embedded-projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

[EDITORAL]

NEW RELIGION

Projekte von und für die Gemeinschaft. Das Embeddded Projects Journal - unser Glaube an die OpenSource Community. Auch 2010 verbreiten wir das gedruckte und digitale Wort für frische Projektideen.

[PROJECTS]

WELCOME

Ausgabe 5 von Embedded-Projects Journal

Die Einleitung dieser Ausgabe widme ich allen, die mir bis hier und heute tatkräftig bei dem open-source-Projekt Zeitschrift und vielen anderen Unternehmungen geholfen oder sie initiiert haben.

Danke Claudia für die ewige Geduld und Unterstützung, Sigi für die vielen Ideen, den Rat und Deine positive Lebenseinstellung, Hubert für die guten Artikel und die Unterstützung auf ganzer Linie, Axl und Erik für die vielen bunten Farben und den reibungslosen Ablauf, Michi für den nettesten Physikstudenten, Jörgi, Melli und Rolfi mit Ihren immer in-guten-Bahnenkreisenden Ideen, Claude für die technischen Inspirationen, Marcel für Deinen Einsatz, Andreas für alles, Roman für die Mails und guten Tipps und Gespräche, Ernst für Dein Interesse, Michi für Struktur und Ordnung im Online-Generator, Tom für den Überbringer des Linux-Virus, Robert, ein fleißiger österreichischer Bastler, Thomas für die prima Beratung im Zeichen des Wapperls, Michi für interessante Gespräche über Sinn, Sein und Werte, Wolf für die gute Zusammenarbeit, Josef und Marta für die immer wieder tatkräftige Unterstützung und Ines unsere fleißige Ameise, die dafür sorgt, dass alle das Heft zugesendet bekommen.

Riesen Dank an alle Sponsoren:

- IBV - Echtzeit- und Embedded GmbH & Co. KG unseren Hauptsponsor
- IN-Circuit GmbH
- LEITON GmbH Printed Cervice Boards
- Simple Solutions
- www.TOM-IC.com
- Weingut Hörner
- MHS-Elektronik GmbH & Co. KG
- Quategra GmbH
- Mixed Mode GmbH
- Weather Dock
- Anix GmbH
- Net of Trust - Solution GmbH

Benedikt Sauter
sauter@embedded-projects.net

shop.embedded-projects.net

THE ONLINESHOP - HARDWARE FOR YOUR PROJECTS

ARM Boards

ARM USB Debugger (ARM-USB-Tiny)

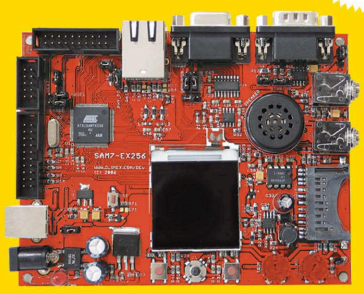
ARM7/ARM9/Cortex-M3 und XScale Debuginterface für OpenOCD und CrossWorks

ARM USB Debugger
44,90 €
embedded-projects SHOP

AT91SAM7X256+TFT+Ethernet (SAM7-EX256)

Bestückt mit einem 32 Bit ARM-Mikrocontroller mit 256 kB Flash, 64 kB RAM, 55 MHz, Ethernet 10/100, USB 2.0, RS232, CAN, MMC/SD-Card Slot und TFT-Display

AT91 SAM7X256 TFT+ Ethernet
119,90 €
embedded-projects SHOP



MSP430 Boards

MSP430F1611 Adapterplatine (MSP430-H1611)

MSP430F1611 mit 48K Bytes Programm Flash, 256 Bytes Daten Flash, 10K Bytes RAM

MSP430F1611 Adapterplatine
24,90 €
embedded-projects SHOP

MSP430 USB JTAG Adapter (MSP430-JTAG-TINY)

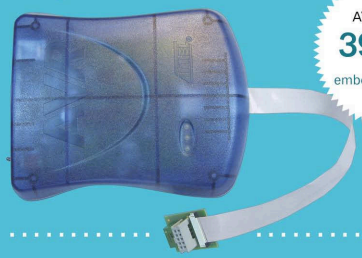
Programmierung/ Debugging für alle MSP430Fxxx Flash-Microcontroller

MSP430 USB JTAG Adapter
64,90 €
embedded-projects SHOP

AVR Boards

Atmel AVRISP mkII (USB)

Original AVRISP mkII In-System Programmer von Atmel.



Atmel AVRISP mkII
39,90 €
embedded-projects SHOP

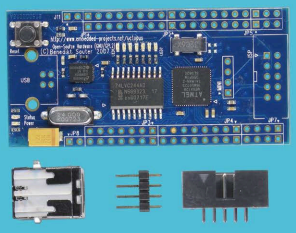
AVR Starterkit (inkl. USBprog, Netzteil und ATMega8)

AVR-Starterkit bestehend aus USBprog, AVR-Starterplatine, ATMega8 und Steckernetzteil.

SET AVR Starterkit
61,90 €
embedded-projects SHOP

Octopus USB

Octopus bietet viele bekannte Schnittstellen aus der Mikrocontrollerwelt über ein einfaches USB Gerät an.

