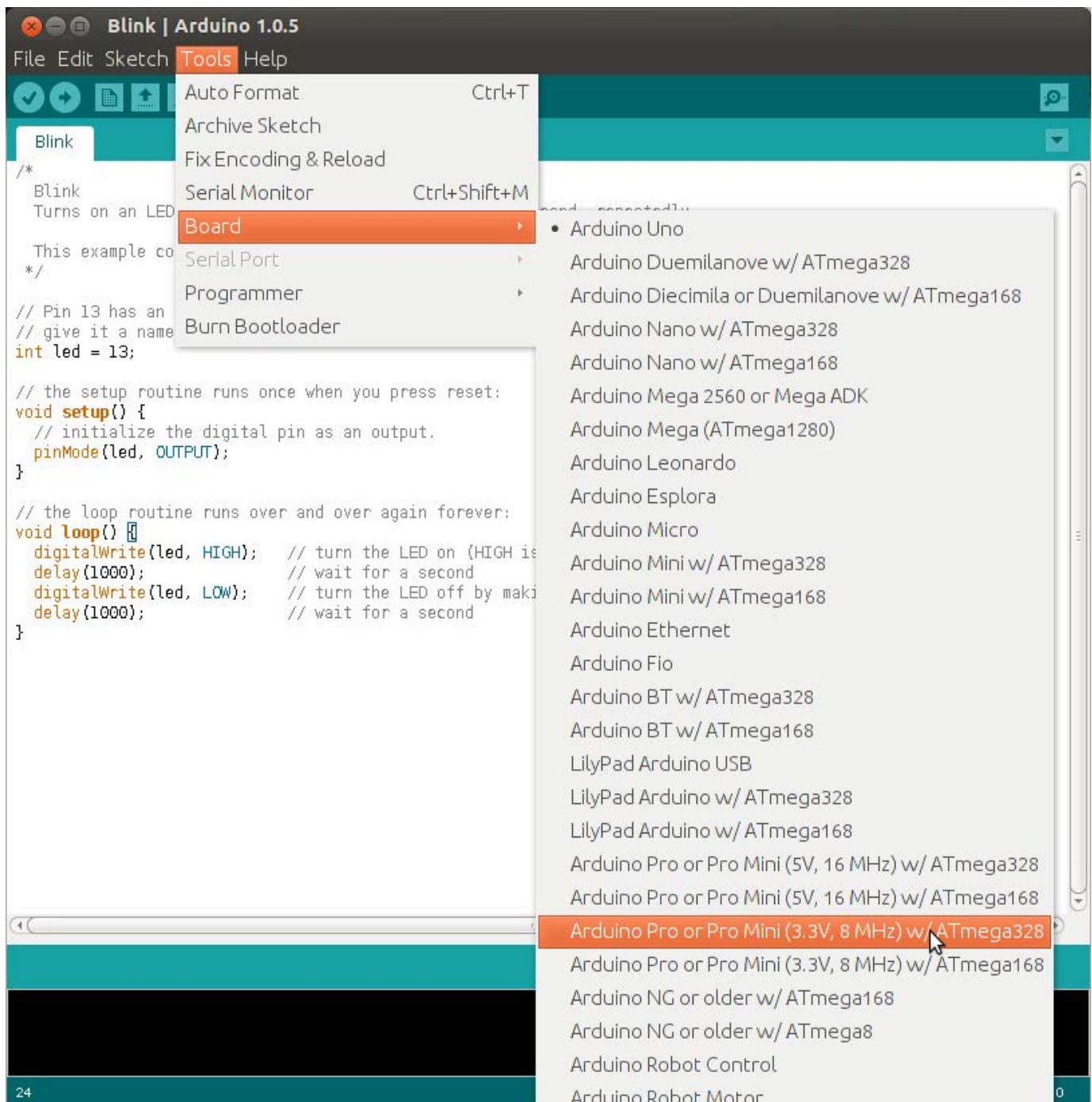


Figure 5. Select Arduino Pro or Pro Mini (3.3V, 8MHz) with ATmega328.

should be rewarded with the message “Done burning Bootloader”. With a bootloader installed, disconnect the programmer and attach the cape to

the BBB.

Preparing the BBB There are three wires required for the software reset and sketch upload feature. Two are

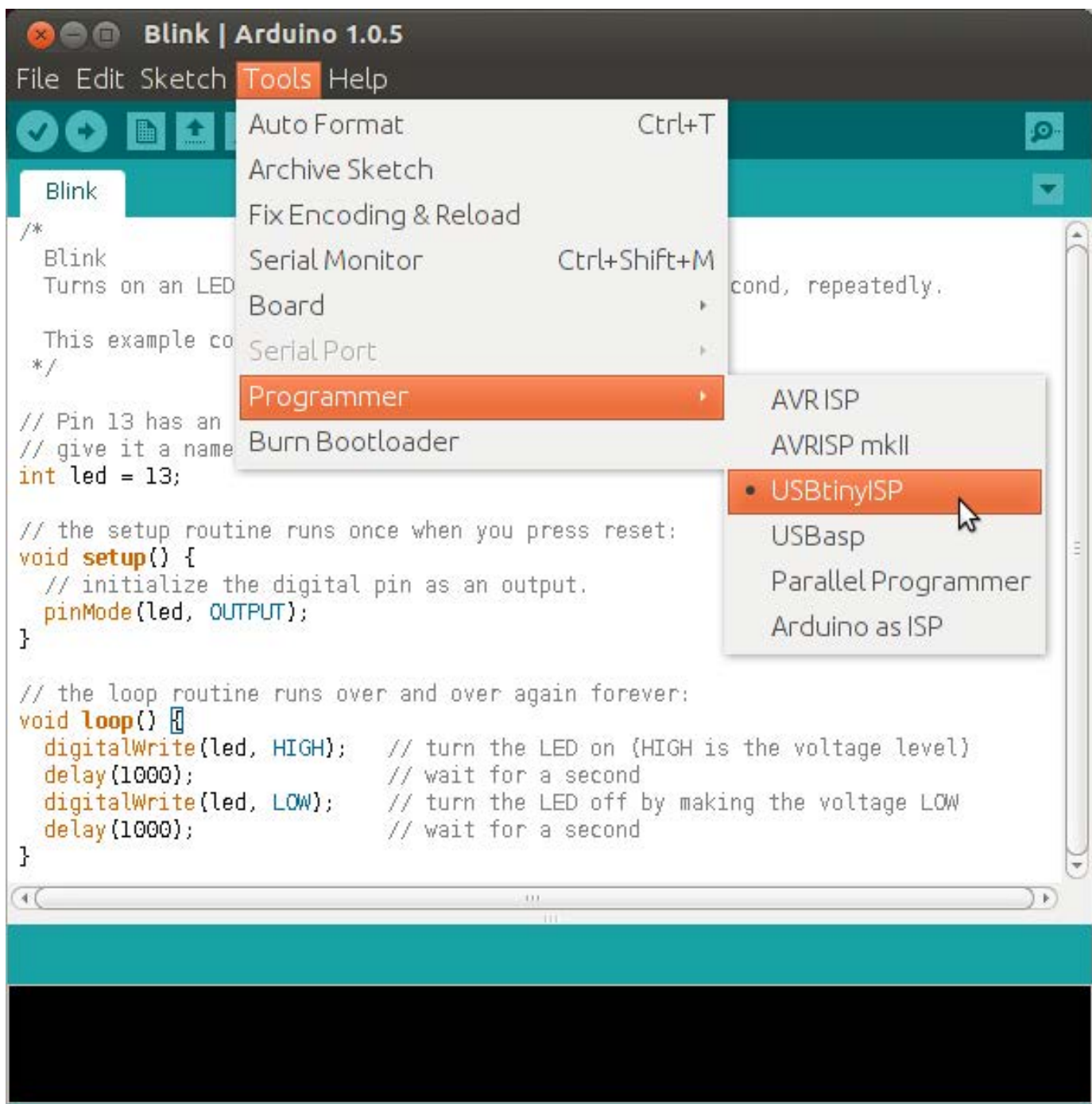


Figure 6. Select USBtinyISP.

for serial Transistor-Transistor Logic (TTL) data transmit (TX) and receive (RX), and the third line is a General Purpose Input Output (GPIO) pin,

used to toggle the reset line. You can manipulate the pins through the exported sysfs, but first you must determine the pins to use. For the

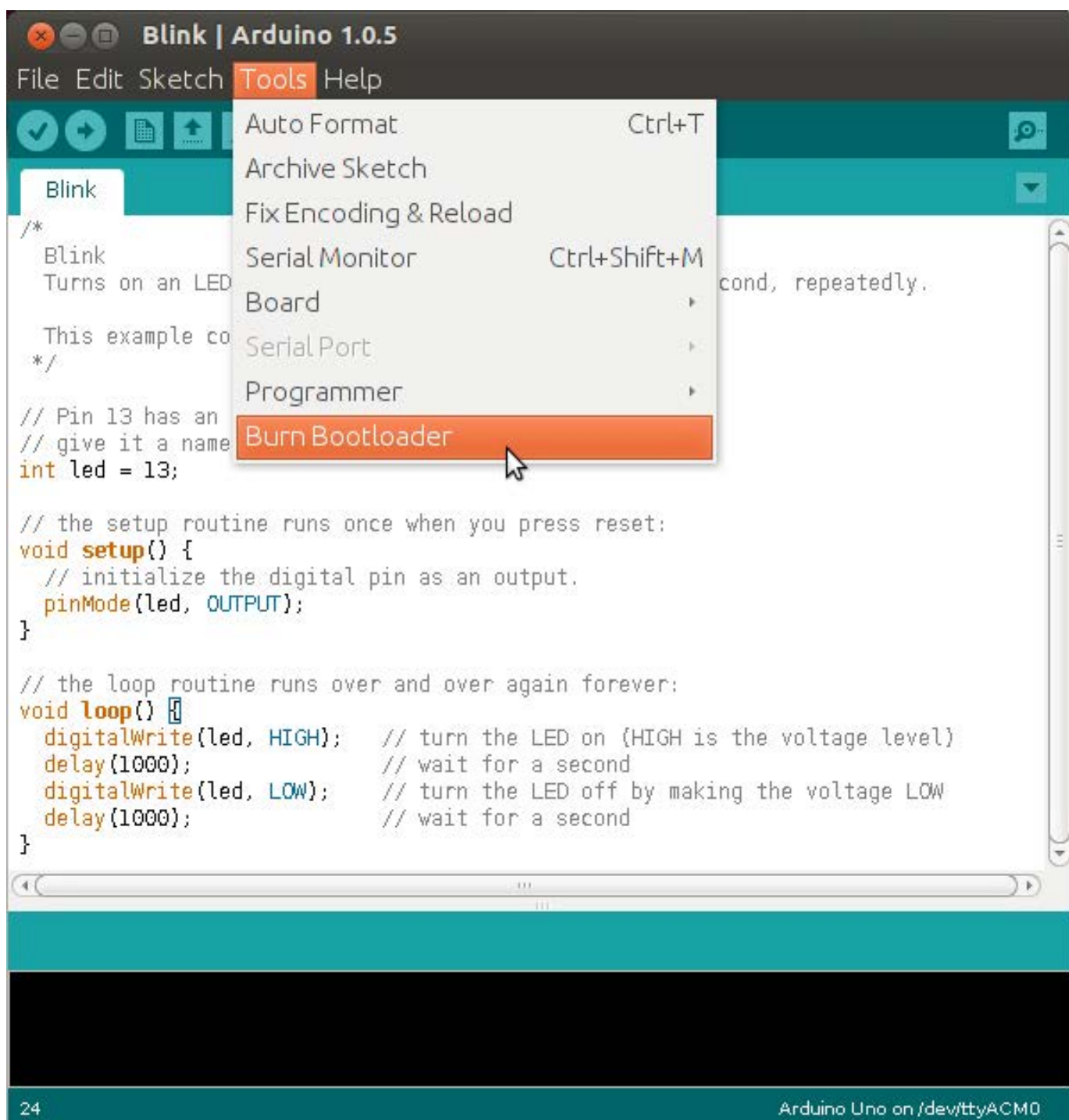


Figure 7. Finally, select Burn Bootloader.

reset line, arbitrarily pick a GPIO pin. For this example, pick pin P9_23, which is GPIO1_17. The BBB System

Reference Manual (SRM) contains the mapping of pins to pin mode in Table 13 for the P9 Header.

Let's further break down the GPIO number. The 1 in GPIO1_17 refers to the GPIO controller. There are four GPIO controllers on the BBB, numbered 0 through 3. The second number, 17, is the pin ID on that GPIO controller. The interface to the Linux kernel's GPIO driver requires you to use a single GPIO number. Therefore, you need to understand the mapping from the BBB's SRM GPIO naming convention to the kernel. To help understand, mount the debugfs, if it's not already mounted with:

```
mount -t debugfs none /sys/kernel/debug
```

Then, view the GPIO information with:

```
cat /sys/kernel/debug/gpio
```

The result should be something like this:

```
GPIOs 0-31, gpio:
gpio-6 (mmc_cd ) in hi
```

```
GPIOs 32-63, gpio:
gpio-49 (sysfs ) out hi
gpio-52 (eMMC_RSTn ) out lo
gpio-53 (beaglebone:green:usr) out hi
gpio-54 (beaglebone:green:usr) out lo
gpio-55 (beaglebone:green:usr) out hi
gpio-56 (beaglebone:green:usr) out lo
```

```
gpio-59 (McASP Clock Enable P) out hi
GPIOs 64-95, gpio:
```

```
GPIOs 96-127, gpio:
```

Here you see the four GPIO controllers on the BBB. The GPIO you want to use is GPIO1_17, which you now know is on GPIO controller 1. You also now know the offset. GPIO0 controls 32 GPIOs, 0–31. GPIO1 also controls 32 GPIOs, 32–63. You want the 17th GPIO on GPIO controller 1, which starts at 32; therefore, the number you need is $32 + 17 = 49$.

Now that you know the mapped pin number, you can control the pins from userspace with basic shell scripting. It's a three-step process. First, export the desired pin. Then, set the direction of the pin for either input or output. Finally, set the pin to a logic high ("1") or low ("0").

Export the pin with the following command, as root:

```
echo 49 > /sys/class/gpio/export
```

With the pin now exported, set the direction and value with the following two commands:

```
echo out > /sys/class/gpio/gpio49/direction
echo 1 > /sys/class/gpio/gpio49/value
```

The first command declares the pin as an output, and the second command sets the reset line high and powers the ATmega328p. Now your reset line is ready to go.

It is a bit trickier to enable the TTL serial lines. Per the BBB SRM, I've chosen UART4 TX and RX, which maps to P9_13 and P9_11. However, because these are going to be used in a different mode, namely serial TTL, the pins cannot be mapped like normal GPIOs. In this case, for the 3.8 kernel series on the BBB, it's easiest to build a device tree fragment and load it at runtime with the BeagleBone's Cape Manager (Capemgr).

The BBB uses the Linux kernel's device tree system, but the Capemgr is a BBB-specific solution. For complete BBB capes, during system startup, the kernel will probe the four authorized EEPROM addresses and try to match the cape name and version, stored in the EEPROM, with compiled device tree files. But, it's also possible to load a compiled device tree object at runtime from userspace.

