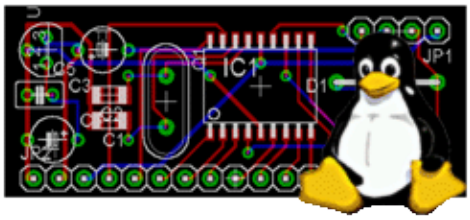
by Aleksandar Milovac
<amilovaclatlptt.yu>

About the author:

Aleksandar ist seit kurzem Absolvent der Technischen Fakultät in Novi Sad, Abteilung für Elektronik und Telekommunikation. Er benutzt Linux seit 1999, in den letzten beiden Jahren Debian GNU/Linux.

Machen wir ein bißchen Elektronik



Abstract:

Als ich noch ein Student der Elektrotechnik war (das ist noch nicht sehr lange her), hatte ich oft die Gelegenheit, mit kommerzieller Software zu arbeiten wie Protel, OrCAD, diversen kommerziellen Compilern... Als ich Linux kennenlernte (früh in '99), fragte ich mich: „Wäre es möglich, das alles unter Linux zu machen?“ Nach vier Jahren freut es mich, sagen zu können: „Ja, es ist möglich!“ In diesem Artikel werde ich dir die Software vorstellen, die du für Linux brauchst.

Translated to English by:
Aleksandar Milovac
<amilovaclatlptt.yu>

Einführung

Linux begann 1991 als ein Hobbyprojekt, aber die Dinge haben sich seither verändert. Heute ist es ein modernes Multiuser−, Multitasking−Betriebssystem, das für seine hohe Stabilität und Zuverlässigkeit bekannt ist. Diese Eigenschaften sind entscheidend, ob du ein Elektrotechniker oder nur ein Bastler bist.

Aber eine der wichtigsten Eigenschaften von Linux ist, daß es sich um ein Open−Source−Projekt handelt. Was hat das mit Elektronik zu tun? Wenn du ein Ingenieur bist, der an einem Projekt arbeitet, könntest du es unter Linux versuchen. Weil die GPL dir erlaubt, die Ideen von anderen zu lernen und deine mit ihnen zu teilen, kannst du Codefragmente benutzen, die bereits geschrieben wurden, und eine Menge Zeit sparen. Dann kannst du dich zum Beispiel auf das Design deiner Hardware konzentrieren. Zudem sind Linux und viele andere Programme in der Open−Source−Gemeinschaft sehr gut dokumentiert und bieten exzellente technische Hilfe. Du schickst dem Autor einfach eine e−mail oder schreibst dich in eine Newsgroup ein − man wird dir gerne helfen. Das sind nur einige von den Vorteilen, die dir Linux für deine Arbeit bietet.

Einige meiner Freunde waren davon nicht überzeugt und fragten: „ Aber warum verlassen wir etwas, das uns vertraut ist, und wechseln zu diesem großen Unbekannten?“ Sie könnten recht haben, aber du findest den Grund nie heraus, wenn du es nicht versuchst. Also versuchen wir es und sehen wir, wo wir landen.

Was brauchen wir?

Jeder, der ein elektronisches Gerät zu bauen versucht, beginnt zuerst mit den Plänen und erstellt dann eine gedruckte Schaltung [*printed circuit board, PCB*], die später gebraucht wird, um alle Komponenten des Gerätes zusammenzusetzen. Dank den modernen Computern können wir das alles in unserem Zimmer neben Bier und Chips machen. Wir kennen alle die Programme, die dafür Verwendung finden: Protel, Pcad, OrCAD... aber sie sind zu teuer für uns Privatnutzer.

Glücklicherweise gibt es ein gutes Programm, das dir alles ermöglicht, ohne dafür Geld zu zahlen. Es heißt EAGLE, was für *Easily Applicable Graphical Layout Editor* steht. Das Programm wird von CadSoft (www.cadsoftusa.com) entwickelt, aber leider ist es Freeware und nicht GPL−Software. Dank den Leuten von CadSoft kannst du es kostenlos auf deinem Linux−Rechner benutzen, aber mit ein paar Beschränkungen. Das PCB mißt 100mm x 80mm, und es kann nur in zwei Schichten erstellt werden. Trotzdem kannst du damit gute PCBs machen. Nachdem ich ein paar Monate lang EAGLE benutzt habe, muß ich sagen, daß es wirklich eine exzellente Software ist. Es ist klein und belegt nicht so viel Speicher wie einige andere Programme. Du wirst etwas Zeit brauchen, um seine Logik zu übernehmen, aber das ist nicht so schwer. Das Interface ist sauber und einfach. Besonderer Dank gebührt dem Library Manager, der einfach und sehr intuitiv ist.