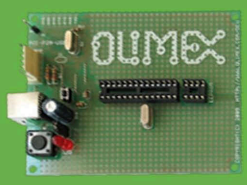


Octopus USB
39,00 €
embedded-projects SHOP

PIC Boards

PIC Entwicklungsplatine (PIC-P28-USB)

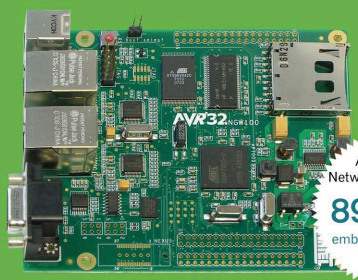
PIC Mikrocontroller Entwicklungsboard für 28-polige ICs + USB RS232.



PIC Entwicklungsplatine (PIC-P28-USB)
24,90 €
embedded-projects SHOP

AVR32 Boards

ATNGW100 Network Gateway Kit

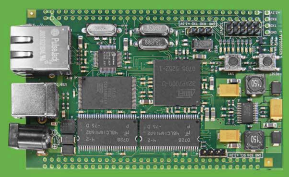


ATNGW100 Network Gateway Kit
89,90 €
embedded-projects SHOP

Grasshopper AVR32-Board

*Open-Source-Version (ohne CD+Kabel)

freie Plattform für die AVR32 Entwicklung



Grasshopper AVR32-Board*
ab 85,00 €
embedded-projects SHOP

Jetzt schneller zum Shop:
www.eproo.net

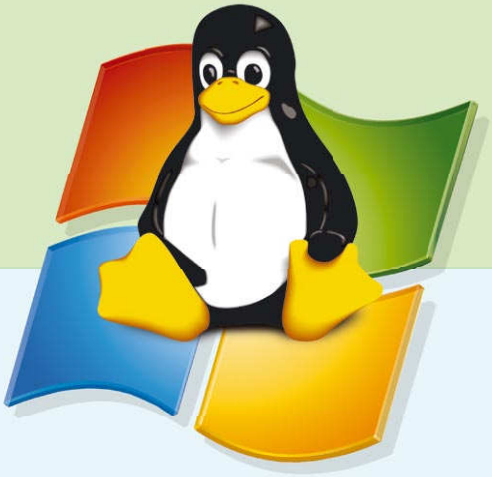
- großes Sortiment an Evaluations- und Testboards
- bekannte Open Source Projekte
- übernommener Artikelbestand von www. mikrocontroller.net
- faire Preise
- Produkte direkt vom Hersteller
- bequeme Zahlungabwicklung und schneller Versand

Jetzt Neu:
Versand weltweit
3,95 Euro

Linux ist nicht Windows

Contest project

Felix Schwarz



Vorwort

Dies ist eine deutsche Übersetzung von Felix Schwarz [7] unter tatkräftiger Mithilfe einiger Leser. Es wurde versucht, die Formulierungen möglichst weitgehend zu übernehmen und die Tonalität des Originalartikels zu treffen. Der Originalartikel ist leider nur noch bei archive.org einsehbar. Es gibt allerdings unter http://linux.oneandoneis2.org/LNW.htm einen vom Originalautor abgeänderten Artikel. Im Folgenden ist mit „ich" der ursprüngliche Autor Dominic Humphries gemeint.

Im folgenden Artikel wird oft der Oberbegriff „Linux" verwendet, obwohl Linux nur der Kernel des gesamten Open-Source Betriebssytems ist. Genau genommen spricht man wenn man von Linux redet nur vom Kernel und mit der Bezeichnung GNU/Linux das Betriebssystem inklusive der verschiedene Open-Source Projekte welche für den Betrieb notwendig sind. Es liest sich einfach besser...

... irgendwo in einem Linux-Forum

„Hi! Ich benutze Linux seit ein paar Tagen und es ist eigentlich großartig. Aber es ist eine Schande, dass [dies oder das] nicht so funktioniert wie unter Windows. Warum schreiben nicht alle Programmierer die ganze Software neu, so dass sie mehr wie Windows funktioniert? Ich bin mir sicher, dass Linux viel mehr Benutzer hätte, wenn sie es täten!"

Sie haben bei Ihrer Antwort vielleicht sogar jemanden angepflaumt, nur um anschließend von einem Linux-Neuling niedergemacht zu werden, der Ihre Antwort (basierend auf jahrelanger Erfahrung mit einem anderen Betriebssystem und ein paar Stunden Linux) so interpretiert, dass seine Idee revolutionär und brillant ist und nur Sie sie nicht mögen, weil Sie ein „altmodischer Linux-Nutzer" sind, der GUIs für eine Ausgeburt des Teufels hält und meint, jeder sollte gezwungen werden, das CLI [„Command Line Interface": Die Kommandozeile oder Shell, Anm. von Felix Schwarz] zu benutzen.

Dieser Artikel zielt darauf ab, Neulingen zu erklären, warum Ihre Ideen eher abgelehnt als freudig begrüßt werden.

Zuallererst das beliebteste Argument: *„Wenn Linux das täte, würden viel mehr Leute von Windows zu Linux konvertieren!"*

Erlauben Sie mir etwas zu erklären, das für das Verständnis von Linux fundamental ist: Die Linux-Gemeinschaft [1] versucht **nicht**, dem durchschnittlichen Windowsbenutzer ein Ersatz-Betriebssystem anzubieten. Das Ziel von Linux ist **nicht** „Linux auf jedem Schreibtisch".