The following snippet shows the key insight to configuring the device tree on the BBB; it is based on the fragments for the BeagleBone Black in `am335x-bone-common.dsti` (the complete code is in my GitHub repository

listed in the Resources section):

```
fragment@0 {
    target = <&am33xx_pinmux>;
    __overlay__ {
        pinctrl_uart5: pinctrl_uart5_pins {
            pinctrl-single,pins = <
                0x070 0x26 /* P9_11 = MODE6 */
                0x074 0x06 /* P9_13 = MODE6 */
            >;
        };
    };
};
```

The “magic” numbers contained in the `pinctrl-single,pins` property warrant some explanation. This device tree snippet is changing the mode of the pins with the kernel's Pin Control subsystem. Specifically, the driver is the `pinctrl-single` driver. The documentation for that driver states that the first 32-bit value in the two 32-bit value pair (known as “cells” in the device tree documentation) is a register offset whose value to be set is the second 32-bit value. The source document that describes the registers is the ARM Cortex-A8 AM335x's Technical Reference Manual. This is the manual for the processor on the BBB. The second value is a hex encoding for the mode and control for that pin. The least significant six bits of this value are encoded per Table 1.

Table 1. Bit Encoding

Bit	0	1
6 - Slew Control	Fast	Slow
5 - Receiver Active	Disabled	Enabled
4 - Pullup or Pulldown	Pulldown	Pullup
3 - Enable Pulls	Enabled	Disabled
2,1,0 - Mux Mode	Mode 0 through 7	

In this example, P9_11 is decoded as follows: fast slew control, receiver active, internal pulldowns are disabled, and mode 6. In the BBB System Reference Manual, mode 6 for this pin corresponds to UART4_RXD, which is exactly what you want.

With the device tree source in hand, compile the file with the device tree compiler (dtc), copy it to the /lib/firmware directory and enable the overlay.

```
dtc -O dtb -o enable-uart5-00A0.dtbo -b 0 -@ enable-uart5.dts
```

```
cp enable-uart5-00A0.dtbo /lib/firmware/
```

```
echo enable-uart5 > /sys/devices/bone_capemgr.*/slots
```

If dtc complains about the @ symbol, update dtc with the following script:

```
wget -c https://raw.githubusercontent.com/RobertCNelson/tools/master/pkgsg/dtc.sh
```

```
chmod +x dtc.sh
```

```
./dtc.sh
```

This will download a patched version of the dtc that supports this feature.

Run `dmesg|tail` to verify that the device tree was loaded successfully, and you should see messages similar to this:

```
bone-capemgr bone_capemgr.8: slot #8: Requesting
    part number/version based 'enable-uart5-00A0.dtbo
bone-capemgr bone_capemgr.8: slot #8: Requesting
    firmware 'enable-uart5-00A0.dtbo' for board-name
    'Override Board Name', version '00A0'
bone-capemgr bone_capemgr.8: slot #8: dtbo
    'enable-uart5-00A0.dtbo' loaded;
    converting to live tree
bone-capemgr bone_capemgr.8: slot #8: #3 overlays
481a8000.serial: ttyO4 at MMIO 0x481a8000
    (irq = 61) is a OMAP UART4
```

The last line tells you that the serial device, /dev/ttyO4, is now ready.

Uploading Sketches

With the hardware in place and the interfaces ready, there are two pieces remaining. You need a mechanism to transmit sketches from the host

computer to the BBB, and then you need a mechanism to upload the sketch to the ATmega328p. The first piece can be solved in numerous ways. I wrote a simple Python Web server that receives sketches via a POST and then launches the script to flash the ATmega328p. There are lots of examples of how to do this in Python, so I'll leave this as an exercise for the reader. Although, you can take a look at `server.py` in my GitHub repo for this tutorial listed in the Resources section

for an example.

The Arduino IDE compiles and uploads sketches directly from the GUI. You still can use the IDE to compile sketches, but your BBB programmer isn't supported. In the Arduino preferences, select verbose output for compilation. When you compile your sketch, the output window will show the location of your compiled file which should be something like: `Blink.cpp.hex`.

The script to upload the sketch

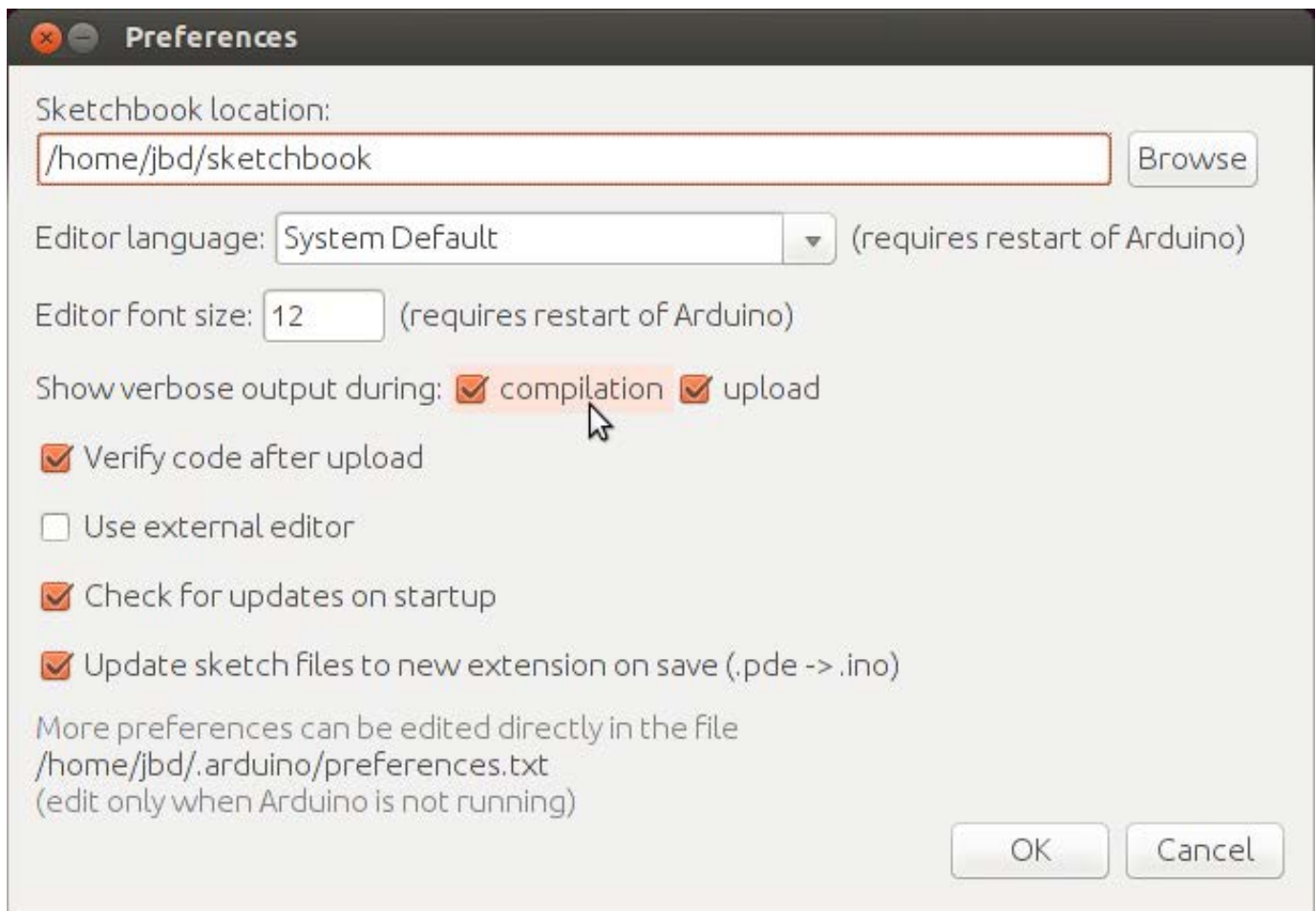


Figure 8. Selecting verbose compilation is helpful in finding the compiled sketch.

to the ATmega328p is a bit trickier. Before proceeding, install the avrdude package on the BBB. Avrdude is the "AVR Downloader/UplodDEr" and is the back-end utility that the Arduino IDE uses to upload sketches. Avrdude alone should be sufficient, but if you want the full tools on your BBB, install the following extra packages:

- gcc-avr
- binutils-avr
- gdb-avr avr-lib

This script works on the principle that the ATmega328p bootloader will monitor the serial TTL lines shortly after a reset for incoming programs. Therefore, you must coordinate the reset of the ATmega328p with the start of the transmission. Use the following bash snippet to upload the sketch:

```
#!/bin/bash
if [ "$#" -lt 1 ]
then
    echo "Usage: $0 sketch.cpp.hex [time to sleep]"
    exit 1
fi

pin=49
serial=/dev/tty04
```

```
if [ "$2" == "" ]; then
    tts=.9
else
    tts=$2
fi

echo Waiting $tts seconds

(echo 0 > /sys/class/gpio/gpio$pin/value \
    && sleep $tts \
    && echo 1 > \
    /sys/class/gpio/gpio$pin/value) &

avrdude -c arduino -p m328p -b 57600 -v \
    -P $serial -U flash:w:$1
```

The first command splits off a subshell that will toggle the reset line. After a brief delay, the avrdude command flashes the sketch. Use the -c option to indicate the arduino programmer, which is the closest setup to this hardware. Since the processor is running at 8MHz vs. 16MHz, cut the serial baud rate in half (-b 57600).

You may need to adjust the timing values. A logic analyzer is very helpful in this situation to see the logic levels; otherwise, you are left to trial and error. If you intend to dive deeper into hardware, a logic analyzer is well worth your investment.

The text should scroll by and you

should see output similar to this:

```
avrdude: reading input file "sketch.hex"  
avrdude: input file sketch.hex auto detected as  
Intel Hex  
avrdude: writing flash (1084 bytes):  
  
Writing | ##### | 100% 0.30s
```

If you've uploaded the standard Arduino Blink sketch, the LED on your cape should be blinking! Congratulations, you've just programmed the ATmega328p from your BeagleBone Black!

Next Steps

This project showed the minimal configuration needed to program an ATmega328p from a BBB. Serial TTL is just one way to communicate with the ATmega328p, however. Many devices can be connected on the I2C bus, including the ATmega328p. What's nice

about I2C on the BBB is I2C is enabled by default on pins P9_19 and P9_20. There are no extra device tree overlays or steps required to configure this. Also, both the BBB and the ATmega328p can communicate over the Serial Peripheral Interface (SPI) bus.

Be warned, hardware hacking is addictive! The first time you build a semi-complicated circuit and watch the LED blink in front of you, it's like the first time you booted into your modified kernel. Have fun and Happy Hacking! ■

Josh Datko is the founder of Cryptotronix (<http://www.cryptotronix.com>), a maker of open-source hardware crypto devices. He is also a submarine veteran and founding member of Loveland CreatorSpace.

|||||
Send comments or feedback via
<http://www.linuxjournal.com/contact>
or to ljeditor@linuxjournal.com.

Resources

- SparkFun Wish List with All Components: <http://sfe.io/w80644>
- GitHub Repo for Tutorial Code: https://github.com/jbdatko/BBB_ATmega328P_flasher
- CryptoCape Project Details: <http://beagleboard.org/project/CryptoCape>
- SparkFun Soldering Tutorial: <https://learn.sparkfun.com/tutorials/how-to-solder---through-hole-soldering>
- Details on the BBB Cape Manager: <http://elinux.org/Capemgr>
- Linux Kernel Pin Control Subsystem: <https://www.kernel.org/doc/Documentation/pinctrl.txt>
- Linux Kernel GPIO Interface: <https://www.kernel.org/doc/Documentation/gpio/gpio.txt>

Build.
Empower.
Inspire.



Women in Technology Summit: *Powering Up!*

June 1-3, 2014 • Santa Clara Marriott • Santa Clara, CA

The Most Important Event for Women in Technology Returns in June!

Join WITI, the world's leading professional organization for executive women in technology, for a variety of hands on speaking engagements and dynamic panel discussions at WITI's Annual Summit.

**REGISTER
NOW!**

Special Offer

Use Promo Code **WOMEN**
and **SAVE \$200!**

Among the Featured Speakers & Session Chairs

- Gwynne Shotwell, President & COO, SpaceX
- Sandy Carter, General Manager Ecosystem Development & Social Business Evangelist, IBM
- Amanda Coolong, COO/Chief Host, TechZulu
- Guy Kawasaki, Author/Consultant/Entrepreneur
- And More... Visit witi.com/summit Today!

Including the WITI Hall of Fame

Key leaders in science and technology will be honored at the 19th Annual WITI Hall of Fame Awards Dinner on Monday, June 2, 2014 in conjunction with WITI's Annual Women in Technology Summit in the Silicon Valley.



witi.com/summit • facebook.com/witi • @WITI • #WITISummit

REGLUE

OPENING UP THE WORLD TO
DESERVING KIDS, ONE LINUX
COMPUTER AT A TIME

An interview with Ken Starks about his organization, Reglue (Recycled Electronics and Gnu/Linux Used for Education).

BRIAN CONNER



Ken Starks

They say you never forget your first computer. For some of us, it was a Commodore 64 or an Apple IIe. For others, it was a Pentium 233 running Windows 95. Regardless of the hardware, the fond memories of wonder and excitement are universal. For me, I'll never forget the night my father brought home our first computer, a Tandy 1000. Nor will I forget the curious excitement I felt toward the mysterious beige box

that took up a large portion of the guest bedroom. This happened at a time when simply having a computer at home gave a school-age child an advantage. I have no doubt my experiences from that time positively influenced my path in life.

In the decades that have passed since the beginning of the personal computer revolution, computers have gone from being a rare and expensive luxury to a mandatory educational

tool. Today, a child without access to a computer (and the Internet) at home is at a disadvantage before he or she ever sets foot in a classroom. The unfortunate reality is that in an age where computer skills are no longer optional, far too many families don't possess the resources to have a computer at home.

I recently had the opportunity to talk with Ken Starks about his organization, Reglue (Recycled Electronics and Gnu/Linux Used for Education) and its efforts to bridge this digital divide.

BC: Tell me about your computing background and how you got started with Linux—the obligatory “what was your first distro” question?

KS: My very first computer was an Optiplex GX-1. It was a Pentium II with a whole 64MB of RAM, and it was an absolute screamer—for its day anyway. By today's standards, it traveled at the speed of smell. In 2001, I bought nine Dell Dimension 4300's with which to build my business. My power-washing company had offices in Austin, San Antonio and Dallas. All of those ran the obligatory Microsoft Windows OS.

In the late summer of 2002, my Tech Guy called me one morning

and informed me that we had a big problem. Our entire system had been infected by the klez virus—all three offices. By the end of the day, it was clear that the only way to clean this up was to reformat and re-install. I promised to bring him the installation disks immediately, and he told me that I didn't need to worry about it. He would take care of it. We closed our offices for the week, and I paid my folks for their “vacation”. When I returned, all of my systems and my network was back up and running Linux—Libranet Linux to be specific. I've never looked back. Even after I sold my business, I remained a full-time Linux user.

BC: How did Helios/Reglue get started?

KS: I am the founder and Executive Director of Reglue, which began life as The Helios Project in 2005. This project started as “a lucky break”. I was working 38 feet in the air, pressure washing a building for my power-washing company when the lift failed and I fell, fracturing my spine at the neck. While restrained and convalescing, my then 11-year-old daughter Amanda asked me “How does a computer work?” I had her get one from the office, and we took it apart on my lap.

It was from there that The Helios Project began. What started as a hobby by way of boredom turned into my life's calling. I enlisted the help of two friends in gathering old machines and fixing them. The work surface evolved from a halo-device wheel chair to the dining room table to the garage at my house and then to a donated facility in Lakeway, Texas, and now a permanent facility in Taylor, Texas.

