Open Source in Embedded Systems

Andrew Greenberg

September 24, 2008

Me

- Directory of Development
  - The TOVA Company (http://www.tovatest.com/)
  - Medical software/hardware to diagnose ADHD
- Associate
  - PSU Electrical and Computer Engineering Department
  - Teach workshops on technology.
  - Portland State Aerospace Society (http://psas.pdx.edu/)
- Embedded Systems Consultant
  - Aerospace and biomedical projects
  - http://www.embeddedmoose.com/
- Masters and bachelors in EE from PSU and Reed College

You

- How many of you:
  - Are programmers?
  - Are hardware designers?
  - Work in embedded systems?
  - Currently use open source?
  - Currently run Linux?

Open Source Technologies

- Open, as in "affording free passage or access".
- Source, as in "the place where something begins".

Source != *.c

- Most of us think of open source as software, but that's too narrow:
  - Programmable logic: VHDL files.
  - Electrical: Schematic capture and PCB layout files.
  - Mechanical: CAD files.
- Open source software is easier and more ubiquitous (hardware is sadly not free).

Open Source Projects

- I'm going to define an open source project as:
  - Any project that has freely available source documents.
  - Can't have NDAs.
  - Has an open source license.
- By this definition, there are TENS OF THOUSANDS of open source projects.

It's All About the License

- No license = legal limbo. Don't go there.
- Many different flavors of licenses.

The BSD Open Source Licenses

- BSD (Berkeley Software Distribution) License, Apache License, etc.
- Synopsis: "Here's the source, don't sue us."
- Pro: You can do whatever you want with the source. Really.
- Con: May not encourage community development of source.

The General Public Licenses (GPL) Open Source License

- Free Software Foundation's GPL, Mozilla License, etc.
- Synopsis: "The source is owned by the community."
- Pro: Stunningly rich open source projects.
- Con: Tricky legal ramifications:
  - Must pass along project's source to your end users.
  - Changes to the project's source must be passed along, too.
  - YOUR code, if mixed with the GPL'd code, must be passed along as well.
  - IANAL.

What makes Open Source Projects Tick?

- Passionate, highly motivated ~~geeks~~ individuals.
- Projects centered around technologies, not customers.
- More and more are being funded, most are labors of love.
- Characterized by:
  - Duplicate projects ("we're better!")
  - Rapid changes ("new! better! incompatible!")
  - Shockingly fast development cycles (6 months)
  - Quality and maturity change per project.

How are Open Source Projects Organized?

- Around Collaboration tools
  - Mailing lists
  - Wikis
  - Source management systems
  - Conferences
- Developers/Engineers:
  - Single author
  - Small handful of active contributors
  - Industry or foundation backed

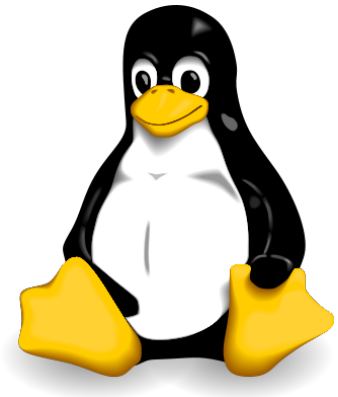
What makes Open Source Projects Good Business?

- Pros:
  - Rapid community response.
  - Consultants available.
  - Some best of breed technologies.
  - Free = leverage for small companies.
  - You can fix it yourself.
  - Features exist/issues fixed rapidly (6 months!)
- Cons:
  - Variable quality.
  - Many too immature.
  - Many are dead.
  - Many have too many bugs.

Qualifying Open Source Projects for Business

- Your business needs criticality
  - Want vs need vs critical
- Activity level
  - Contributions/downloads over time
- Community size
  - How many authors? Project managers? Backed by?
- Maturity
  - Been around for a while, or brand new?
- Status
  - Pre-alpha or released? Version number?

Example: Linux



- Actually, a terrible example.
- Too many independent projects lumped together:
  - Linux kernel (kernel.org)
  - X windows manager and video drivers (x.org)
  - Gnome (gnome.org) or KDE (kde.org) desktop managers
  - BASH shell, utilities (fsf.org)
  - Apache web server (apache.org)

Example: NeoFreerunner Cell Phone by OpenMoko.com

- Open source phone
  - Software is all open source (in this case, Linux-based).
  - Schematic and layout files are open source.
  - You have the right to modify and change the hardware and software.
  - You'd be insane to build it yourself.