Wirklich. Es ist wirklich nicht das Ziel [2]. Ja, beides sind Betriebssysteme. Ja, beide **können** für den gleichen Zweck eingesetzt werden. Aber das macht Linux zu einer **Alternative**, nicht zu einem **Ersatz**. Dies scheint vielleicht nur eine unbedeutende Unterscheidung zu sein, aber sie ist tatsächlich extrem wichtig.

Auto!= Motorrad

Linux <=> Windows ist wie **Motorräder <=> Autos**:

Beides sind Fahrzeuge, die Sie über Straßen von A nach B bringen. Aber sie haben unterschiedliche Formen, unterschiedliche Größen, besitzen andere Steuerelemente und sie **funktionieren auf völlig verschiedene Weisen**. Man kann sie nicht beliebig austauschen. Beide sind für gewisse Dinge besser geeignet und für andere weniger gut. Sie sollten das Zweckmäßigere wählen und nicht einfach irgendeines und von diesem erwarten, dass es alles machen kann, was das andere könnte.

Jemand, der ein Auto fährt, könnte eines Tages in einem langen Stau stehen und ein Motorrad schnell neben sich vorbeifahren sehen. Er könnte die Fähigkeit des Motorradfahrers beneiden, etwas zu ignorieren, das ein lähmendes Problem für ein Auto darstellt. Wenn der Autofahrer dann sagen würde *„Ich weiß alles über Autos, also muss ich auch alles über Motorräder wissen!"*, läge er falsch.

Wenn dieser Fahrer ein Motorrad kaufen würde und dann feststellt, dass er verwirrt wäre, dass das Gaspedal ein drehbarer Handgriff ist statt eines mit dem Fuß zu bedienenden Gaspedals, könnte er sich beschweren, dass Motorräder mit einem Gaspedal ausgestattet werden sollten.

- Wenn dieser Fahrer eine Frau und zwei Kinder hätte, könnte er es als Mangel betrachten, dass auf dem Motorrad nur eine Person Platz findet. Er könnte vorschlagen, dass das Motorrad neu konstruiert werden könnte, so dass vier Personen Platz fänden, mindestens aber zwei nebeneinander.
- Wenn dieser Fahrer beim Versuch zu fahren feststellen würde, dass er umkippt, weil er nicht daran gewöhnt ist, die Balance halten zu müssen, könnte er vorschlagen, dass Motorräder mit vier Rädern konstruiert werden sollten.

- Wenn der Fahrer feststellen würde, dass er sich in Kurven in der Schräglage wiederfindet, würde er vielleicht vorschlagen, dass Motorräder mit Stabilisatoren ausgestattet werden sollten, um sie in den Kurven aufrecht zu halten.
- Wenn der Fahrer sein Motorrad vor Diebstahl schützen wollte, könnte er sich beschweren, dass es keine Türen gibt, um potenzielle Diebe draußen zu halten, so dass sein Motorrad viel leichter gestohlen werden könnte als ein Auto.
  Wenn der Fahrer ein Motorradhelm als Last empfände, könnte er vorschlagen, dass ein Airbag in den Lenker als Alternative zu dem lästigen Helm eingebaut werden könnte.

Und in jedem dieser Fälle würde er **falsch** liegen. Weil er denkt, dass ein Motorrad ein Auto **ersetzt**, meint er, es könne und solle alles tun, was ein Auto auch kann. Er denkt, dass es genauso funktionieren könnte wie ein Auto, dass „fehlende" Funktionen eines Autos einfach aufgepfropft werden könnten.

In der gleichen Weise machen wohlmeinende Linux-Neulinge Vorschläge, wie Linux ähnlicher dem gemacht werden könnte, an das sie gewöhnt sind. Und sie kommen aus den gleichen Gründen nicht voran. Linux und Windows können beide für die gleichen Zwecke eingesetzt werden, aber das gleiche gilt für Autos & Motorräder. Das heißt aber nicht, dass direkt eines gegen das andere tauschen kann und es heißt auch nicht, dass Eigenschaften unmittelbar austauschbar sind oder sein sollten.

Zu viele Leute denken, dass eine Migration von Windows zu Linux ähnlich sei wie ein Wechsel von BMW zu Mercedes. Sie denken, dass die Steuerungsmöglichkeiten die gleichen sein sollten, ihr Wissen direkt übertragbar sein sollte und sämtliche Unterschiede sollten rein kosmetisch sein.

Sie denken
„Ich brauche ein Auto, um eine Straße zu benutzen. Ich brauche ein Betriebssystem, um einen Computer zu benutzen. Autos funktionieren alle auf die gleiche Weise, daher sollten Betriebssysteme alle auf die gleiche Weise funktionieren."
Aber das stimmt nicht.
„Ich brauche ein Fahrzeug, um eine Straße zu benutzen. Ich brauche ein Betriebssystem, um einen Computer zu benutzen. Ich weiß, wie man Auto fährt, aber ich habe keine Ahnung von Motorrädern. Ich weiß, wie man Windows benutzt, aber ich habe keine Ahnung von Linux." - das wäre korrekt.