The HeliOS Project evolved into Reglue in the summer of 2012. Since 2009, HeliOS had been working under the umbrella of Software in the Public Interest. As an invite-only organization, HeliOS was glad to be included. It gave us the opportunity to give tax receipts to donors and gave HeliOS some recognition that it might not have had otherwise. By 2012, HeliOS had outgrown its association with SPI and had a fantastic opportunity to become its own 501(c)(3)—an opportunity that HeliOS simply didn't have the money to pursue.

That's when friend and supporter Don Davis stepped in to help. Don had an almost identical project named Reglue (Recycled Electronics and Gnu/Linux Used for Education) working in and around San Marcos, Texas, but being a Doctoral Candidate, he found little time to participate in the project. Don and his directors invited

me and our directors to a meeting. Don's board voted each of the HeliOS members in, and once that was done, each one in Don's organization resigned their positions. That left the HeliOS Project operating in an already-established nonprofit organization.

However, the HeliOS Project did not completely go away. It remains an active project under the Reglue umbrella. HeliOS is the educational arm of Reglue, responsible for all classes and computer education.

BC: What were the factors leading to the decision to use Linux? Which distros have been used and which one is being used now?

KS: In 2005, we tried to start with Windows because that was what everyone used. This effort was short-lived, however, because the best Microsoft could do was to offer us Windows XP SP1 licenses for \$50 each. For the first few years, we were not a nonprofit, and the whole thing was being funded out of my pocket. Even doing only ten computers a year would bankrupt us. It was clear Microsoft wasn't concerned about helping us. That's when we decided to become a 100% Linux shop.

We started out with Mepis and used it for a few years. We had

brief flirtations with Connectiva and Mandrake (and as an aside, I still believe that Connectiva was the best Linux distro available at the time). When they combined with Mandrake to become Mandriva, I didn't like the constant nagging to "upgrade to a fuller Mandriva experience". In 2009, we moved full time to Ubuntu.

Right now, we are using a combination of three Linux distros: ZorinOS 6.4 Educational version 6.4 LTS (Based on Ubuntu), an educational respin of Linux Mint 13 (Maya) with the Cinnamon desktop and OpenSUSE LiFe-Education edition. We had been working with prodigy extraordinaire Ikey Doherty of SolusOS fame, but when the help promised from the community for the 2.0 release backed out, he had to walk away. SolusOS was to be our distro when Ikey finished with our respin.

We will continue using Zorin and Mint, but if either one of those one-man-driven distros disappear, we always have OpenSUSE to fall back on. Besides, OpenSUSE's Studio on-line tool is easy enough for our non-tech volunteers. If an ISO needs certain tools, it can be done on the fly with Studio.

The OpenSUSE system is used for older kids and machines with 4 gigs of RAM or more and the PAE kernel, while ZorinOS and Mint are reserved

for younger kids and machines with 1–3 gigs of RAM.

BC: *What are the challenges of using Linux? I would imagine that the varying levels of hardware support by the Linux kernel and/or availability of drivers must present some challenges when trying to assemble working PCs from such a wide variety of hardware.*

KS: The challenges of hardware aren't even worthy of mention anymore. Somewhere around the 2.7 kernel release, wireless went from "Wireless sucks in Linux" to "Holy crap, wireless works in Linux". Of course, we still have a number of the Broadcom drivers and Texas Instrument chunks that still need a hard connection to download the wireless device firmware, but for the most part, we have few problems anymore.

The biggest challenge we face is keeping parents or siblings from blowing away the Linux system and putting Windows on it because they see something different and they freak. They can't play *World of Warcraft* out of the box, so they screw up the kid's computer by installing a cracked version of Windows. Then they call and complain a week later that they have a virus. We now have the parent or

guardian sign, saying that if another operating system is introduced, we will not support that computer any longer. We make sure they understand that. It happens only in maybe 8% percent of the cases, but a small organization like ours cannot spend time re-traveling our steps to fix someone else's mistake.

Besides, the kids take to Linux right away. We rarely have issues now, and most of those are failed hardware, which we replace. We're seeing more and more school districts using Chromebooks and Google on-line tools now. Google Docs has gained traction in the schools, and that has lessened the choke point Microsoft had with Office. The kids just turn in their work, and it is read or printed from Google Drive. As a taxpayer, I'm personally enjoying it. It bothers me that the schools we pay taxes to support spend money on software when they have no real need to do so. Up until lately, they had an excuse, but now with Google being such a factor, schools are beginning to take advantage of it and save money at the same time.

BC: What is Reglue's position regarding non-OSS (binary blob) drivers and software? Is the focus on ideals or on the best possible user experience? Are those mutually exclusive?

KS: At first, I was a hard-core Stallman-ista. I could recite to you pertinent parts of the GPL chapter and verse, but as it is with all unbending ideology and dogma, it will wither away when exposed to the direct light of pragmatism. We refer to this code as our "naughty bits". As much as I agree to the principles of Richard Stallman's Manifesto, it just doesn't work in the real world. I can't give a kid a computer, then present him with a list of things he can't do with it.

So in essence, without these tools, we are giving these kids a multi-component, 40 pound typewriter. I live and work in the real world, so I've come to an understanding. I'll use FOSS when I can and augment it with binary/proprietary blobs when I can't.

BC: How are the recipients of Reglue PCs chosen? Are there any expectations/requirements placed upon them?

KS: Now that we are well-known in a tri-county area, our referrals come from Child Welfare Offices, policemen, firemen, school teachers and other public workers as well as neighbors and friends. We really don't have a matrix that we use to qualify recipients—that hasn't worked for us. A single mom with four kids might

look like she's living large making 4K a month, but with the cost of daycare and other expenses, she's literally living at the poverty line.

These are the people we are trying to help. Every one of our candidates is considered on a case-by-case basis. Once a family is selected, we make an "initial visit" to the home to see if they qualify. Most of the time, though, when we show up for an initial visit, we have their computer in the delivery vehicle. Once we check things out, we install on the spot.

BC: How many PCs have you placed in the community? Do you provide follow-up support?

KS: To date, we've installed close to 1,600 computers.

Once a Reglue computer is installed, it is supported for the child's entire school attendance, even through college. Unfortunately, many of the people we help move around a lot, so the recorded address of any given install has a 65% chance of being incorrect after the first year. But yes, we provide support for as long as it's needed, up the point of replacing the entire computer/laptop.

BC: What are Reglue's biggest successes/achievements?

KS: We've had many shining moments, but one in particular stands out. We were approached by a young woman named Christina Collazo in the summer of 2009. She is the Director for Sí Se Puede Learning Center. One of her centers was located in a large Catholic Church on the east side of Austin. She asked me if our organization could set up a 25-station computer learning center in the Church. It was to be a center open to the community at large, as well as being used by daycare kids and parents. She figured she would ask for the moon and settle for what she could get. To her surprise, we were able to put that project together.

Shortly after that, we received notice that an icon of Linux support was dying of brain cancer. Bruno Knaapen was a well-known provider of support and encouragement for those new to Linux. He was personally responsible for answering more than 40,000 inquiries on Scot's Newsletter Forum. I thought it only fitting to name our largest learning center as The Bruno Knaapen Technology Learning Center. We had the privilege of doing so before he died. It made him happy at a time when happiness was in short supply for him.

While we have had our victories, I think building that center stands out



Kids at The Bruno Knaapen Technology Learning Center

the most. Since then, we've set up four more, but nothing as large as the 2009 effort.

BC: *What is the status of Project Reglue now? And what are your plans for the next five years?*

KS: At the moment, we're juggling our normal business with getting

our shop in better shape. I was off work for more than a year while I recovered from cancer treatments. However, that didn't slow down the flow of donated computers and equipment. Our volunteers were bringing in computers of all types and stacking them in the workshop. When I was able to go to work, I walked into a pile of computers, monitors and other hardware that we still don't have properly sorted, but we're making progress.

At this time, we have two working projects. One is The 12 Geeks of Christmas. We are asking 12 people from anywhere in the US to find a child who's family cannot afford a computer. We will ship a laptop or desktop to the volunteer to set it up in the child's home, give them some instruction and provide support in the event they might need it (note: this interview was conducted in late November 2013, please see the Resources section at the end for follow-up on this project).

The second thing we have working is a partnership with the Taylor Texas Fire Department's "Red Santa" effort. They will identify ten kids in the area who need a computer, and we will install one, even on Christmas day if necessary. Of course, this isn't just for those who

celebrate Christmas. Anyone from any faith can participate. I'm not concerned about what banner it's done under, as long as it's done.

But speaking to the future, we hope to get back to the level where we once were. Installing 200–300 computers a year is a huge undertaking. Now that I have much of my health returning, there is no reason we can't do that. We're also going to look more closely to funding. The money we have now won't last long, so we have to plan for the future. The biggest problem we've faced in getting funds is the lack of grants used for day-to-day expenses. Foundations want to give you equipment or money for equipment, but we already have equipment. What good is having computers to give if you can't put fuel in the vehicle to deliver them? Hopefully, we'll find that source in the coming year.

BC: *Recently you wrote a blog post about how you're working with an Internet service provider to bring high-speed Internet into the homes of Reglue recipients. Can you tell us more about the status of that effort?*

KS: Getting Internet connections for

our Reglue kids can be a challenge. Many of our kids mow lawns, babysit and do odd jobs around their neighborhood to pay for their monthly Internet costs. But sometimes, that's just not possible.

For the most disadvantaged families we serve, we've set aside enough money to help these people get connected. We call this our Prometheus Project. We've made arrangements with Time Warner to pay the initial fees and three months Internet service for these people. At this time, we have enough to help 20 families a year. That gives them 90 days to budget for their Internet costs (see the Resources section at the end of this article for more information on this project).

BC: *What advice do you have for people who are considering starting an effort like Reglue in their community?*

KS: Brian, first off, you need to have a source of project funding. For four years I ran the whole thing out of my house and out of my pocket. That was fine when I was making a six-figure income, but when you whittle that down by 90%...well, you simply can't do it.

You also need to have a source of

WEBCASTS



Learn the 5 Critical Success Factors to Accelerate IT Service Delivery in a Cloud-Enabled Data Center

Today's organizations face an unparalleled rate of change. Cloud-enabled data centers are increasingly seen as a way to accelerate IT service delivery and increase utilization of resources while reducing operating expenses. Building a cloud starts with virtualizing your IT environment, but an end-to-end cloud orchestration solution is key to optimizing the cloud to drive real productivity gains.

> <http://lnxjr.nl/IBM5factors>



Modernizing SAP Environments with Minimum Risk—a Path to Big Data

Sponsor: **SAP** | Topic: **Big Data**

Is the data explosion in today's world a liability or a competitive advantage for your business? Exploiting massive amounts of data to make sound business decisions is a business imperative for success and a high priority for many firms. With rapid advances in x86 processing power and storage, enterprise application and database workloads are increasingly being moved from UNIX to Linux as part of IT modernization efforts. Modernizing application environments has numerous TCO and ROI benefits but the transformation needs to be managed carefully and performed with minimal downtime. Join this webinar to hear from top IDC analyst, Richard Villars, about the path you can start taking now to enable your organization to get the benefits of turning data into actionable insights with exciting x86 technology.

> <http://lnxjr.nl/modsap>

WHITE PAPERS



White Paper: JBoss Enterprise Application Platform for OpenShift Enterprise

Sponsor: **DLT Solutions**

Red Hat's® JBoss Enterprise Application Platform for OpenShift Enterprise offering provides IT organizations with a simple and straightforward way to deploy and manage Java applications. This optional OpenShift Enterprise component further extends the developer and manageability benefits inherent in JBoss Enterprise Application Platform for on-premise cloud environments.

Unlike other multi-product offerings, this is not a bundling of two separate products. JBoss Enterprise Middleware has been hosted on the OpenShift public offering for more than 18 months. And many capabilities and features of JBoss Enterprise Application Platform 6 and JBoss Developer Studio 5 (which is also included in this offering) are based upon that experience.

This real-world understanding of how application servers operate and function in cloud environments is now available in this single on-premise offering, JBoss Enterprise Application Platform for OpenShift Enterprise, for enterprises looking for cloud benefits within their own datacenters.

> <http://lnxjr.nl/jbossapp>

WHITE PAPERS



Linux Management with Red Hat Satellite: Measuring Business Impact and ROI

Sponsor: **Red Hat** | Topic: **Linux Management**

Linux has become a key foundation for supporting today's rapidly growing IT environments. Linux is being used to deploy business applications and databases, trading on its reputation as a low-cost operating environment. For many IT organizations, Linux is a mainstay for deploying Web servers and has evolved from handling basic file, print, and utility workloads to running mission-critical applications and databases, physically, virtually, and in the cloud. As Linux grows in importance in terms of value to the business, managing Linux environments to high standards of service quality — availability, security, and performance — becomes an essential requirement for business success.

> <http://lnxjr.nl/RHS-ROI>



Standardized Operating Environments for IT Efficiency

Sponsor: **Red Hat**

The Red Hat® Standard Operating Environment SOE helps you define, deploy, and maintain Red Hat Enterprise Linux® and third-party applications as an SOE. The SOE is fully aligned with your requirements as an effective and managed process, and fully integrated with your IT environment and processes.

Benefits of an SOE:

SOE is a specification for a tested, standard selection of computer hardware, software, and their configuration for use on computers within an organization. The modular nature of the Red Hat SOE lets you select the most appropriate solutions to address your business' IT needs.

SOE leads to:

- Dramatically reduced deployment time.
- Software deployed and configured in a standardized manner.
- Simplified maintenance due to standardization.
- Increased stability and reduced support and management costs.
- There are many benefits to having an SOE within larger environments, such as:
 - Less total cost of ownership (TCO) for the IT environment.
 - More effective support.
 - Faster deployment times.
 - Standardization.

> <http://lnxjr.nl/RH-SOE>

Oscad: Open-Source Computer-Aided Design Tool

Looking for an open-source CAD tool for your tablet? Look no further—Oscad is the answer!

RAKHI R AND KANNAN M. MOUDGALYA

Are you fascinated by electronic systems? Have you ever wondered how they work? Do you design electronic circuits as a hobby? Have you been discouraged by the high cost of proprietary design tools? Oscad is the answer! Oscad is a free and open-source computer-aided design (CAD) tool that lets you design, simulate, analyze and produce printed circuit board layouts for your favourite electronic circuits. Oscad is created using open-source software packages: KiCad (<http://www.kicad-pcb.org/display/KICAD/KiCad+EDA+Software+Suite>),

Ngspice (<http://ngspice.sourceforge.net>), Scilab (<http://www.scilab.org>) and Python. It is useful for students, teachers, professionals and entrepreneurs who are looking for an alternative to expensive proprietary CAD tools.

This article introduces Oscad and its capabilities with a lot of illustrative examples. It also describes various resources, such as our book on Oscad *Oscad: An Open Source EDA Tool for Circuit Design, Simulation, Analysis and PCB Design* (<http://www.oscad.in/resource/book/oscad.pdf>), tutorials and example projects. Additionally,

we outline initiatives to promote Oscad to the public.

Installation and Validation

Download the Oscad installer for Linux available at the Oscad Web page (<http://www.oscad.in/downloads>). Go to the Spoken Tutorial page (<http://www.spoken-tutorial.org>) and click on the Video Search option. In the Select Language drop-down menu, choose English and click Locate Tutorial. Listen to the Spoken Tutorial titled "Introduction to Oscad". Follow the steps shown in the tutorial to install Oscad and validate the installation. Note that the tutorial covers the installation and use of Oscad in Ubuntu Linux.