Giving back some of what you get

- There's always give and take with the community
- How can you give back?
  - Just using the software is good
  - File bug reports
  - "Say it in cash"
    - Sponsor an OSS project (tax deductible?)
    - Hire OSS developers to fix/add features
    - Buy them hardware to test on
  - Advertise the fact that you support OS projects.

Everyone hates infrastructure

- Desktop operating systems.
- Server operating system.
- Project management tools.

Open source desktop OS

- GNU/Linux
  - Debian distributions (Debian, Ubuntu).
  - Red Hat distributions (Red Hat, openSUSE, Mandrake).
  - Others (Gentoo, Slackware, Arch).
- FreeBSD and NetBSD
- OpenSolaris
- ReactOS
- FreeDOS
- Plan 9

Linux as desktop OS

- In my opinion, Linux is ready for general use.
  - Network-based installers usually just work.
  - Desktops exceed Windows functionality, and are pretty.
  - 23,000+ "bundled" apps in Debian Linux.
- Hints:
  - Try to use sane hardware, or well used (e.g., Thinkpads).
  - Parallel play for a while: linux and Windows.
  - Try a community-based distro: Ubuntu, Debian, openSUSE.
  - Find a user's group. They'll actually help you.
  - Find a guru. Possibly, pay them.
- Demo!

Linux in a Windows world

- Cross-over office: For $40, run IE and MS Office 2003 on Linux.
- KVM is a x86 emulator (like vmware).
  - Runs any flavor of Windows just fine.
  - Has hardware support (USB).
  - Particularly good for USB JTAG programmers.
- Demo!

Linux as server OS

- Hard to configure right, but not terrible.
- A scary amount of services.
- Terribly secure, if done right.
- After a while, you'll want one too!
  - Offsite backups.
  - Services (email).
  - Accessible data caches.
  - Secure gateways (VPNs).
  - Compile, test and monitor embedded devices.
  - Project collaboration.
- Demo! (if network available)

Real Infrastructure

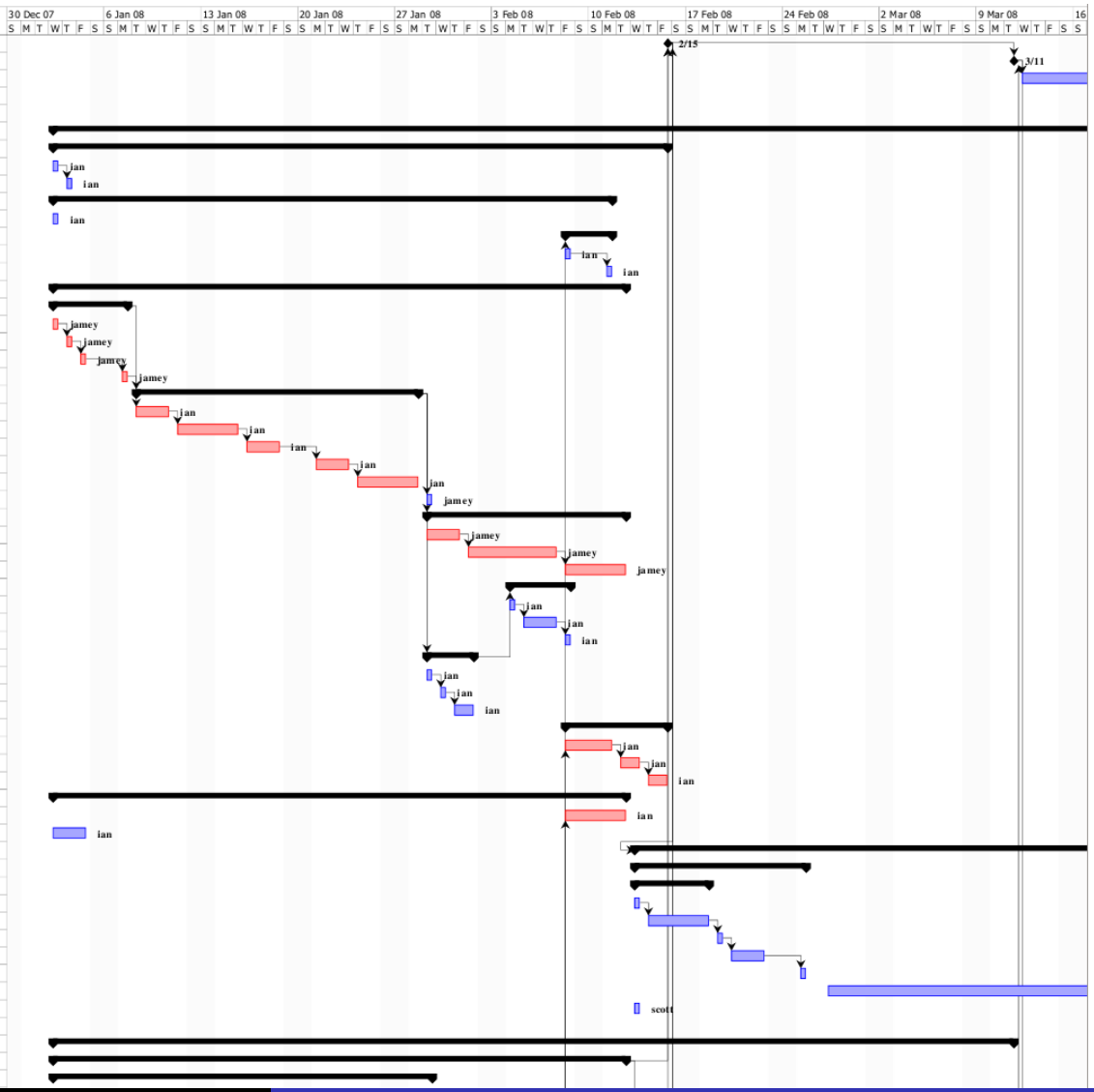
- Mailing lists (mailman).
- Wikis (ikiwiki,trac).
- Source repositories (git,subversion).
- Issue trackers (trac,bugzilla).
- After a while, you'll want one too!
- Demo! (if network available)