Ein erfahrender Computer-Benutzer

Ein Windowsnutzer muss erkennen, dass er nur ein erfahrener **Windows**-Nutzer ist, nicht ein erfahrener **Computer**-Benutzer; genauso wie ein Autofahrer **nur** ein Autofahrer ist, nicht ein Alle-Straßenfahrzeuge-Fahrer. Ein Windows-Nutzer mit Linux muss erkennen, dass er gerade wieder ein Anfänger geworden ist, genauso wie ein Autofahrer auf einem Motorrad. Ein Windows-Nutzer muss gewillt sein zu lernen, dass es auch andere Wege gibt, eine Aufgabe zu erledigen, so wie ein Autofahrer sich daran gewöhnen muss, dass ein Lenker das Lenkrad ersetzt und ein Helm eine Notwendigkeit darstellt, obwohl er vorher nie einen brauchte. Und sie müssen darauf vorbereitet sein zu akzeptieren, dass „unterschiedlich" nicht „schlechter" heißt.

Diese simplen Fakten verursachen große Schwierigkeiten für die eingeweihten Windows-Benutzer. Sie kommen zu Linux mit vielen tief verwurzelten Windowsgewohnheiten und der Einstellung *„Ich weiß genau, wie man einen Computer zu benutzen hat, vielen Dank."*
Das Problem ist nur, dass sie genau das nicht wissen. Sie wissen nur, wie man Windows benutzt. Wenn sie mit einem anderen Betriebssystem arbeiten, können diese „power user" genau die sein, die die größten Probleme haben: Sie müssen viel mehr umlernen.

Wo ein Neuling einfach sagen würde *„Ich weiß es nicht"* und anfangen würde, es auszuprobieren oder in Foren zu fragen, wird der Windows-Power-User sagen: *„Ich **weiß** wie man das macht, ich mache nur das, das, das und dann ... Es funktioniert nicht! Blödes Betriebssystem!"*
Und dann werden sie sagen: *„Wenn erfahrene Nutzer wie ich es nicht zum Laufen bekommen, wird ein Anfänger keine Chance haben! Linux ist nicht einmal annähernd für die Desktop-Benutzung geeignet!"*.
Sie erkennen nicht, dass all ihr Wissen gegen sie arbeitet und ihnen **mehr** Probleme bereitet als den **weniger** erfahrenen Nutzern. Sie haben den Fehler gemacht zu denken, dass Linux eine andere Software ist, die das gleiche macht wie Windows, wobei es in Wirklichkeit eine andere Software ist, die **andere** Dinge macht. Linux macht keinen schlechten Job für die gleichen Aufgaben, es macht einen **guten** Job für **alternative** Aufgaben.

Linux ist eine Alternative zu Windows, aber kein Ersatz. Es wird niemals ein Ersatz sein, weil beide unterschiedliche Ziele haben. Microsofts Ziel ist es, ihre Software auf möglichst vielen PCs zu haben, weil ihre Priorität auf Gewinn liegt. Linux hat solche Ziele nicht, weil Linux frei ist. Es hat andere Prioritäten.

Um das zu verstehen, muss man FOSS (Freie Open Source Software) [3] verstehen. Es ist sehr verständlich, dass Linux-Anfänger es noch nicht verstehen - sie kennen es noch nicht. Sie sind gewöhnt, an proprietäre Software zu denken. Lassen Sie es mich daher erklären:

Was ist Open-Source?

Typische FOSS Software wird von jemandem erstellt, der herumschaut, keine existierende Software findet, die er mag, und daher seine eigene schreibt. Weil er so ein netter Kerl ist, schmeißt er anschließend den Quelltext ins Internet und sagt: „Bedient euch". Er kann das machen, weil es ihn nichts kostet, die Software zu kopieren, so dass es ihn nicht mehr kostet, der ganzen Welt seine Software zu geben, als wenn er sie nur für sich selbst benutzen würde. Er erleidet keinen Nachteil dadurch, seine Software wegzugeben.

Wichtig ist aber: Er **profitiert** auch nicht davon, seine Software zu veröffentlichen. Ob sie von einer Person oder eine Milliarde benutzt wird, macht für den Entwickler keinen Unterschied. Natürlich wird er Befriedigung dadurch bekommen, dass er ein populäres Produkt erschaffen hat: Die Anzahl der Benutzer kann ein netter Ego-Schub sein; ein Weg, wenn Sie so wollen, „of keeping score". Aber es bringt ihm kein Geld: Es ist FOSS.

Wenn die Software ein Erfolg ist, interessieren sich andere Leute dafür und helfen, sie zu verbessern. Das ist der größte Vorteil von FOSS: Jeder Benutzer ist ein potenzieller Entwickler. Jeder kann

mitmachen und seinen Teil dazu beitragen, damit die Software besser arbeitet, mehr macht, weniger Fehler hat. Es ist großartig, wenn eine Software eine Gemeinschaft von Entwicklern anzieht. Es ist großartig **für die Software**. Es verbessert die Software. Es bereichert den Entwickler aber nicht. Es beansprucht nur mehr von seiner Zeit.

FOSS ist das genaue Gegenteil von Software wie Windows: FOSS dreht sich nur **um die Software**. Es geht **nicht** um die Anzahl der Endbenutzer. Software, die zwar gut funktioniert, aber nur wenige Benutzer hat, ist für die kommerzielle Software-Welt ein Flop, aber für FOSS ein Gewinn.