Getting Started with Oscad

To invoke Oscad, double-click on the Oscad icon on your desktop. Alternately, you can type `oscad` in a terminal to launch it. This opens up the Oscad project window (Figure 1).

Here you can open an existing Oscad project or create a new one. Click on Project and then New. Choose a directory and enter a project name. A vertical toolbar will open up. This is the Oscad Toolbar (Figure 2), and it has the following tools:

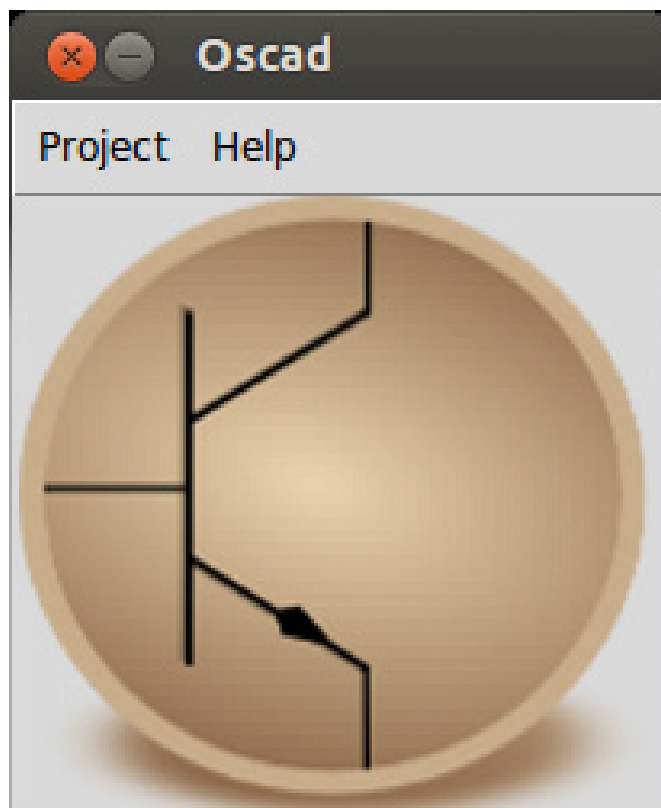


Figure 1. Oscad Project Window

- **Schematic Editor:** lets you create circuit schematics, perform electric rules checks and generate netlists for simulation and PCB design. EESchema (the schematic editor in KiCad) is used as the schematic editor.
- **Analysis Inserter:** lets you add simulation parameters for DC, Nested DC, AC and Transient analyses.
- **Netlist Converter:** converts the netlist generated by the schematic editor (EESchema) into

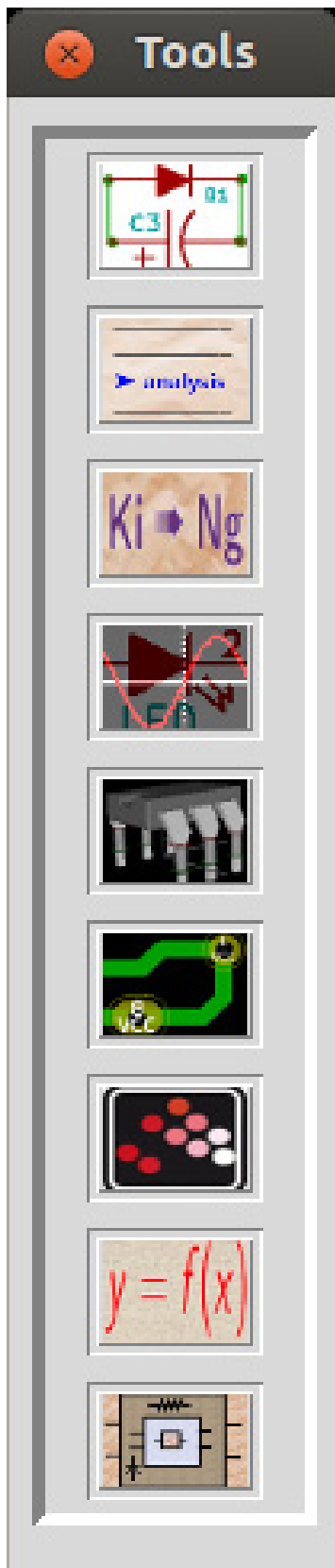


Figure 2. Oscad Toolbar

Ngspice-compatible format. It also adds the analysis parameters generated by the Analysis Inserter, plot commands, model and subcircuit information and other control statements to the netlist.

- **Ngspice:** simulates the netlist produced by the Netlist Converter and displays the results.
 - **Footprint Editor:** maps each component in the netlist to a corresponding footprint module. Manual and automatic mapping is possible. CvPcb (the footprint editor in KiCad) is used as the footprint editor.
 - **Layout Editor:** lets you create and edit PCB layouts for the circuit. PCBnew (the layout editor in KiCad) is used. It also lets you choose design rules,
- add layers, vias and plot Gerber files.
- **SMCSim:** stands for Scilab-based Mini Circuit Simulator. It generates symbolic equations for the circuit and solves them using Scilab. This is a unique feature of Oscad. At the time of this writing, this feature is available only for analog circuits.
 - **Model Builder:** helps you build models for devices, such as diodes and transistors (BJT, MOSFET) in your circuit.
 - **Subcircuit Builder:** lets you create subcircuits and use them for simulation. For example, an operational amplifier like UA741 or a timer chip like LM555N can be defined as subcircuits. You can create your own subcircuits and attach them to the main schematic.

Together, these tools make OScad a complete electronic system design suite.

The Design Flow

Let's look at the steps involved in the design of an electronic system. For simplicity's sake, let's consider the example of an RC circuit. The design flow is broadly divided into three stages.

Schematic Creation Chapter 5 of

our *Oscad* book explains schematic creation in detail. It also is covered in the Spoken Tutorial: Schematic Creation and Simulation Using OScad (<http://oscad.in/resources/tutorials/Oscad>). First, let's create the schematic of an RC circuit. Click on the Schematic Editor tool from the OScad toolbar to launch it. Add the OScad libraries to your project using

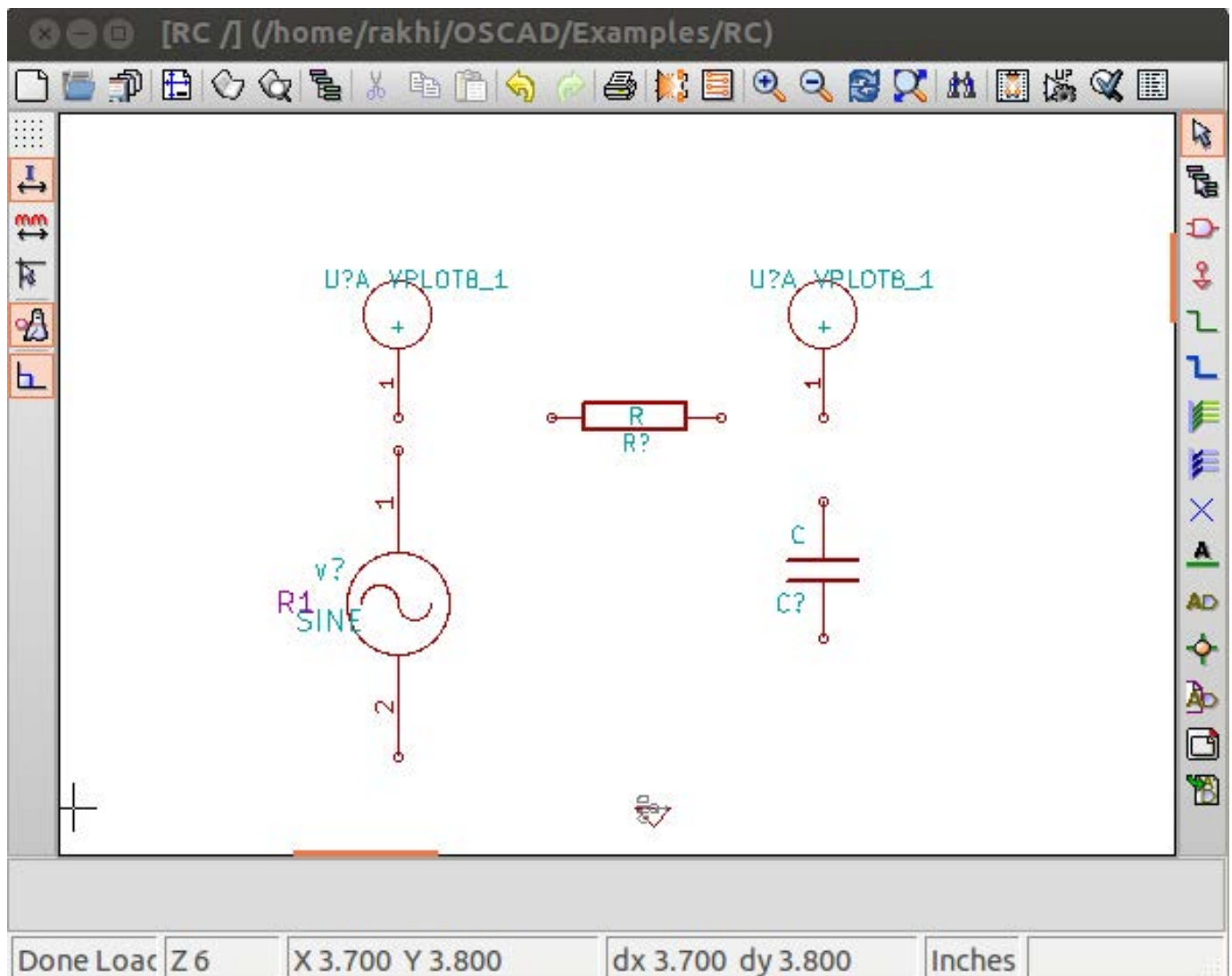


Figure 3. Components Placed in the Schematic Editor

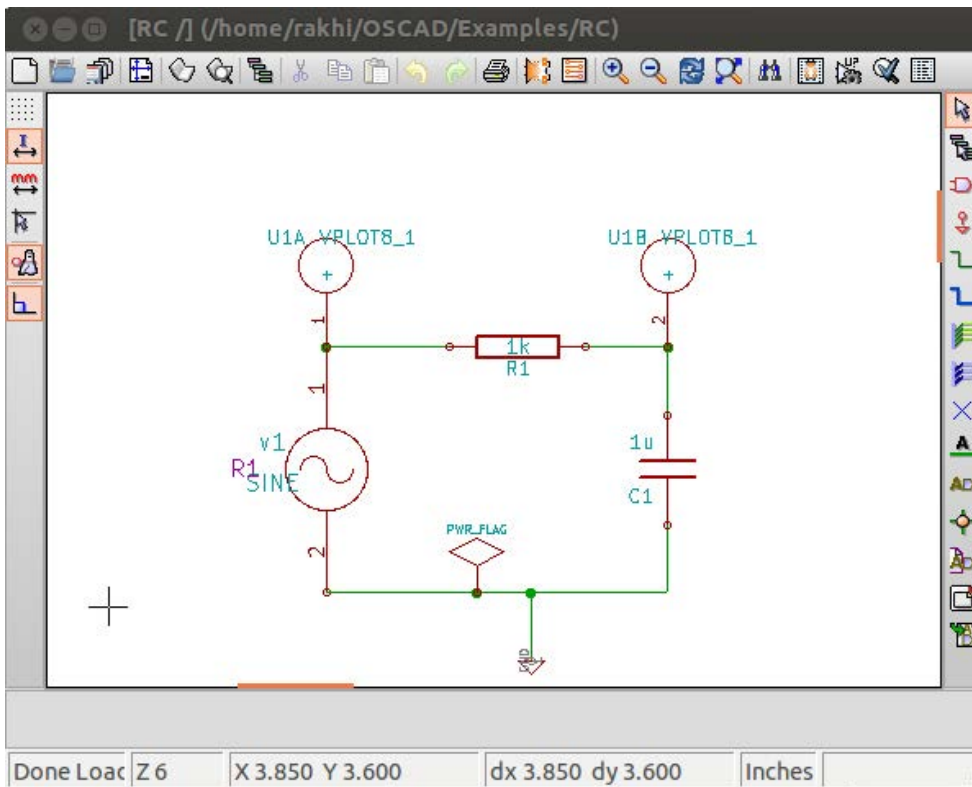


Figure 4. Final RC Circuit for Simulation

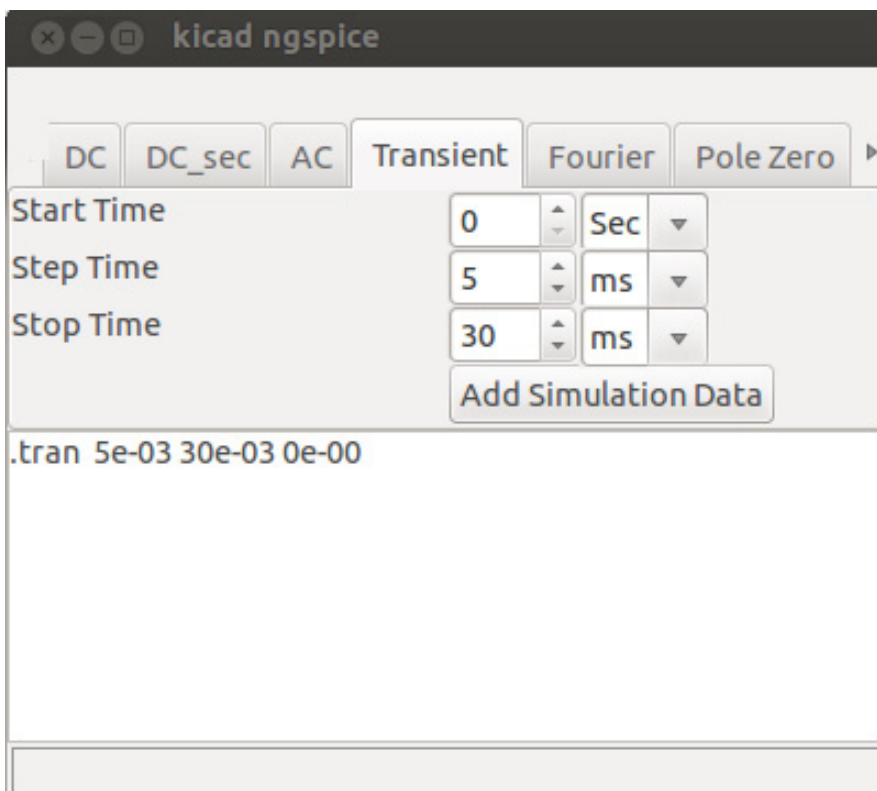


Figure 5. Adding Simulation Parameters

the Library option available in the Preferences menu. These libraries are available in the library folder in the Oscad installation directory. Add components to your schematic using the Place a Component option available from the right toolbar. Place a resistor, a capacitor, a ground and a sine source. Figure 3 shows the components placed in the Schematic Editor.