Project Management

- TaskJuggler.
- GanntProject.
- OpenProj.
- dotproject (web based).

OpenProj

	Name	Duration	Resource N...	Predecessors
1	Alpha ready (feature Complete Prototype)	0 days		6;57;79
2	Beta ready (10 - 100 units)	0 days		109;1
3	Beta testing	30 days?		2
4	Ship it!	0 days		3;119
5	**Java User Interface**	**55 days?**		
6	**Alpha**	**33 days?**		
7	Browse subjects and sessions	1 day?	ian	
8	New Subject interface	1 day?	ian	7
9	**New Session interface**	**29 days?**		
10	To Linux PTE (CD and on-disk)	1 day?	ian	
11	**To USB dev**	**2 days?**		**79**
12	Send test format to USB device	1 day?	ian	
13	Receive test data from USB device (--test credits)	1 day?	ian	12
14	**Report Generator**	**30 days?**		
15	**Analysis code**	**4 days?**		
16	Summary of raw scores	1 day?	jamey	
17	Analysis (norms)	1 day?	jamey	16
18	Misc analysis?	1 day?	jamey	17
19	Narrative generator	1 day?	jamey	18
20	**Report pages**	**15 days?**		**15**
21	Page 1 - Summary	3 days?	ian	19
22	Page 2 - Analysis data	3 days?	ian	21
23	Page 3 - Test and Symptom validity	3 days?	ian	22
24	Page 4 - Suplemental data	3 days?	ian	23
25	Intra-test comparison	3 days?	ian	24
26	Report previewer	1 day?	jamey	20
27	**Save report**	**11 days?**		**20**
28	to PDF	3 days?	jamey	
29	to RTF	5 days?	jamey	28
30	To HTML	3 days?	jamey	29
31	**Export Files**	**5 days?**		**35**
32	Zipped XML (TOVA 8)	1 day?	ian	
33	CSV	3 days?	ian	32
34	Hidden: TOVA 7	1 day?	ian	33
35	**Import Files**	**4 days?**		**20**
36	TOVA 7 files import wizard	1 day?	ian	
37	TOVA 8 XML files importer	1 day?	ian	36
38	Deduplication interface	2 days	ian	37
39	**Order test credits**	**6 days?**		
40	Query test credit information	2 days?	ian	104
41	Generate and send test credit requrest (w/coupon?)	2 days?	ian	40
42	Receive and accept test credit authorzation	2 days?	ian	41
43	**Help**	**30 days?**		
44	USB device diagnostics	3 days?	ian	79
45	Software update UI	3 days?	ian	
46	**Beta**	**25 days?**		**1**
47	**Preferences**	**9 days?**		
48	**HIPAA options**	**4 days?**		
49	Name, subject number display	1 day?		
50	Loose vs strict search	3 days?		49
51	Test format	1 day?		50
52	Report	3 days?		51
53	Custom Subject fields	1 day?		52
54	Online Help	15 days?	maria	
55	Contact Info	1 day?	scott	
56	**Windows Package**	**50 days?**		
57	**Alpha**	**30 days?**		
58	**Kexec driver**	**20 days?**		