FOSS dreht sich um hochqualitative Software, Software, die Dinge TUN kann. Wenn Sie daran interessiert sind, sie zu benutzen, wird von Ihnen auch erwartet, dass Sie Zeit investieren, **wie** sie zu benutzen ist. Software wurde von Leuten geschrieben, die viel von ihrer Freizeit ohne persönlichen Gewinn investiert haben, und Ihnen kostenlos zur Verfügung gestellt. Das Geringste, was Sie tun können, um diesen Beitrag zurückzuzahlen ist, ein klein bisschen Ihrer eigenen Zeit zu investieren, bevor Sie sich beschweren, dass etwas nicht so funktioniert wie ähnliche Windows-Software.

„Ah, jetzt habe ich dich." sagt der Anfänger süffisant „Es gibt Linux-Projekte mit dem Ziel, Windows zu ersetzen, und nicht nur eine Alternative zu sein."

Es ist leicht zu erkennen, woher diese Idee kommt. KDE und Gnome zum Beispiel bieten eine Desktopumgebung an, die deutlich Windows-ähnlicher ist als typische Linux-Windowmanager und die CLI. Linspire ist eine Distribution, die nahezu vollständig auf der Idee basiert, Linux Windows-ähnlich zu machen.

Trotzdem - paradoxerweise - stützen diese Projekte **meinen** Punkt mehr als den des Anfängers.

Warum? Weil diese Projekte normale FOSS-Projekte sind, die sich nur darum drehen, Software besser zu machen. Der einzige Unterschied ist, dass eine der Definitionen von Qualität bei diesen Projekten ist *„Wie einfach kann ein Windows-Nutzer es verwenden?"*.

Sobald Sie das einbeziehen, können Sie nur zustimmen, dass dies 100% typische Linux-Projekte sind mit dem einzigen Ziel, Software besser zu machen. Diese Projekte werden von überdurchschnittlich selbstlosen Linux-Entwicklern betrieben: Sie erschaffen keine Programme für ihren eigenen Nutzen, weil sie Linux sehr gut kennen. Stattdessen programmieren sie Software nur für den Nutzen anderer Leute: Software, die den Übergang von Windows zu Linux einfacher macht.

Diese Entwickler haben erkannt, dass es Windows-Nutzer gibt, die zu Linux wechseln möchten, und sie haben große Anstrengungen hineingesteckt, um eine Linux-Umgebung zu erzeugen, die Windows-Nutzer angenehm vertraut finden. Aber sie haben das **nicht** getan, um zu versuchen, Windows zu ersetzen, obwohl das Endresultat diesen Eindruck vermitteln könnte. Das Endziel macht den Unterschied: Das Ziel ist nicht die Erschaffung eines Windows-Ersatzes, das Ziel ist, den Übergang von Windows-Benutzern zu Linux zu vereinfachen.

Feindseligkeit aus der Community gegenüber diesen Projekten ist nicht ungewöhnlich. Manchmal aus rationalen, verständlichen Gründen (*„KDE ist ein Ressourcenfresser, benutze daher Fluxbox (KDE, GNOME, Fluxbox, etc. sind Desktoanwendeungen"*), aber manches scheint ein irrationales, feindliches, „Windows-ähnliche Software ist schlecht" Verhalten zu sein. In Wirklichkeit ist dies aber kein Anti-MS oder Anti-Windows-Verhalten. Stattdessen ist es die viel verständlichere Abneigung gegen das, was man nicht versteht.

Der ‚typische' Linux-Nutzer ist ein Hobbyist: Er benutzt Computer, weil Computer Spaß machen, Programmierung Spaß macht, Hacking Spaß macht. Und Linux ist ein weitaus besseres Betriebssystem für einen Hobbyhacker: Er kann es nach Belieben bis zur untersten Ebene auseinander nehmen und wieder zusammensetzen.

Anmerkung von Felix Schwarz:
Der Begriff ‚Hacker'/'Hacking' sollte hier verstanden werden als ‚eine Person, die sich daran erfreut, ein tiefgreifendes Verständnis der internen Arbeitsweise eines technischen Systems zu erlangen' (Quelle: Wikipedia [4]). Ähnlich liest sich die Definition des Chaos Computer Clubs [5].

Demgegenüber steht der ‚Cracker', der sich (häufig destruktiv) mit der Ausnutzung von Sicherheitslücken beschäftigt.

Allerdings gibt es zurzeit unter jenen, die zu Linux kommen, einen großen Anteil an solchen, die weder Hobbyisten noch Hacker sind. Sie wollen einen Computer, der „einfach funktioniert", einen Computer, der wie Windows funktioniert. Sie sind nicht daran interessiert, Linux zu konfigurieren, damit es so funktioniert, wie sie es wollen; sie wollen, dass es sofort nach dem Anschalten funktioniert.

Und das ist auch vollkommen OK, aber aus der Perspektive eines typischen Linux-Nutzers ist das wie jemand, der ein Legoauto haben will, das bereits aufgebaut und zusammengeklebt geliefert wird, damit nichts auseinander fallen kann. Ihrem Verständnis ist das total fremd. Der einzige Weg, wie sie reagieren können, ist mit einem erstaunten *„Warum könnte **irgendjemand das** wollen?"*.