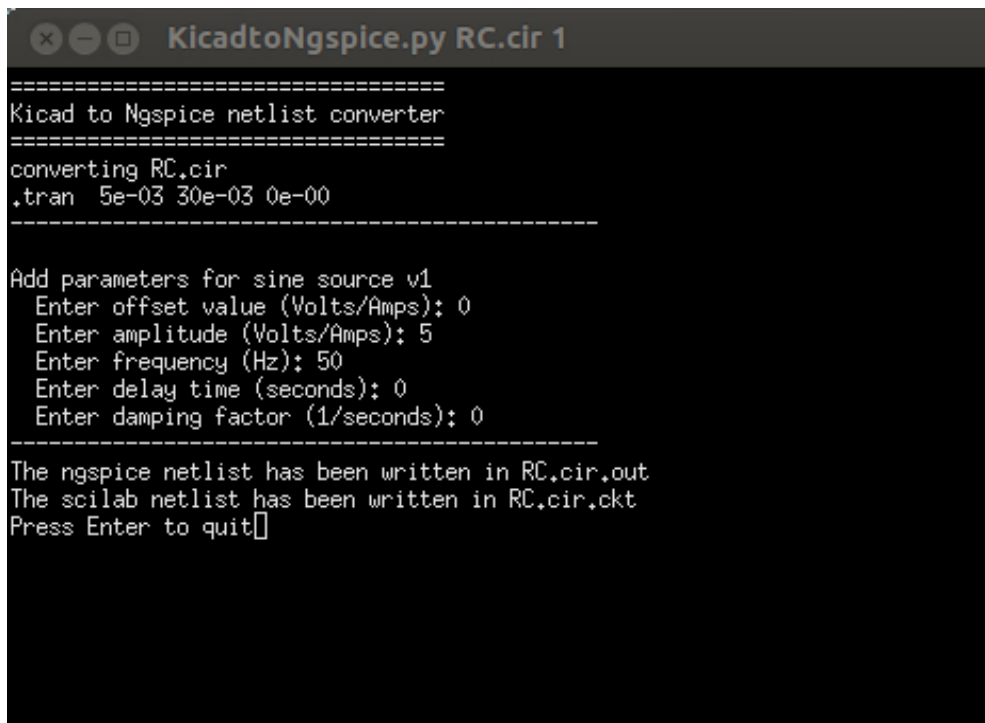
Assign values to the components and connect them using wires as shown in Figures 3 and 4. Annotate the schematic and check for electric rules (erc). You may want to connect voltage or current plot components for viewing outputs after simulation,

but these are not required for PCB design. Figure 4 shows the completed circuit schematic for simulation. Note the additional component PWR_FLAG. This is used to eliminate ERC errors.

Netlists can be generated separately for simulation and PCB design. Click on Generate Netlist from the top toolbar and choose Spice. Uncheck the option Prefix references "U" and "IC" with "X" and click on Netlist. Save the netlist. It has extension .cir.

Simulation

Chapter 6 of our *Oscad* book details simulation of circuits using Oscad. This is also explained in the Spoken



```

KicadtoNgspice.py RC.cir 1
=====
Kicad to Ngspice netlist converter
=====
converting RC.cir
.tran 5e-03 30e-03 0e-00
-----

Add parameters for sine source v1
Enter offset value (Volts/Amps): 0
Enter amplitude (Volts/Amps): 5
Enter frequency (Hz): 50
Enter delay time (seconds): 0
Enter damping factor (1/seconds): 0
-----

The ngspice netlist has been written in RC.cir.out
The scilab netlist has been written in RC.cir.ckt
Press Enter to quit

```

Figure 6. Give parameters for the sine source during the netlist conversion.

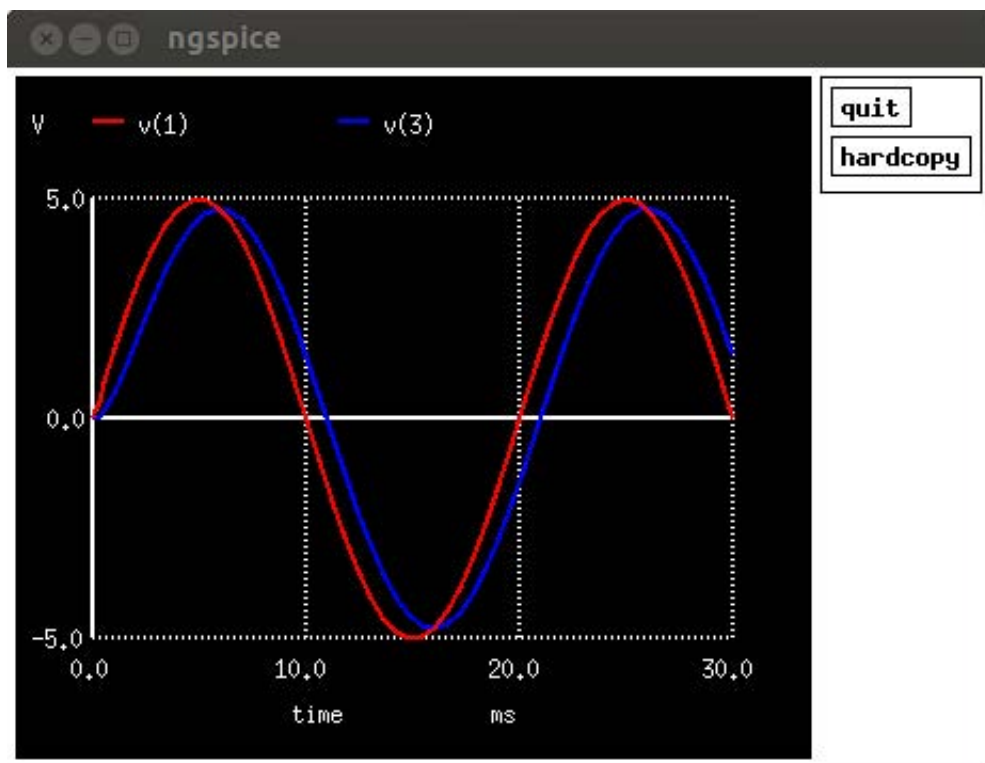


Figure 7. Transient Simulation Output of the RC Circuit

Tutorial: Schematic Creation and Simulation using Oscad. The steps are as follows. Click on the Analysis Inserter tool from the Oscad toolbar to add simulation parameters. Choose Transient, and enter the parameters: start time = 0 seconds, step time = 5 ms (milliseconds) and stop time = 30 ms. Click on Add Simulation Data (Figure 5). Save and exit, and don't change the filename. Next, click on the Netlist Converter tool from the Oscad toolbar. Enter the parameters of the sine source: offset = 0, amplitude = 5, frequency = 50, delay time = 0 and damping factor = 0 (Figure 6). Press Enter to quit. Click on the Ngspice tool

from the Oscad toolbar to simulate the circuit and view the output plots. Figure 7 shows the plots.

PCB Design Once you are satisfied with the simulation results, you can start the PCB design. You may want to replace the sine source with a connector in your schematic. Remove the plot components. Figure 8 shows the modified schematic for the PCB design.

Next, let's generate the netlist for the PCB design. To do this, click on Generate Netlist, select Pcbnew, and click on Netlist. Note that the extension of the netlist is .net. Save the netlist, and save and close the Schematic Editor. Chapter 7 of our

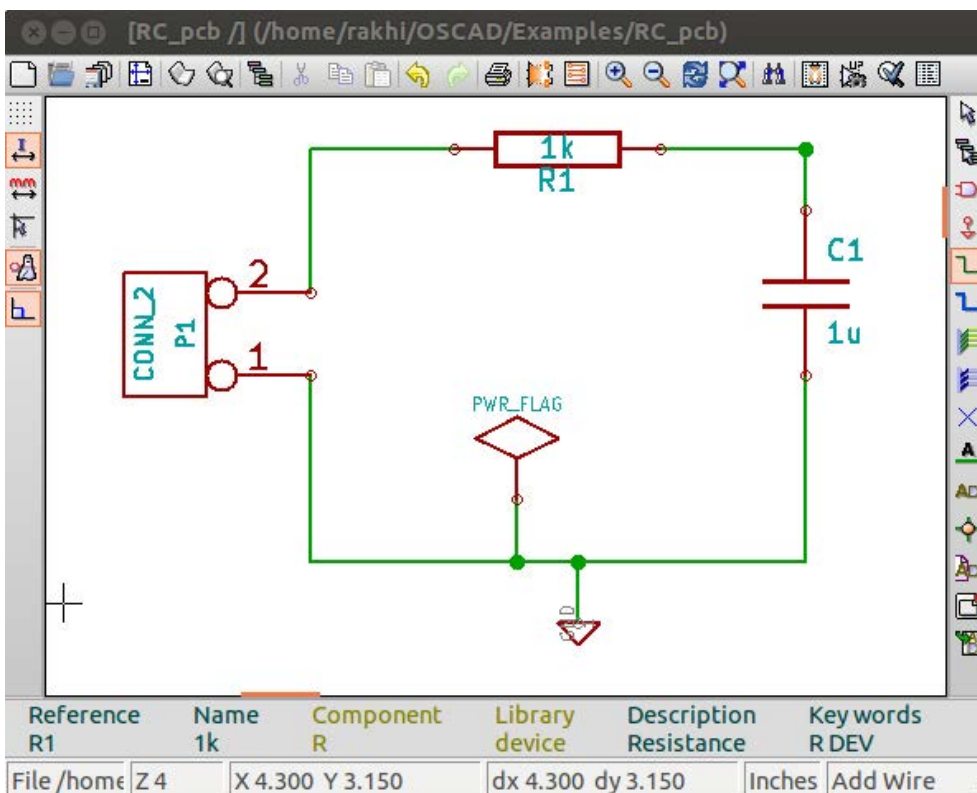


Figure 8. Final Schematic for PCB Design

Oscad book explains PCB design using Oscad. You also can refer to the Spoken Tutorial: Designing Printed Circuit Board using Oscad. There are two stages in PCB design, and they are explained below.

1) *Footprint Mapping*: The first step in PCB design is to assign a footprint

to each of the components in the schematic. Footprint refers to the physical layout (for example, through-hole, surface mount) on the printed circuit board to which the component will be mounted. Click on the Footprint Editor tool from the Oscad toolbar. It shows the components in your design on the left side and the available footprints on the right side. You can assign footprints to your components either manually or automatically. Figure 9 shows the final component list with

the footprint mapping done. You can view the footprints in 2-D and 3-D. Browse and save the netlist in the project folder and close the Footprint Editor.

2) *Layout Design*: Launch the Layout Editor from the Oscad toolbar. Import your netlist using the Read Netlist option in the menu bar. Then browse and open the netlist. Click on Read Current Netlist. The component footprints will be placed in the top-left corner of the layout editor window. Place

Component ID	Component Name	Footprint Name
1	C1 -	1u : C1
2	P1 -	CONN_2 : SIL-2
3	R1 -	1k : R3
1	C1	
2	C1-1	
3	C2	
4	SM0603	
5	SM0805	
6	SM1206	
7	SM1206POL	
8	SM1210	
9	SM1210L	
10	SM1812	
11	SM1812E	
12	SM1812L	
13	SM2112L	
14	SM2512	

Components: 3 (free: 0) | Footprints (filtered): 15

Figure 9. Footprint Mapping for the RC Circuit

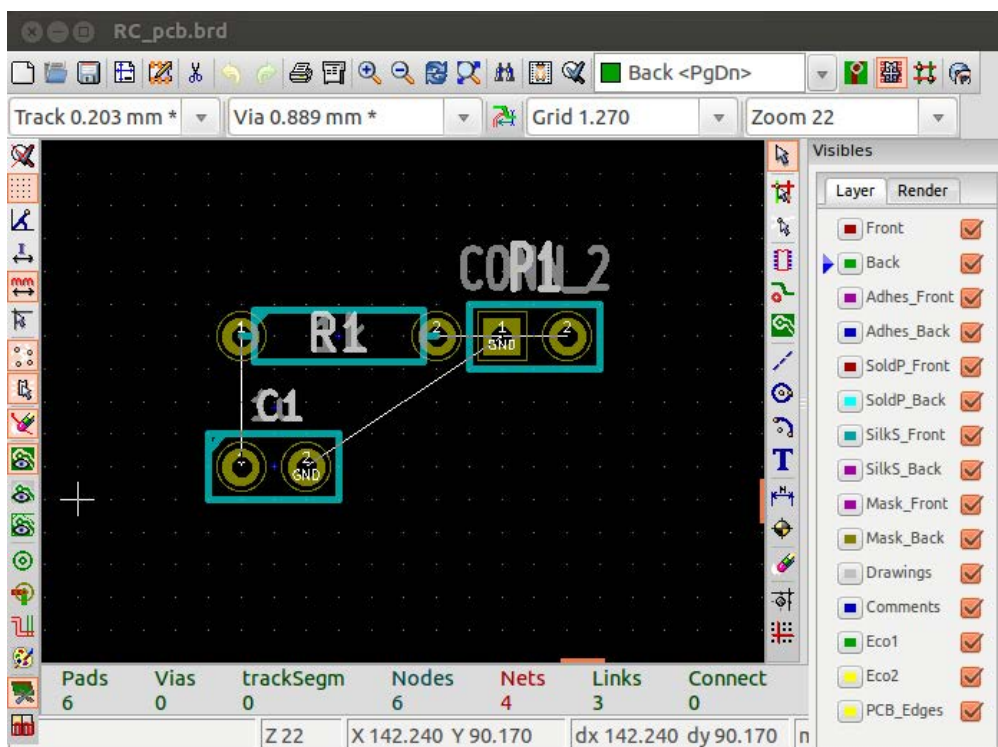


Figure 10. Footprint Modules of the RC Circuit at the Center of the Layout Editor Window

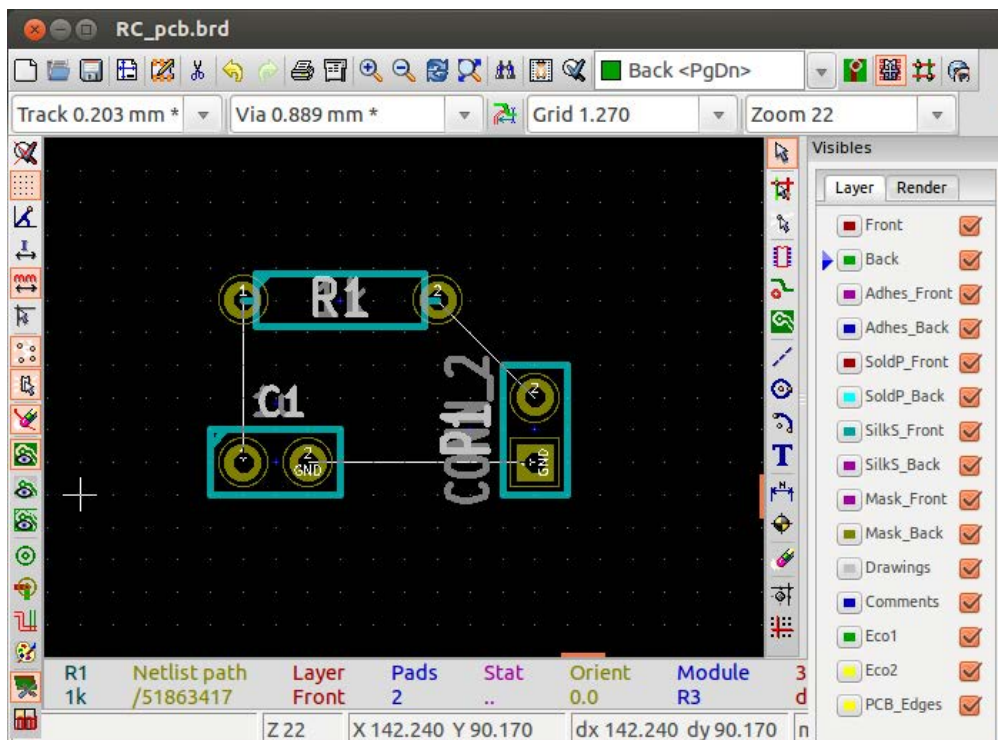


Figure 11. Final Placement of Footprints of the RC Circuit

them at the center of the editor window. Figure 10 shows the footprint modules of the RC circuit placed at the center of the layout editor window.