Office tools

- Office: OpenOffice.
- Web: Firefox.
- Mail: Thunderbird or Evolution.
- Layout: inkscape and/or LaTeX.
  - This presentation is done using LaTeX Beamer.

Software tools

- Equal to, or better, than commercial tools
- Compilers linkers, utilities
- Integrated Development Environment (IDE)
- Debuggers
- Programmers
- And more!

GNU Toolchain: open source software development tools

- GNU = GNU's Not Unix: open source Unix code.
- Really the *nix in Linux: Technically it's "GNU/Linux".
- Provides build tools, compilers, utilities, etc.

GCC - GNU Compiler Collection

- The compiler under most open source projects: GCC.
- 4.3.2 as of 2008-09-04.
- Ridiculous number of languages:
  - C, C++, Fortran, Pascal, Objective C, Java, and Ada.
- Ridiculous number of architectures:
  - A29K, Alpha, ARC, ARM, AVR, AVR32, Blackfin, ETRAX CRIS, HC12, H8/300, D30V, DSP16xx,FR-30, FR-V, IA-32 (x86), x86-64, IA-64, Intel i960, IP2000, M32R, M68HC11, MCORE, MIPS, MMIX, MN10200, MN10300, Motorola 68000, Motorola 88000, NS32K, PA-RISC, PDP-11, PowerPC, ROMP, R8C/M16C/M32C, SPU, Stormy16,SuperH, System/390/zSeries, SPARC, VAX, V850, Xtensa.
- Ridiculously cross platform and cross-target (any host platform can compile to any target architecture).
  - Linux, Windows (both via Cygwin and the mingw) and Mac.

BinUtils - GNU binary utilities

- Necessary binary utilities:
  - as - assembler
  - ld - linker
  - gprof - profiler
  - addr2line - convert address to file and line
  - nm - list symbols in object files
  - objdump - dump information about object files
  - readelf - display content of ELF files
  - size - list total and section sizes
  - strings - list printable strings

Other GNU tools

- make
- autoconf
- GDB (GNU Debugger)

Eclipse IDE

- Eclipse 3.2 - "Europa", or "Actually usable", release.
- Java-based, so runs in Windows, Mac and Linux.
- A big ugly brute that does everything.
- Huge project, supported by many industries.
  - Eventually will engulf must other IDEs.
- CDT - C Development Toolkit.
- Various embedded debug plugins (usually via gdb).

KDevelop IDE

- Part of the KDE Desktop Project.
- Full featured.
- Focussed primarily on C,C++ on a Linux host.

Programmers

- Tend to be architecture/manufacturer specific.
- For example, openocd.
  - ARM architecture.
  - Supports FTDI FT2232 and parallel port-based JTAG programmers.
  - Demo!

Misc.

- NSIS - open source Windows installer.
- Code documentation systems (doxygen,javadoc, etc).
- Various Linux packaging tools (apt, yumm, portage, etc).