Es ist erstaunlich. Wenn Sie ein vorgefertigtes Modellauto haben wollen, kaufen Sie sich ein Spielzeugauto. Wenn Sie ein Auto haben wollen, das Sie selbst aufbauen und auseinander nehmen können, kaufen Sie Lego. Warum sollte jemand ein Legoauto wollen, das nur als gewöhnliches Spielzeugauto verwendet werden kann? Der ganze Witz an Lego ist, dass man Spaß dabei haben kann, es selbst zusammenzubauen!

So reagiert ein typischer Linux-Nutzer auf die *„Warum kann es nicht einfach funktionieren?"*-Abteilung:
„Wenn Sie etwas wollen, das einfach funktioniert, verwenden Sie Windows. Wenn Sie es hacken wollen, verwenden Sie Linux. Warum wollen Sie zu Linux wechseln, wenn Sie kein Interesse daran haben, Vorteile aus seiner Open-Source-Herkunft zu ziehen?"

Die Antwort ist normalerweise, dass sie **nicht** wirklich zu Linux wechseln wollen. Sie wollen nur **weg** von Windows: Sie rennen weg vor den ganzen Viren, sie fliehen vor der Malware, sie streben danach, frei zu sein von den Beschränkungen, wie sie ihre bezahlte Software verwenden dürfen, sie versuchen, den Fängen des E.U.L.A. zu entkommen. Sie versuchen nicht, sich **in** Linux einzufinden, sie versuchen, von Windows **wegzukommen**. Linux ist schlicht und einfach die bekannteste Alternative.

Früher oder später...

Sie könnten denken, *„Okay, das erklärt, warum Entwickler keine bewussten Anstrengungen unternehmen, damit ihre Software so funktioniert wie Windows. Aber sicherlich könnte Linux-Software eine GUI haben, die für Windowsbenutzer nutzerfreundlich ist, ohne mit den FOSS-Prinzipien in Konflikt zu geraten?“*

Es gibt ein paar Gründe, warum das nicht der Fall ist.

Erstens:

Glauben Sie **wirklich**, dass jemand, der eine Software schreibt, ihr absichtlich eine lausige Benutzerschnittstelle [6] gibt?

Wenn jemand einen großen Batzen seiner Freizeit dafür opfert, um eine Software zu schreiben, wird er die Benutzerschnittstelle (UI) **so gut wie möglich** machen. Die UI ist ein sehr wichtiger Teil der Software: Es ist sinnlos, Funktionalität zu haben, auf die man nicht mittels einer UI zugreifen kann. Vielleicht wissen Sie es nicht, aber es gibt **immer** einen Grund, warum die UI so funktioniert wie sie es tut. Der Grund? **Weil es die beste Oberfläche ist, die der Programmierer schaffen konnte.**

Bevor Sie darauf bestehen, dass eine Windows-ähnliche Oberfläche die Software besser machen würde, behalten diesen Punkt im Hinterkopf: *Der Schöpfer dieser Software, ein Programmierer, der per Definition **weit** mehr als Sie über diese Software weiß, stimmt nicht mit Ihnen überein.* Er könnte daneben liegen, aber die Chancen stehen schlecht.[7]

Zweitens:

Es GIBT bereits nette, Windowsnutzer-freundliche GUI Oberflächen. Mir fällt spontan keine Funktion ein, die man nicht über eine GUI kontrollieren könnte, unabhängig davon auf welcher Ebene. Sie können den Kernel kompilieren (make xconfig), Ihre Firewall einstellen (fwbuilder), Ihre Festplatte partitionieren (qtparted) ... alles da, schön, interaktiv, intuitiv und benutzerfreundlich.

Aber der „Veröffentlichungszyklus“ von Linux ist nicht wie bei Windows. Sie bekommen nicht das fertige, hochglanzpolierte Paket von Anfang an. GUIs bringen Komplexität, aber keine zusätzliche Funktionalität zur Software. Ein Entwickler setzt sich nicht hin und entwirft eine schöne GUI, die nichts tut. Er setzt sich hin und schreibt eine Software, die das tut, was er benötigt.

Das erste, was eine Software tut, ist, dass sie von der Kommandozeile aus benutzbar ist. Es wird vermutlich alle möglichen Aufrufoptionen und vielleicht eine längere Konfigurationsdatei haben. So fängt es an, weil diese Funktionalität notwendig ist. Alles andere kommt später. Und selbst wenn die Software eine nette GUI hat, ist es wichtig sich zu erinnern, dass es normalerweise immer noch komplett über die Kommandozeile und die Konfigurationsdateien gesteuert werden kann.

- Dies ist so, weil das CLI viele Vorteile bietet: Das CLI ist universell. Jedes Linux-System hat ein CLI. Jedes Programm kann von der Kommandozeile aus ausgeführt werden. Es ist einfach, Software auf anderen Rechnern per Kommandozeile zu steuern.

Nichts davon stimmt für die GUI: Einige Linux-Rechner haben kein X11 Windowsystem installiert, manche Software hat kein GUI, manche Software ist nicht von GUI-Menüs aus verfügbar. Meist ist es nicht einfach oder praktisch, ein GUI-Werkzeug auf einem anderen Rechner zu benutzen.