Arrange the components to minimize track length. Specify design rules like Track Width, Via Diameter and so on using the Design Rules menu. Choose the PCB layer from the layer options on the right and start drawing tracks. To draw a track, use x the shortcut key. Figure 11 shows the final placement of the components, and Figure 12 shows the tracks drawn. This is a

single-layer PCB layout, with a track width of 0.8mm. Complete the layout by drawing PCB edges.

You can create multi-layered PCB layouts using the Layout Editor. You can create your own footprint modules too. You also can get a 3-D view of the layout you created. Figure 13 shows the 3-D view of the PCB layout created above.

Do a design rules check by clicking on the Perform design rules check option from the menu bar. Ensure that there are no errors, and save your board. You can choose to generate Gerber files for

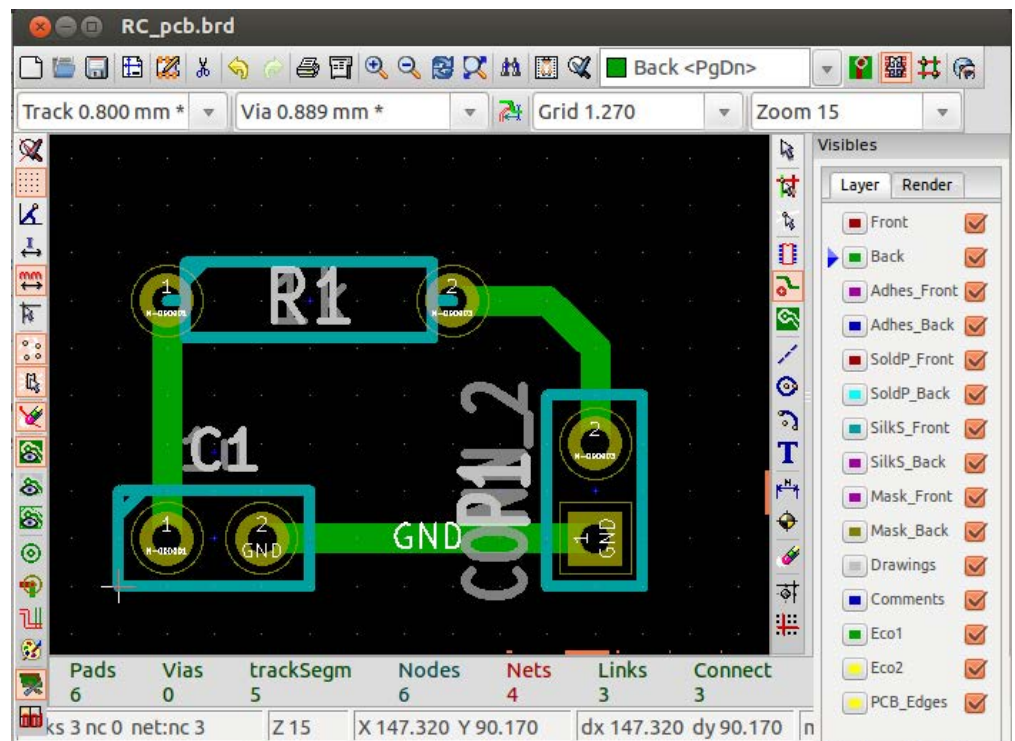


Figure 12. All Tracks Drawn, Single-Layer PCB Layout

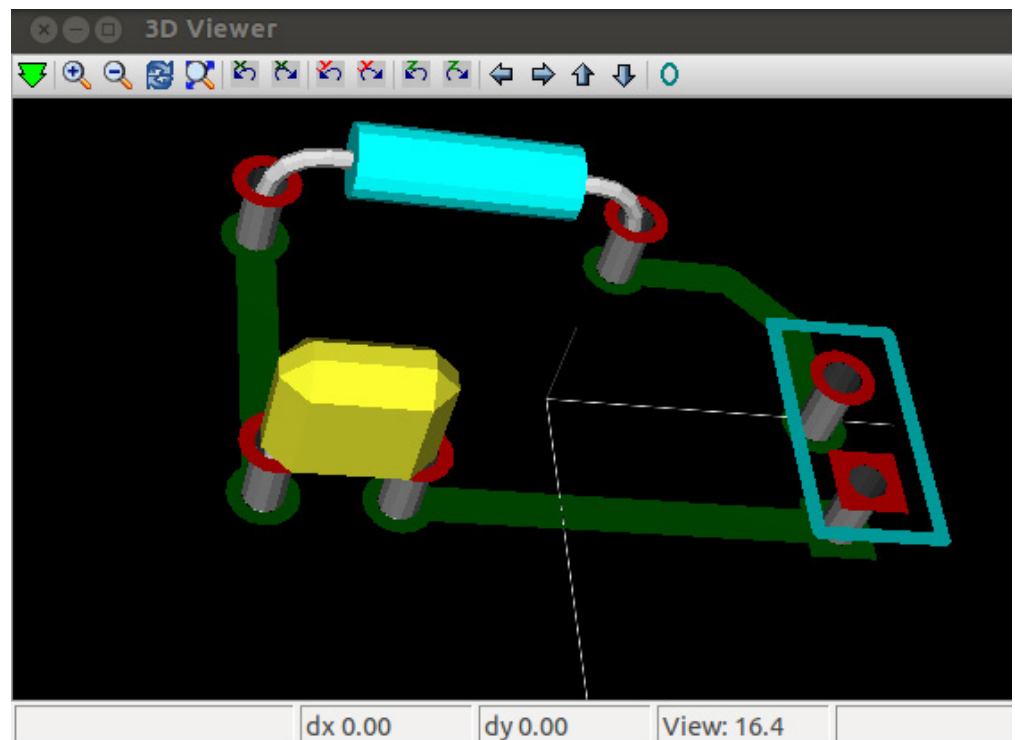


Figure 13. 3-D View of the RC Circuit PCB Layout

your PCB. You also can get DXF/HPGL/PostScript plots of your PCB layout. Now you are ready to fabricate your PCB!

Advanced Features in Oscad

Oscad has the following advanced features:

- Model Builder
- Subcircuit Builder

■ Scilab-based simulator

Model Builder The accuracy of the simulation depends on the accuracy of the models used. In Oscad, the user can define and edit device models. Currently, this feature incorporates models for diodes, BJTs, MOSFETs and so on. To illustrate the Model Builder's features, let's look at the example of a Bridge Rectifier circuit using diodes. Figure 14 shows

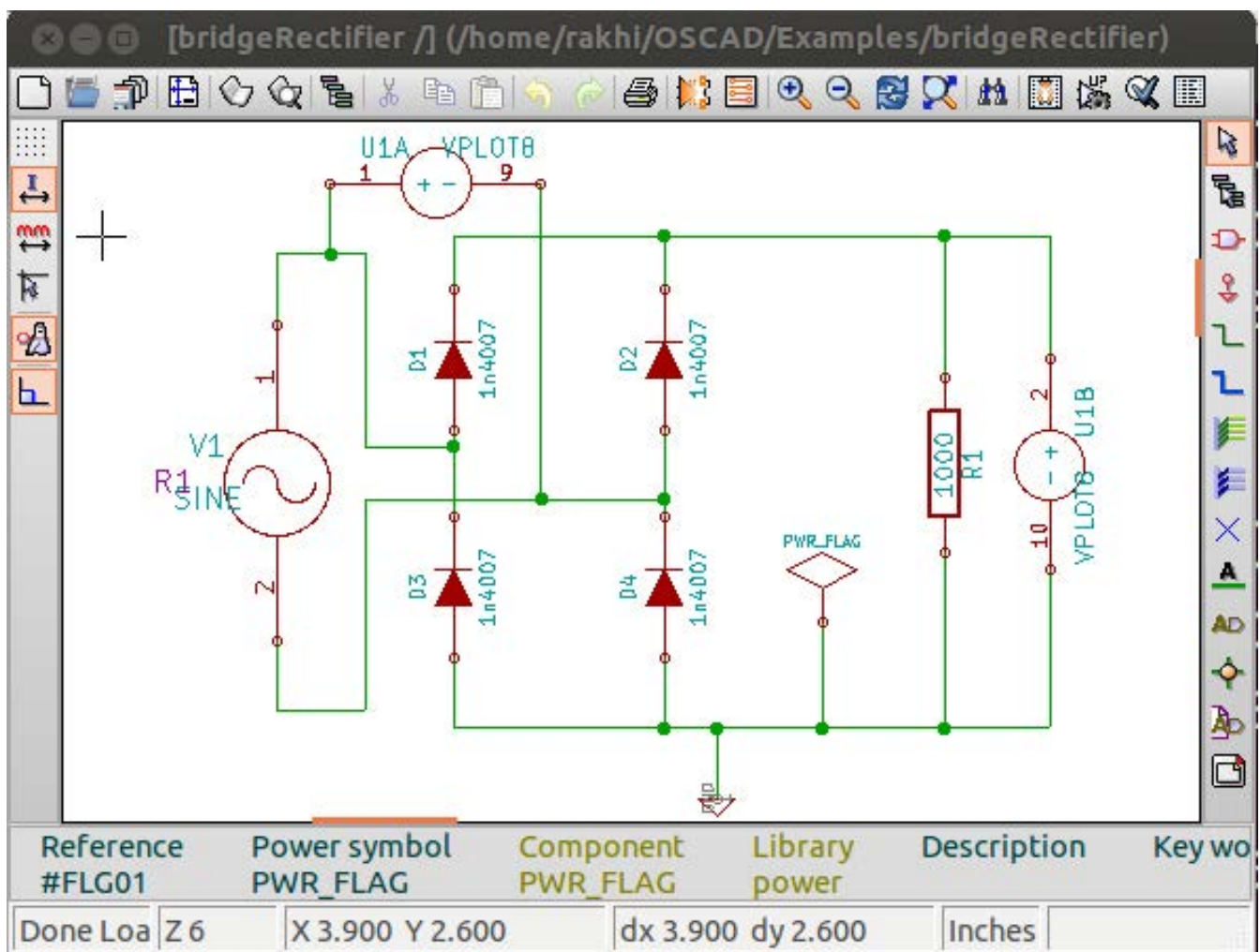


Figure 14. Bridge Rectifier Circuit Using Diodes

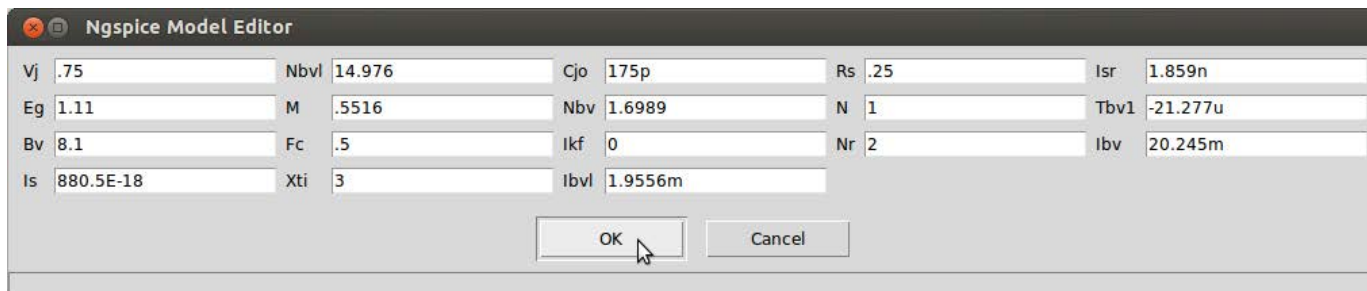


Figure 15. GUI to Edit the Diode Model

the circuit.

Launch the Model Builder tool to edit the diode model. Figure 15 shows the GUI to key in the diode's parameters. You can enter the model parameters as given in a datasheet of your device. Save the

model. You can re-use this model in other projects using the Export option (Chapter 8 of our *Oscad* book explains the Model Builder in more detail).

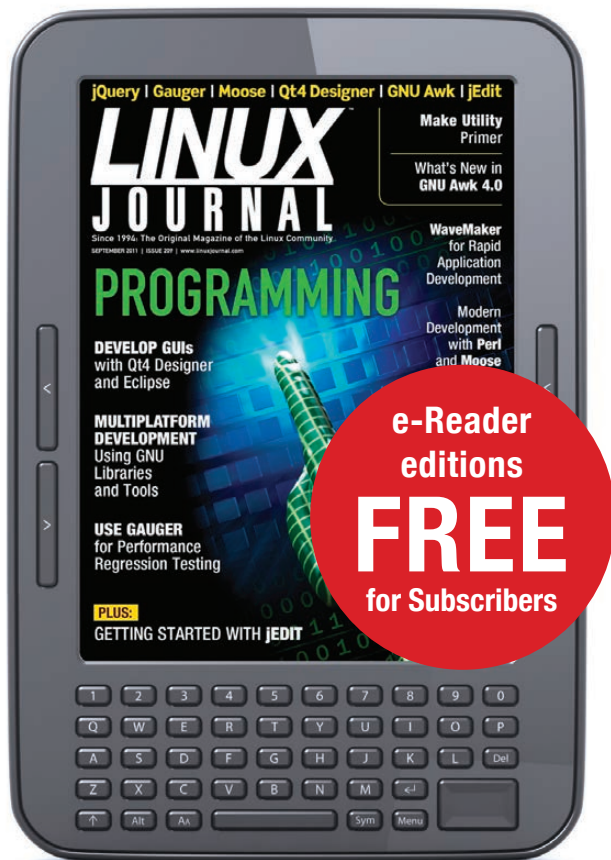
Subcircuit Builder The Subcircuit Builder helps you break complex