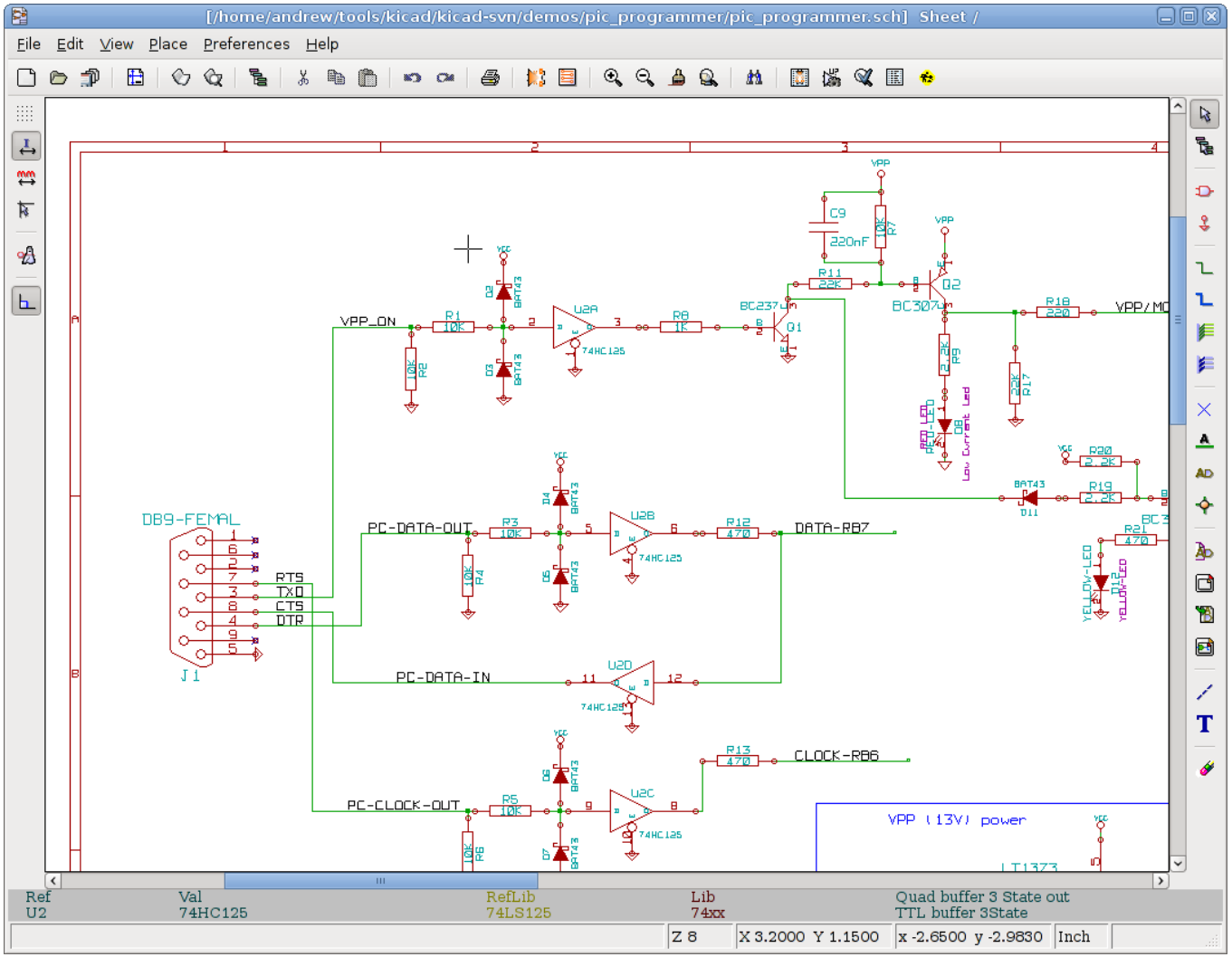
Electrical tools

- Not yet up to commercial tools.
- Schematic Capture / PCB layout.
- Circuit Simulation.
- FPGA design.
- Missing: Parts management.