Und zuletzt können verschiedene GUI-Oberflächen existieren, die dieselbe Aufgabe erledigen, und niemand kann sagen, welche Sie installiert haben.

Erinnern Sie sich also, wenn Sie fragen *„Wie mache ich...?“*, wird Ihnen meistens gesagt, wie Sie es mit Hilfe der Kommandozeile erledigen können. Das **heißt nicht**, dass es **nur** über die Kommandozeile gemacht werden kann. Es zeigt einfach die relative Wichtigkeit, die die GUI während der Entwicklung eines Softwareprojekts hat verglichen mit der CLI

- Windows ist absolut GUI-zentriert. Es ist ein GUI-basiertes Betriebssystem mit einer lausigen Kommandozeile (die aber bald verbessert wird). Es gibt eigentlich keine Nicht-GUI Windows-Software. Das führt dazu, dass Leute meinen, eine GUI sei ein vitaler und integraler Teil einer Software. Aber unter Linux wird Software veröffentlicht, sobald sie funktionsfähig ist. Erst nachdem sie stabil und relativ fehlerfrei sowie einigesan Funktionalität zu bieten hat, ist es sinnvoll, eine GUI hinzuzufügen.
  Versuchen Sie, Software ohne eine hilfreiche GUI als Vorschau („sneak preview“) anstatt als fertiges Produkt zu sehen. FOSS ist sehr selten nur „fertig“, sie wird immer verbessert. Mit der Zeit **wird** sie benutzerfreundlich gemacht. Aber meistens ist es wichtiger, die Software **funktionsfähig** zu machen anstatt sie **nur äußerlich zu verschönern**. Seien Sie froh, dass Sie die Funktionalität lange vor allen Warmduschern bekommen haben, die eine gute GUI brauchen, anstatt die Software von morgen bereits heute zu verlangen. FOSS ist mehr ein Weg als ein Ziel.

Das letzte, was Sie im Kopf behalten sollten, ist, dass GUIs für Software oft separate Programme sein werden. Sie könnten sogar komplett unabhängig vom originalen Programm, von völlig anderen Entwicklern programmiert werden. Wenn Sie eine GUI wollen, ist es nicht unwahrscheinlich, dass es zwei getrennte Installationen sein werden, statt alles in einem Stück zu haben.

Das bedeutet zugegebenerweise, dass es einen zusätzlichen Schritt gibt, um das schwer fassbare „Windows-mäßige“ GUI-Verhalten zu bekommen, aber sollte nicht davon ablenken, dass Sie jetzt sofort fast alles über eine schöne Windows-ähnliche Oberfläche machen können. Merken Sie sich nur: Eine GUI ist normalerweise der LETZTE Schritt, nicht der erste. Linux bevorzugt das Aussehen nicht gegenüber der Funktion.

Drittens:

Linux wurde ganz bewusst für den gut informierten, erfahrenen Nutzer gebaut statt für den ungebildeten Anfänger. Aus zwei Gründen:

- Unwissenheit mag ein Segen sein, aber nur für kurze Zeit. Wissen ist ewig. Es mag Tage, Wochen oder Monate dauern, um Ihren Wissenstand vom „Linux-Anfänger“ zum „durchschnittlichen Linux-Benutzer“ zu heben, aber wenn es einmal da ist, haben Sie noch viele Jahre der Linuxnutzung vor sich.
  Viel Code zu erzeugen, um Software einfacher für Neulinge zu machen, wäre so als ob man immer Stützräder an Fahrrädern anbringen würde. Es wäre einfacher für den Start, aber danach? Ich bin mir sicher, dass Sie jetzt kein Fahrrad mit Stützrädern kaufen würden. Und zwar nicht, weil Sie ein „anti-user-friendly“ Freak sind. Nein, einfach weil es für Sie nutzlos wäre und auch nutzlos für alle außer Anfänger, Stützräder würden Sie nur stören.
- Unabhängig wie gut Software ist, sie ist immer nur so gut wie der Benutzer. Die sicherste Tür der Welt ist kein Hindernis für Diebe, wenn Sie das Fenster offen, die Tür unverschlossen oder die Schlüssel im Schloss lassen. Selbst mit dem besten Motor der Welt würden Sie nicht weit kommen, wenn Sie Diesel anstatt Benzin einfüllen. Linux legt **alle** Macht in die Hände der Benutzer. Das beinhaltet die Macht, es kaputt zu machen. Keiner gewinnt in dieser Situation. Der einzige Weg, dass Linux weiterhin gut arbeitet, ist, **genug zu lernen, um zu wissen, was Sie tun**. Wenn man es einem Nutzer einfacher macht, an Funktionen herumzudoktern, die er nicht versteht, ist es nur wahrscheinlicher, dass er zufällig etwas kaputt machen wird.

Viertens:

Wo haben Sie im Text oberhalb einen Weg gesehen, dass FOSS überhaupt davon profitieren würde, wenn es viele typische Windowsnutzer anziehen würde?

Nehmen Sie sich Zeit, lesen Sie es noch einmal, wenn Sie mögen. Ich werde warten.