LINUX JOURNAL

on your
e-Reader

Customized
Kindle and Nook
editions
now available

LEARN MORE



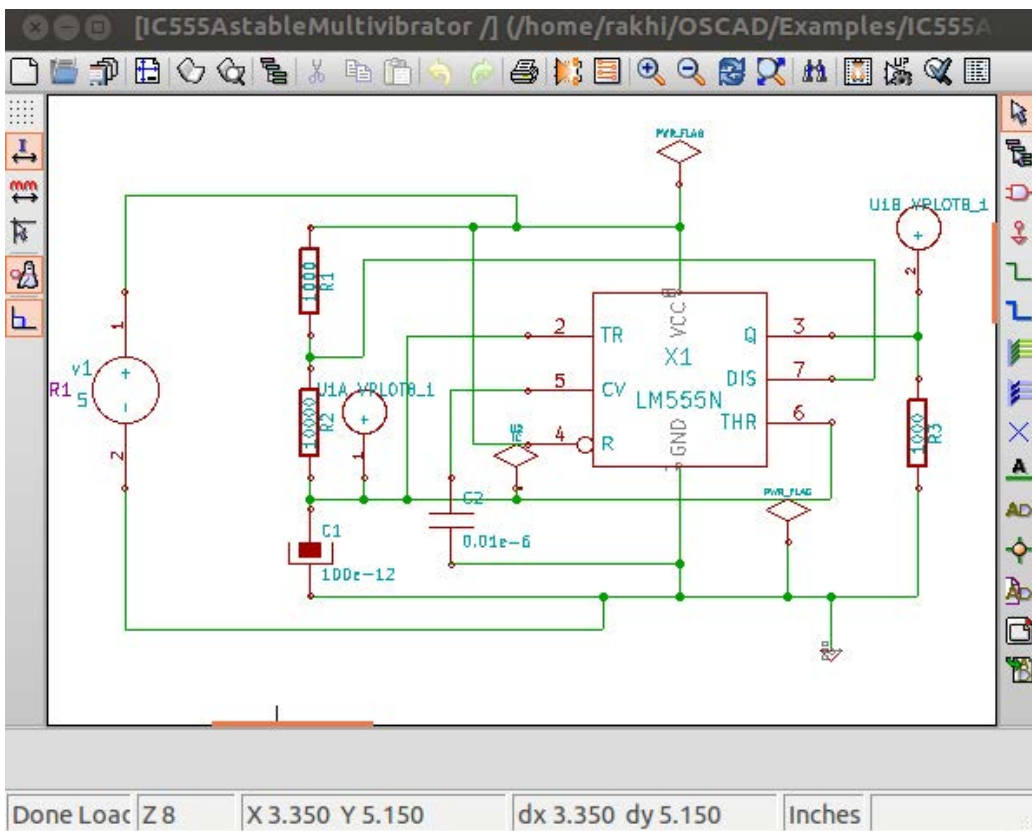


Figure 16. Astable Multivibrator Circuit Using a 555 Timer

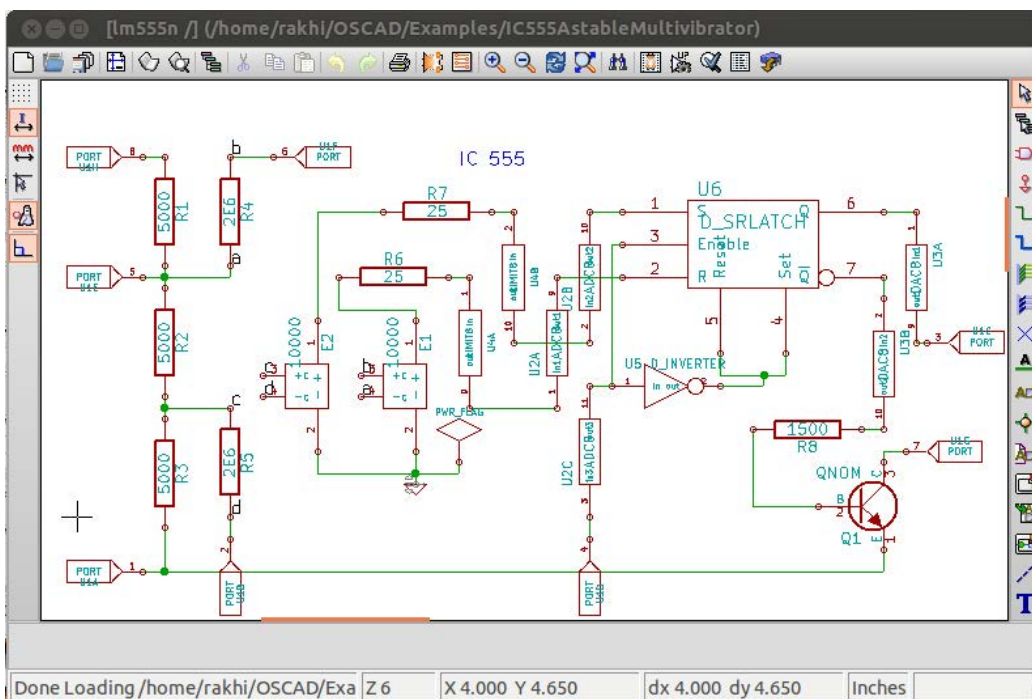


Figure 17. LM555N Subcircuit

circuits into subcircuits. These subcircuits can be re-used in other projects. This approach makes designs scalable, hierarchical and modular. Let's look at the example of the Astable Multivibrator that uses a 555 timer (LM555N). Figure 16 shows the Astable Multivibrator circuit. To develop the LM555N subcircuit, launch the Subcircuit Builder tool and create the internal schematic for LM555N. Figure 17 shows the

subcircuit of LM555N. Note the port connections.

The OScad subcircuit library already has subcircuits for LM555N, UA741 and so on. (See Chapter 8 of our *Oscad* book for more details on the Subcircuit Builder.)

Scilab-Based Mini Circuit Simulator (SMCSim) Simulators use mathematical models to model the behavior of electronic circuits. Unfortunately, no simulator gives the system of equations it solves. This impedes understanding of how a circuit simulator works. SMCSim provides mathematical equations

that replicate the behavior of the circuit. This feature is unique to OScad. It gives users more insights into their circuits and helps them appreciate the simulation results.

SMCSim works in three modes: normal, symbolic and numerical. In normal mode, SMCSim solves the circuit equations and gives the final simulation result. In symbolic mode, it gives symbolic equations along with the result. In numerical mode, it gives symbolic equations, intermediate numerical values of the components and elements in system matrices, and the final

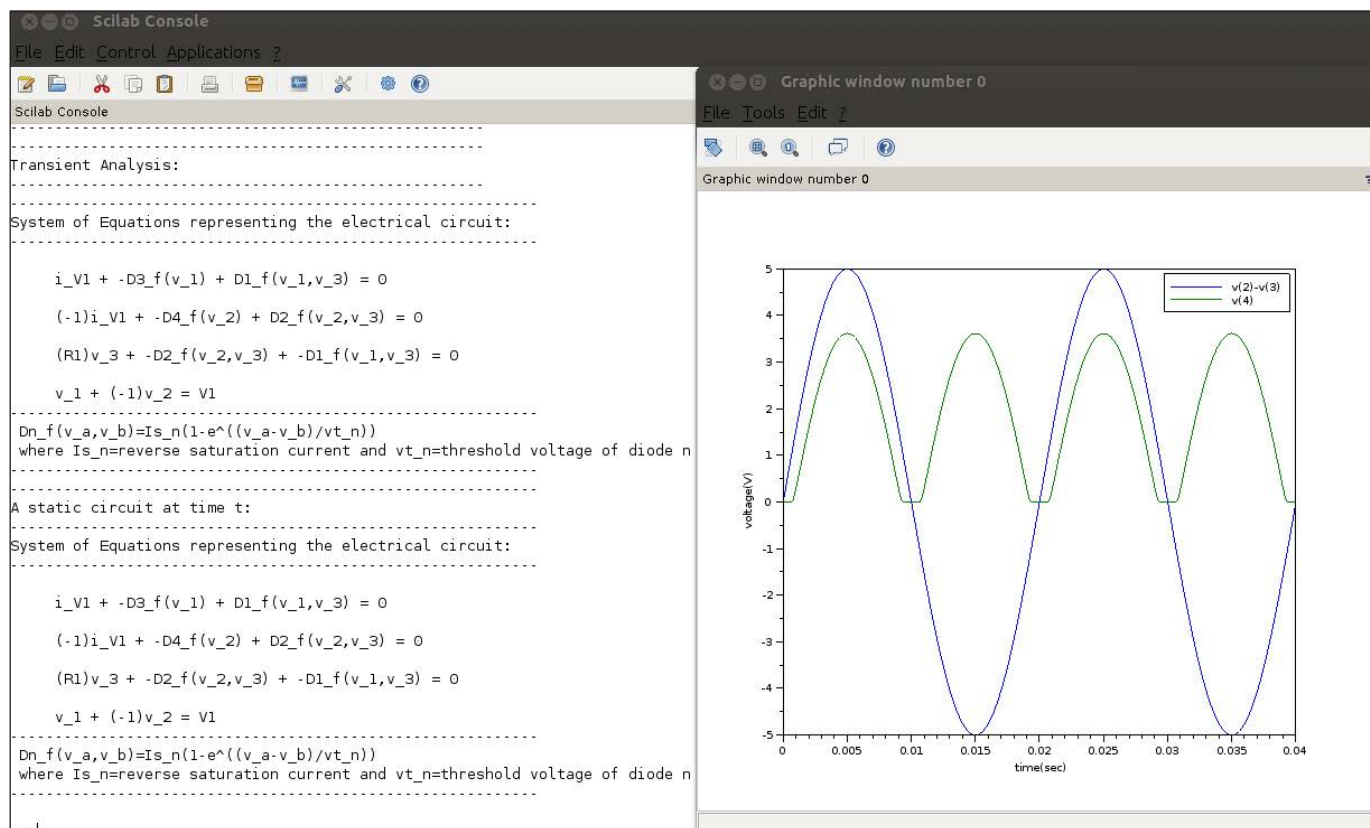


Figure 18. Equations for Bridge Rectifier Circuit and SMCSim Simulation Results

simulation result. Figure 18 shows the equations generated for the Bridge Rectifier circuit presented earlier. It also shows the SMCSim simulation results. (Chapter 9 of our *Oscad* book explains more about SMCSim.)

Oscad on Aakash, the World's Lowest-Cost Tablet

Aakash (<http://aakashlabs.org>)

provides a great platform for learning and education. It can run GNU/Linux and most GNU/Linux applications (http://aakashlabs.org/builds/linux-on-aakash_CSI_July03.pdf). As KiCad, Ngspice, Scilab and Python run on GNU/Linux, you easily could port Oscad also onto the Aakash. As the Aakash tablet costs about \$35, for less than \$40 (including a keyboard and a mouse), you can have access

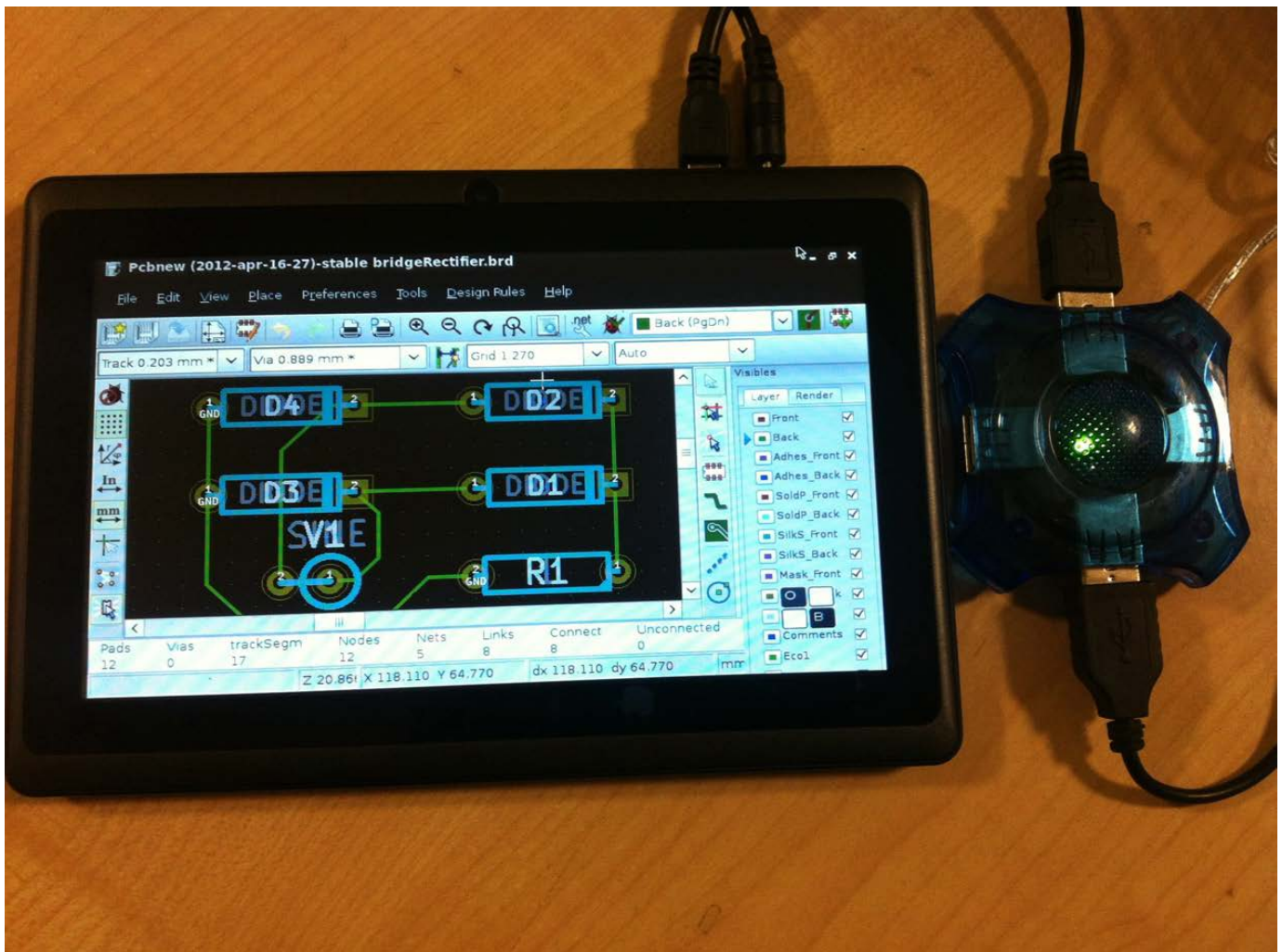


Figure 19. PCB layout of Bridge Rectifier on Aakash. A USB hub is used with Aakash to connect a mouse and a keyboard.