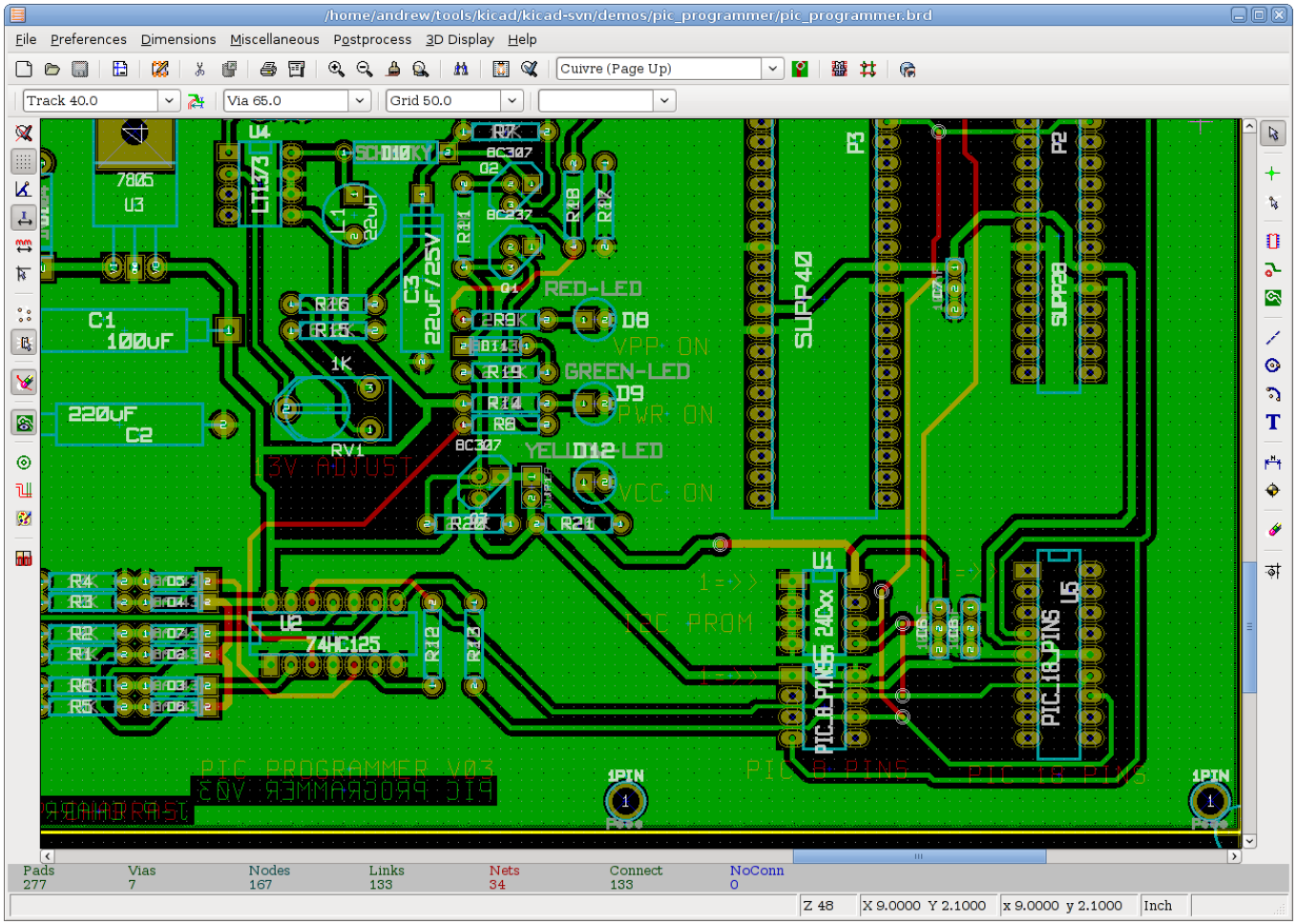
KiCAD

- Functionality:
  - Project Manager.
  - Schematic Capture (eeschema).
  - PCB layout with 3D view (pcbnew).
  - 3D part editor (Wings3D).
  - Gerber viewer (gerbview).
- Lowdown: Shiny and new, but not quite there. Clearly going to be the best.
- Activity: high.
- Maturity: Almost there (critical: missing undo in board layout).
- Community: Largish.
- http://iut-tice.ujf-grenoble.fr/kicad/

KiCAD - eeschema



KiCAD - pcbnew



gEDA

- Functionality: Too much to list. It's a loose conglomeration of open source projects, highlighted by:
  - Netlist component and attribute editor (gattrib).
  - Gerber viewer (gerbv).
  - Schematic capture (gschem).
  - PCB layout (PCB).
  - Spice implementation (ngspice).
- Lowdown: Each component is more mature, but components aren't integrated. Painful to use.
- Activity: High on some components, dead on others.
- Maturity: Mature, but clunky.
- Community: Largish.
- http://www.geda.seul.org/

iverilog - Icarus Verilog

- Functionality: Verilog compiler and optimizer.
- Lowdown: ?
- Activity: Fairly high.
- Maturity: Mature.
- Community: Small.
- http://www.icarus.com/eda/verilog/

qucs - Quite Universal Circuit Simulator

- Functionality: Simulation, with rough schematic capture. Includes microwave components and charts.
- Lowdown: Solid simulator, nice GUI.
- Activity: High.
- Maturity: Getting there.
- Community: Fairly small.
- http://qucs.sourceforge.net/