Außer EAGLE gibt es noch andere Programme, die in Universitäten und in der Industrie benutzt werden. Eines davon ist ist das berühmte Matlab (mathworks.com). Es ist quasi ein Standard in Numerik, DSP und Systemmodellierung. Zwar gibt es eine Linux−Version − aber es ist zu teuer. Wenn du nicht viel Geld hast, kannst du das Programm Scilab (scilabsoft.inria.fr) benutzen. Es kostet nichts, und du kannst den Quellcode oder das kompilierte Programm herunterladen. Es kann so gut wie alles, was Matlab kann: Numerik, DSP... Es hat eine Befehlssyntax, die derjenigen von Matlab ähnlich ist. Es ist für X geschrieben, und du wirst die Tcl−Bibliothek brauchen, um es zu installieren. Es hat sogar eine Toolbox namens Scicos (www.scicos.org), die analog zu Matlabs Simulink ist.

Wie steht es mit der Programmierung von Microcontrollern und anderen Sachen? Kann das unter Linux geschehen? Natürlich! Linux ist bekannt für seine hervorragende Unterstützung von vielen Programmiersprachen. Du kannst Programme für deinen Lieblings−Microcontroller schreiben und kompilieren. Du kannst ihn sogar unter Linux programmieren. Kein Bedarf für diese teuren Compiler und anderen Kram. Wenn du viele verschiedene Microcontroller, EEPROMs u.a. programmieren mußt, empfehle ich den Programmierer PonyProg (www.lancos.com/prog.html). Mit ihm und mit passender Hardware kannst du PICs, AVRs, eine Menge verschiedener EEPROMs und mehr programmieren. Er läuft unter X, und er ist einfach und effizient.

Wenn du nur ein AVR−Fan bist, kannst du einen Programmierer namens SP12 nehmen. Er läuft auf der Kommandozeile, und er kommt mit angemessenen Plänen für einen Hardware−Programmierer, sehr einfach und leicht benutzbar. Ich habe ihn für ein Projekt an der Universität eingesetzt, und er hat prima funktioniert. Natürlich ist er nicht der einzige AVR−Programmierer. Davon gibt es viele. Du kannst C− oder Assembler−Programme dafür schreiben. Wenn du dich für C entscheidest, dann solltest du dir den Einsatz des AVR−GCC−Compilers überlegen. Guido Socher hat 2002 einen Artikel darüber geschrieben, den ich erwähnen möchte. Lies ihn, wenn du mehr wissen willst. Er ist sehr gut.

Für alle PIC−Fans ist hier etwas Schönes. Ich benutze Debian GNU/Linux auf meinem Rechner. Als ich mit aptitude die Paketlisten durchging, um zu sehen, ob es etwas über PICs gibt, fand ich ein paar Programme. Das erste heißt Picasm, es ist ein Assembler für PIC−Microcontroller. Das zweite ist simulpic, offensichtlich ein PIC−Simulator. Ich habe diese Programme noch nicht getestet, weil ich PICs in der Vergangenheit nicht verwendet habe, aber du kannst es versuchen. Als PIC−Programmierer empfehle ich Picprg (www.brianlane.com) von Brian Lane. Es ist ein einfaches und leicht zu benutzendes Programm.

Weil wir über Elektronik, Programmierung und ähnliches unter Linux sprechen, sollte ich eine weitere interessante Sache erwähnen. Vor kurzem wurde Linux in Bereiche wie das Design von integrierten System und Echtzeit−Betriebssysteme eingeführt. Wenn du Linux als Echtzeit−Betriebssystem benutzen willst, das zum Überwachen von industriellen Prozessen oder zur Kontrolle von Maschinen dient, solltest du zwei sehr beliebte Echtzeit−Erweiterungen für Linux ausprobieren: RTLinux (www.fsmlabs.com) und RTAI (www.rtai.org). Beide stehen unter der GPL und sind sehr gut dokumentiert.

Fazit

Das war eine kurze Präsentation von kostenloser bzw. Open−Source−Software für die tägliche (und fortgeschrittene) Elektronik. Wenn du darüber nachdenkst, Elektronik zu machen, solltest du Linux probieren. Wie du siehst, gibt es eine Menge Programme, die du benutzen kannst.

In den letzten vier Jahren habe ich nach kostenloser bzw. Open−Source−Software für Linux gesucht, die alle Programme ersetzt, die ich an der Universität oder auf meinem Windows−Rechner zu Hause verwendet habe. Ich brachte es fertig, alles zu ersetzen, womit ich unter Windows arbeitete. Ich suche weiterhin nach besserer Software. Aber die Dinge sind nun viel einfacher mit Linux.

Referenzen

- Debian GNU/Linux
- EAGLE−Software
- Scilab
- PonyProg−Programmierer
- SP12−Programmierer
- Picprg−Programmierer
- RTLinux−Homepage
- RTAI−Homepage
- AVR−GCC libc
- LinuxFocus−Artikel 231: Programming the AVR Microcontroller with GCC [auch in deutscher Übersetzung vorhanden, A.d.Ü.]

2005−01−11, generated by lfparser_pdf version 2.51