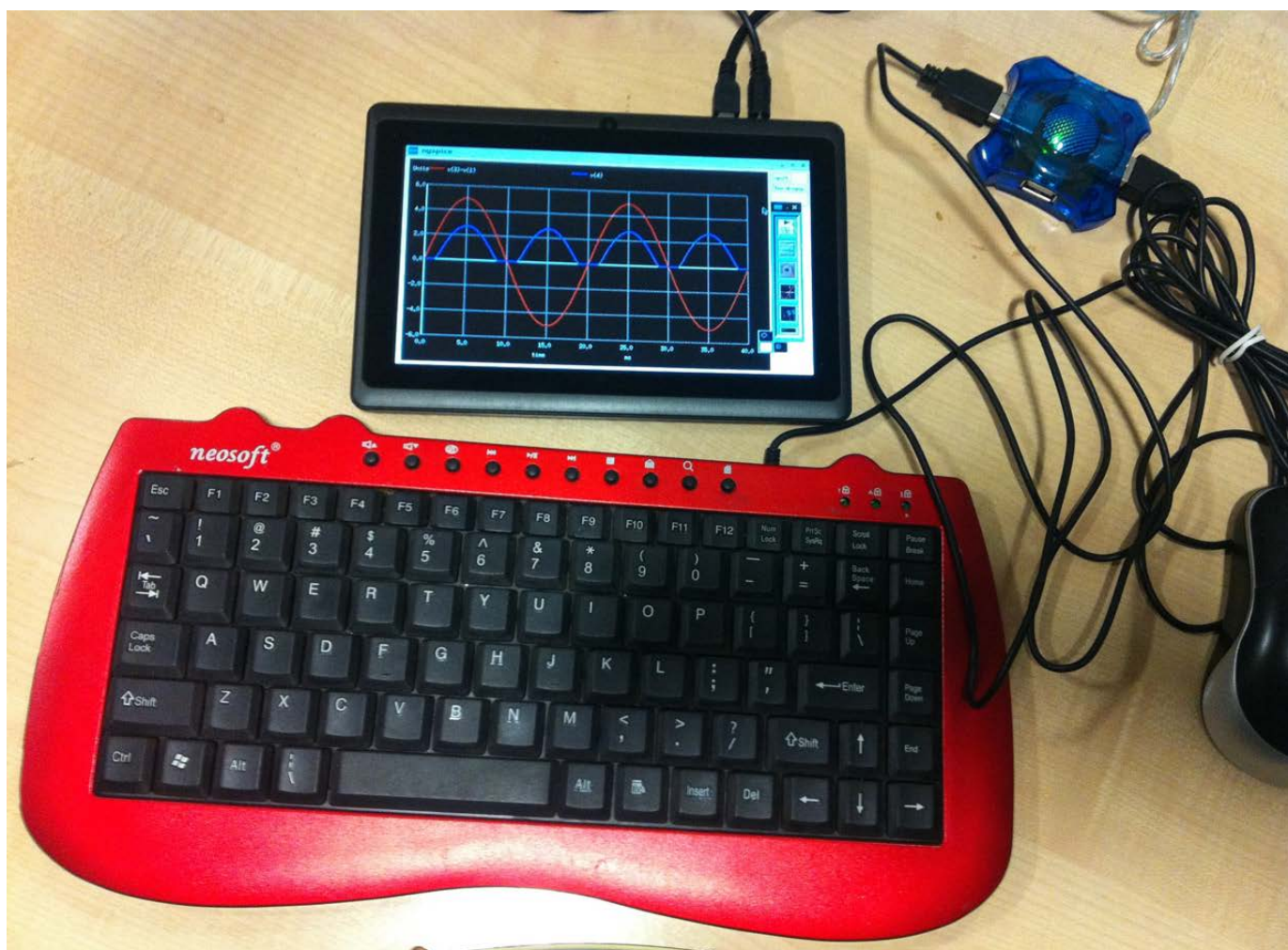


Figure 20. Oscad running on Aakash with an external keyboard and mouse attached. It also shows output from Ngspice.

to a powerful CAD system. This can help students who are enthusiastic about circuit design but can't afford expensive hardware and software.

Installing Oscad on Aakash is similar to its GNU/Linux desktop version. You can follow the same instructions for the desktop version mentioned earlier in this article. The benefit of using Oscad on Aakash lies in its portability and performance. Although

a capacitive touchscreen is available on Aakash, you can attach an external keyboard and mouse. Sometimes keyboard shortcuts are easier to use than touch controls when running design and simulation tools. Figures 19 and 20 show a few photographs of Oscad running on Aakash.

Tutorials and Documentation

Three Spoken Tutorials that

explain installing and using Oscad are available. As Oscad uses KiCad and Ngspice, two different sets of tutorials on each of these also are available. You can download all the tutorials from <http://www.spoken-tutorial.org>. We recently published a book on Oscad titled *Oscad: An Open Source EDA Tool for Circuit Design, Simulation, Analysis and PCB Design* published by Shroff Publishers and Distributors, June 2013 (<http://www.shroffpublishers.com/detail.aspx?cat=0&title=5687>). An e-version of this book is available free of charge at <http://www.oscad.in/resource/book/oscad.pdf> for non-commercial use. Lots of example projects are also available at <http://www.oscad.in/downloads> to help you get started.

Oscad Activities

We have a host of activities through which we promote the use of Oscad:

- SELF workshops through Spoken Tutorials: these are two-hour workshops conducted using Spoken Tutorials and an associated instruction sheet. An institution that wants to organize a SELF workshop should provide the following: 1) A volunteer (who can be a faculty member or a student), 2) one computer per participant with audio output capability, and 3) one head-phone per participant. We get very encouraging feedback from SELF workshops conducted all over India.
- Textbook Companion on Oscad: an Oscad Textbook Companion (OTC) provides Oscad-based circuit schematics, simulation results and PCB design for solved problems from standard textbooks on electronic circuits. OTCs are available for *Microelectronic Circuits: Theory and Applications* 5th edition, Adel S. Sedra, Kenneth C. Smith, Oxford University Press (2009); and *Electronic Devices and Circuit Theory* 10th edition, Louis Nashelsky, Robert L. Boylestad, Pearson (2009). You can download them free of charge from http://www.oscad.in/textbook_run. OTCs for three more textbooks are currently in progress (http://www.oscad.in/books_progress). We invite students, teachers and professionals to create OTCs for other standard textbooks as well. OTC not only creates a large pool of useful projects in Oscad,



DOC SEARLS

A Cool Project for Microsoft: Adopt Linux

Embracing Linux would be good for Microsoft, good for Linux and good for the world.

“Do you know Linux? WE ARE HIRING!” That’s what billboards from HostGator (<http://www.hostgator.com>) have been saying for the past several years. That company is not alone. Demand for Linux talent is high and getting higher. In a February 2014 report (http://marketing.dice.com/pdf/LinuxJobsReport_2014.pdf), the Linux Foundation and Dice detailed a picture that was already obvious:

The explosive demand for Linux talent is intensifying. Seventy seven percent of hiring managers have “hiring Linux talent” on their list of priorities for 2014, up from 70 percent a year ago. More than nine in ten hiring managers plan to bring Linux professionals on board in the

next six months. Furthermore, 46 percent plan to boost their hiring of Linux pros in 2014, a 3-point increase over 2012.

And that’s not all:

Linux professionals know they’re a hot item, and they’re fully aware of how their skills have contributed to their success in the workplace. In fact, a full 86 percent of survey respondents said that knowing Linux has advanced their career opportunities. But when asked why they sought out a Linux career in the first place, only 17 percent ranked money and perks highest. Instead, 51 percent cited their passion for Linux, and 64 percent wanted to work with Linux because

of its pervasiveness in modern-day technology infrastructure.

Exactly the same could have been said of DOS in 1985 and Windows a decade later. It cannot be said of either today, even though both still are used all over the place. In 2012, for example, United Airlines moved its reservation system to SHARES (<http://servicesangle.com/blog/2012/03/08/united-airlines-goes-from-the-web-to-dos>), which runs on DOS. And Microsoft's sundowning of Windows XP (<http://www.microsoft.com/en-us/windows/enterprise/endofsupport.aspx>) seems to have given cramps to nearly the entire retail banking industry (<http://www.theverge.com/2014/1/20/5326772/windows-xp-powers-95-percent-of-atms-worldwide>), which finally will move its ATM machines from XP to Windows 7 (<http://www.businessweek.com/articles/2014-01-16/atms-face-deadline-to-upgrade-from-windows-xp#p2>) or to Linux (<http://www.atmmarketplace.com/article/129594/Brazilian-banks-look-to-Linux-for-ATMs>). It's not going to Windows 8, which Supersite for Windows (<http://winsupersite.com>) guru Paul Thurrot calls "a debacle"

(<http://winsupersite.com/windows-8/threshold-be-called-windows-9-ship-april-2015>) and says is "tanking harder than Microsoft is comfortable discussing in public". He also says the next version of Windows, code-named Threshold, "recasts Windows 8 as the next Vista", adding "there's no way to sugarcoat this. Windows 8 has set back Microsoft, and Windows, by years, and possibly for good."

According to Ali R. Babaoglan, who, presumably, pays to read Gartner's expensive reports (<http://www.alibabaoglan.com/blog/gartners-technology-predictions-2014-2015-2016>):

90 percent of enterprises will bypass broad-scale deployment of Windows 8. Gartner claims that most enterprises and their trusted management vendors are not yet prepared for the change to Windows 8, and says enterprises will want to wait for more stability before proceeding. While Microsoft as a technology company can make these changes at a more advanced pace, the market will take time to mature, and most enterprises will sit on the sideline for now.

In today's world, the closest thing to Windows' longtime domination

In today's world, the closest thing to Windows' longtime domination of PCs is Android's domination of mobile devices.

of PCs is Android's domination of mobile devices. There the only thing keeping Android from prevailing as utterly as Microsoft once did over PCs is Apple devices running iOS. According to comScore's January 2014 report (https://www.comscore.com/Insights/Press_Releases/2014/3/comScore_Reports_January_2014_US_Smartphone_Subscriber_Market_Share), Android led smartphone platforms in the US with a 51.7% share, followed by Apple (iOS) with 41.6%, then Microsoft (Windows Phone, http://en.wikipedia.org/wiki/Windows_Phone) with 3.2% and BlackBerry with 3.1%. More telling is the number of OEMs making phones for each operating system. Apple and BlackBerry have just one each: themselves. Microsoft has HTC, Nokia and Samsung. Android has HTC, Samsung and all the rest (http://en.wikipedia.org/wiki/Comparison_of_Android_devices), which are too many to list. According to Sundar Pichai (http://en.wikipedia.org/wiki/Sundar_Pichai), the Senior VP who oversees apps at Google, more than a billion devices running

Android have been activated (<https://plus.google.com/+SundarPichai/posts/NeBW7AjT1QM>).

Microsoft would be far wiser to join that juggernaut than to fight it. And, if it does, here are a few things Microsoft could bring to the Linux/Android table that nobody else will, including Google:

1. **Microsoft is a personal computing company.** It started with the gleam that appeared in Bill Gates' eye when he saw the MITS Altair, and it's still there. Google, on the other hand, is a server computing company. Servers gleamed in Larry Page's and Sergey Brin's eyes when they first put Google Search atop a pile of Linux boxes, and it's still there too. One thing companies can't change is where they come from, and Microsoft has a huge legacy advantage on the personal front.
2. **I mean, really personal.** The customers Microsoft cares most about are individual human beings, not just B2B corporate ones.

Microsoft has vast call centers with well-trained people whose job is to solve individual customers' problems. Got a problem with Gmail or Google Maps? Try calling somebody. It soon will be clear that Google would rather not talk to any users personally, much less operate a call center. (A high-level Google executive once told me that Google wouldn't operate call centers for ordinary folk because that would—no kidding—result in revenues of less than \$1 million per employee.)

3. **Linux could use a personal touch.**

Depending on the kindness of geek friends and strangers isn't enough, and never has been. It should be clear by now that Linux could use a first-rank tech giant willing to get personal with individual customers. The old usual suspects—IBM, SAP, Oracle, HP and so on—are B2B companies at heart and never have been comfortable dealing with the creatures they call "end users".

4. **Microsoft would be in a buyer's market for talent.**

Perhaps the most brilliant thing Apple ever did was give Macintosh obsessives a place to work and a "genius" title. I would bet there are far more Linux obsessives who deserve the

same title, and would be glad to help Microsoft make the most of Linux, and to help both customers and users do the same.

5. **Microsoft and Linux DNA already are beginning to mix.**

Exhibit A: the Microsoft/SUSE Alliance

(<https://www.moreinterop.com>).

Yes, it's a B2B one (in compliance with demand by countless businesses for mixed Linux/Windows solutions), but it could easily expand into B2C as well.

6. **Microsoft + Linux would give IT a new lease on life.**

For the past several years, IT budgets have been shrinking and/or shifting (<http://chiefmartec.com/2011/12/follow-the-money-from-it-to-marketing>) over to marketing and other departments. One big reason is that Windows is moribund while Linux's strength has been in servers and embedded (especially mobile). By embracing Linux for desktops and laptops, Microsoft would cover the entire non-Apple desktop, laptop and mobile waterfront inside the enterprise.

7. **Microsoft + Linux would create a whole new hardware OEM game.**

Back in the Windows 95, 98, NT and XP days, when Microsoft said

Remember how Bill Gates didn't get the Internet at first, and then turned a 180 in 1995?

"Jump!" every desktop and laptop OEM would say "How high?" Then Bill Gates left, Windows got lame, Apple got hot, and mobile got hotter. As a result, Windows hardware OEMs have been bailing and failing left and right. IBM unloaded its PC hardware business on Lenovo in 2005 (http://en.wikipedia.org/wiki/ThinkPad#Acquisition_by_Lenovo). HP announced it was folding its PC business in 2011 (http://money.cnn.com/2011/08/18/technology/hp_pc_spinoff/index.htm?iid=EL), when it still had the market lead. (It reversed the decision later, but the damage was done.) Then Dell fell into so much trouble that it made moves to go private (with help from Microsoft) a few months ago (<http://www.reuters.com/article/2013/02/05/us-dell-buyout-idUSBRE9140NF20130205>). In all three cases, the direction was away from personal devices and toward corporate service offerings. For all those reasons and more, Microsoft became its own OEM with Surface tablets in 2012 (<http://www.microsoft.com/en-us/news/press/2012/jun12/06-18announce.aspx>). Ever used

one? They're pretty slick.

Remember how Bill Gates didn't get the Internet at first, and then turned a 180 in 1995? Read the memo (<http://www.wired.com/thisdayintech/2010/05/0526bill-gates-internet-memo/all>). Linux wasn't on Bill's radar at the time, but BSD was. The top item under "actions required for the Windows platform" begins this way:

1. Server. BSD is working on offering the best Internet server as an integrated package. We need to understand how to make NT boxes the highest performance HTTP servers.

Today the reality of Linux is of a piece with the reality of the Internet. Neither is going away. Both are co-evolving in the minds of every geek adding value to them. Both transcend the interests of every company contributing to them, including Google. If Satya Nadella (http://en.wikipedia.org/wiki/Satya_Nadella) looks at reality with the same clear eyes Bill Gates cast

on the Internet in 1995, he might see the wisdom of embracing Linux with the same enthusiasm and commitment.

How would Microsoft do that, exactly? It could start by joining the Linux Foundation (<http://www.linuxfoundation.org>) and perhaps by purchasing Attachmate (<http://www.attachmate.com>), which owns SUSE, plus what's left of Novell (<http://www.zdnet.com/blog/open-source/attachmate-reveals-novell-suse-and-linux-plans/8771>). But I think the easiest route would be to *open source the intention*. Nadella could start by announcing a round of hiring toward harmonizing Linux and Windows, and in the process ask the geek world what Microsoft should do, specifically. I guarantee that the answers will be better than any one of us alone can imagine. And that will be good for Microsoft, Linux and everybody who depends on both. ■

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.



Send comments or feedback via
<http://www.linuxjournal.com/contact>
or to ljeditor@linuxjournal.com.

Advertiser Index

Thank you as always for supporting our advertisers by buying their products!

ADVERTISER	URL	PAGE #
AnDevCon	http://www.andevcon.com/	7
DrupalCon Austin	http://austin2014.drupal.org	45
Drupalize.me	http://www.drupalize.me	120
EmperorLinux	http://www.emperorlinux.com	15
Silicon Mechanics	http://www.siliconmechanics.com	3
Texas Linux Fest	http://texaslinuxfest.org	2
Women in Tech	http://www.witi.com/center/conferences/2014/summit/	83

ATTENTION ADVERTISERS

The *Linux Journal* brand's following has grown to a monthly readership nearly one million strong. Encompassing the magazine, Web site, newsletters and much more, *Linux Journal* offers the ideal content environment to help you reach your marketing objectives. For more information, please visit <http://www.linuxjournal.com/advertising>.

drupalize.me

Instant Access to Premium Online Drupal Training

- ✓ *Instant access to hundreds of hours of Drupal training with new videos added every week!*
- ✓ *Learn from industry experts with real world experience building high profile sites*
- ✓ *Learn on the go wherever you are with apps for iOS, Android & Roku*
- ✓ *We also offer group accounts. Give your whole team access at a discounted rate!*

Learn about our latest video releases and offers first by following us on Facebook and Twitter (@drupalizeme)!

Go to <http://drupalize.me> and get Drupalized today!