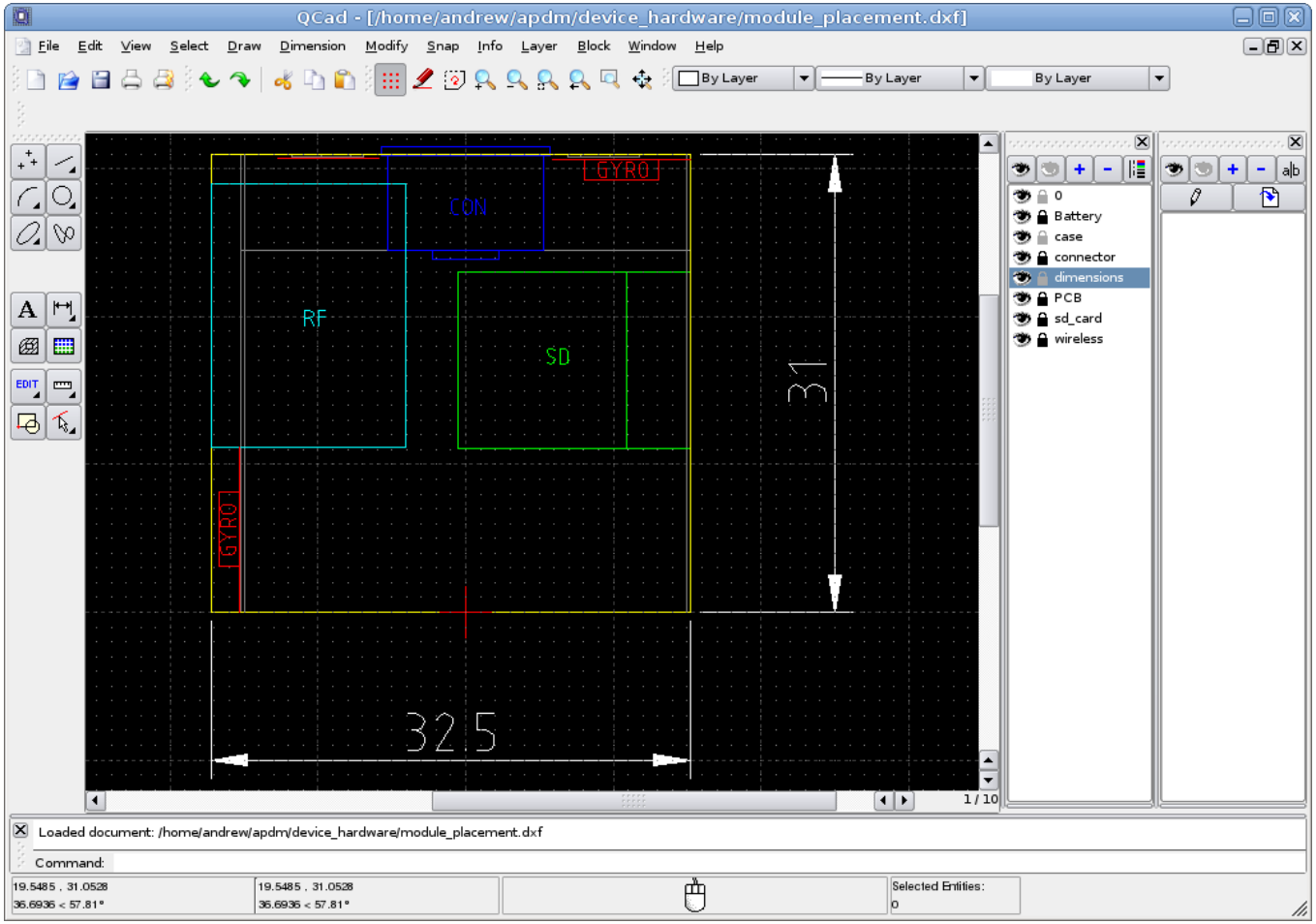
Mechanical tools

- Not even close to commercial tools.
- 2D CAD.
- 3D CAD.
- Simulation and modeling.

QCAD Community Edition

- Functionality: Fully functional 2D CAD program.
- Lowdown: UI a bit clunky, but solid.
- Activity: High (backed by a company).
- Maturity: Mature.
- Comunity: Large.
- http://www.qcad.org/

QCAD


QCad - [/home/andrew/apdm/device_hardware/module_placement.dxf]
File Edit View Select Draw Dimension Modify Snap Info Layer Block Window Help
By Layer
By Layer
By Layer
GYRO
CON
RF
SD
GYRO
325
0
Battery
case
connector
dimensions
PCB
sd_card
wireless
1 / 10
Loaded document: /home/andrew/apdm/device_hardware/module_placement.dxf
Command:
19.5485 , 31.0528
36.6936 < 57.81°
19.5485 , 31.0528
36.6936 < 57.81°
Selected Entities:
0

3D CAD Software

- No big central project (like qcad).
- CAE live Linux distribution - http://www.caelinux.com/
  - FreeCAD - 3D CAD like SolidWorks (very alpha)
  - NetGen/ngsolv - mesh generation and FEA solver.
  - OpenFoam - Fluid dynamics, with an emphasis on combustion.
- Open Cascade Framework: http://www.opencascade.org/
- BRL framework: http://brlcad.org/
- Blender 3D scene modeler: http://www.blender.org/

Open Source Components

- Sure, the tools are great.
- But it's open source, not open tools.
- Way too many projects to list.
- Amazing leverage, but watch the license.
- Hey, buddy, want some source?

Good uses of open components

- Non-mission critical functionality.
- Mission critical, but not your expertise.
- Functionality that can be launched (as opposed to linked).

Don't reinvent it: ship an open source library

- Do you really want to implement ZIP in your program?
- No, so ship your program with libz.
- Your customer will not care one bit.
- Provide the source on your web side, or on the install CD.
- For example:
  - Graphics and multimedia files (libjpeg,libmpeg).
  - Secure communications (libssh, libssl).

(Real Time) Operating Systems

- Plain old Linux (MMU, $>$ 8 MB flash, $>$ 8 MB RAM).
  - Linux 2.6 is "firm" real time (with enough some tweaks, becomes hard).
- uCLinux (no MMU, $>$ 2 MB flash, $>$ 2 MB RAM).
- eCOS - Embedded Configurable Operating System.
  - Compiled to a library.
  - LGPL license allows proprietary code.
  - Huge number of architectures supported.
  - Very feature rich.
- FreeRTOS (ARM only (so far), $<$ 25K FLASH)

Protocol stacks

- Ethernet, CAN, USB, you name it, it's out there.
- Embedded File systems (JFFS2, FAT, EXT3).
- Careful!
  - BSD or LGPL licenses: proprietary code OK.
  - GPL license means your code is now GPL, too.

Embedded services

- License is irrelevant to your code for services.
  - They're called, not linked.
- Web servers (lighttpd).
- Secure communications (openssh,scp,sftp).

Example 1: Deeply embedded medical device

- Disposable medical device with microcontroller, battery, sensors.
- 8 hours of operation using a CR1203 battery ($<$ 1 mA current draw).
- We used open source tools:
  - GCC for AVR Microntrollers.
  - gdb support via `avrdude` support of JATAGICE mkII programmer.
  - Code and documentation stored in git.
- No open components (bare metal C program).

Example 2: The T.O.V.A. 8.0

- Test Of Variables of Attention.
- Real time neuropsychological measure of attention.
- Helps clinicians diagnose disorders like ADHD.
- Cross-platform (Win, Mac, Linux) user interface with database.
- USB device to generate stimuli and record responses.

Example 2: The T.O.V.A. 8.0 - Java User Interface

- Tools
  - Eclipse 3.2 IDE.
  - NSIS installer for Windows.
  - ikiwiki + git: code and document repository.
- Components
  - Derby database.
  - Hibernate object persistence.
  - XSL/XSLT/FO libraries for PDF report generation.

Example 2: The T.O.V.A. 8.0 - USB Device

- NXP LPC2368 (ARM7TDMI) with SPI-based video generation and CAN-based input device.
- Tools
  - Eclipse 3.2, debugging with gdb via Eclipse CDT.
  - GNU toolset.
  - openocd.
- No open components (bare metal C program).

Future of Open Source

- Attracts talented and enthusiastic engineering resources.
- Will fill specialized niches where:
  - Barrier to entry is high (hard to get)
  - Interoperability is low (hard to share)
- Must first solve real problems in software engineering:
  - Handoff from single author to group.
  - Code reuse is hard across projects.
- Businesses will begin to understand it's leverage
  - Funding will go up.
  - Higher funding brings (usually) higher maturity.
- It's certainly not going away, and it's probably in your future.

Thank you

- Questions? Comments?
- You can reach me at: andrew@thetovacompany.com
- Slides available at: http://www.embeddedmoose.com/