Die Richtschnur von Linux und FOSS ist, „gute Software zu machen“. Es geht **nicht** um „Windows-ersetzende Software herstellen“. Das einzige, das eine Horde typischer Windowsnutzer zu Linux beitragen wird, sind Beschwerden. Über was werden sie sich beschweren? *„Es funktioniert nicht wie Windows.“*

Nein, tut es nicht. Wenn es wie Windows funktionieren würde, würde Linux besch***** sein. Es wäre eine minderwertige Kopie, die niemand verwenden würde. Der Grund, warum Leute Linux so leidenschaftlich mögen, ist, dass es **nicht** so funktioniert wie Windows. Es **nimmt nicht an**, dass Sie ein ewig ungebildeter Anfänger sind, es **versteckt nicht** seine Abläufe und Mechanismen vor Ihnen.

Windows chauffiert sie herum; Linux gibt Ihnen die Schlüssel in die Hand und setzt Sie auf den Fahrersitz. Wenn Sie nicht fahren können, ist das Ihr Problem. Und Ihr Fehler. Viele Leute werden Ihnen helfen, wenn Sie fragen. Und wenn Sie einen Vorschlag machen, wie etwa einen Tempomaten einzubauen, könnten Sie auch irgendwohin kommen: Es belässt die Kontrolle beim Fahrer, aber nimmt ihm ein paar Dinge ab. Aber sie werden sehr schnell abgefertigt, wenn Sie versuchen, jeden zu überzeugen, dass Linux wirklich, wirklich einen Chauffeur braucht.

„Aber Linux würde so viel etablierter!“, schreit der Anfänger.

Das könnte sein. Aber wie viele Linux-Entwickler würden profitieren, wenn Linux etablierter würde? Linux ist frei im Sinne von Freibier. Keiner der Leute, die Linux erschaffen haben, profitiert davon, dass es eine größere Benutzerzahl hat. Keiner der Leute in den Linux-Foren profitiert davon, dass Linux eine größere Benutzerzahl hat. Linux‘ Ziel ist **nicht** „eine große Benutzerzahl zu bekommen“ - das ist das Ziel proprietärer Software.

Linux‘ Ziel ist es, ein wirklich gutes Betriebssystem zu schaffen. Die Entwickler sind beschäftigt mit dem Hinzufügen neuer Funktionen, dem Entfernen von Fehlern und der Verbesserung bestehender Implementierungen. Sie beschäftigen sich nicht damit, Plakatwerbung aufzuhängen, wie gut ihr Zeug ist. Das sollte Ihnen etwas darüber sagen, wo ihre Prioritäten liegen.

Anmerkung von Felix Schwarz:
Erfahrene Anhänger freier Software wird folgende Aussage bekannt vorkommen:
„Freie Software muss frei sein im Sinne von Freiheit und **nicht** wie Freibier."

Der Kern freier Software liegt also in der Freiheit, nicht darin, dass sie u.U. kostenlos abgegeben wird. Im vorangegangenen Absatz wird aber von monetären Anreizen gesprochen und de facto wird die meiste freie Software kostenlos angeboten. Insofern ist Linux **auch** frei im Sinne von Freibier.
Da dieser Begriff im Originalartikel genauso benutzt wurde, möchte ich die Begrifflichkeit hier nicht ändern. Ich persönlich hätte allerdings aber auch eine andere Formulierung wie z.B. „kostenlos" verwendet, um falsche Assoziationen zu vermeiden.

Und schauen Sie, was dieser Ansatz mit Linux' Benutzerbasis getan hat: Es hat sie vergrößert. Linux begann winzig und ist riesig geworden. Der Grund, warum Linux so großen Anklang gefunden hat? Weil der Fokus immer auf der Qualität lag. Die Benutzer, die von Linux angezogen wurden, wollten Freiheit und Qualität, die nur FOSS Ihnen geben kann. Linux wurde groß, weil es sich nicht darum kümmerte, wie groß es wurde. Entwickler waren nur damit beschäftigt, dass es funktioniert und zwar gut funktioniert, und so zogen sie Benutzer an, die ein Betriebssystem wollten, das funktioniert und zwar gut funktioniert.

Wenn man all das plötzlich wegwerfen würde und sich statt dessen darauf konzentrierte, sich darauf zu konzentrieren, dass Linux Windows ersetzt, würde dies genau die Sache zerstören, die Linux zu dem gemacht hat, was es heute ist. Es gibt da draußen Firmen, die Linux' Wachstum sehen und daran verdienen möchten. Sie sind frustriert über die GPL, die es ihnen sehr schwer macht, Linux zu Microsoft-Preisen zu verkaufen. *„Linux wird sterben, wenn es so offen bleibt,"* sagen sie, *„weil niemand Geld damit machen kann."*

Sie erkennen nicht, dass mit einem proprietären Linux die Gans geschlachtet würde, die die goldenen Eier legt. Linux wurde groß, weil es FOSS war und niemand versuchte, es zu einem Windows-Ersatz zu machen. Linux gedeiht, weil es Windows von einer Seite angreift, wo Microsoft es niemals besiegen kann: Offenheit und Qualität.

Für die meisten Windowsnutzer ist Linux eine schlechtere Windowskopie. Es hat weniger offensichtliche Funktionalität, weniger Integration und viel mehr Komplexität. Diese Art von Nutzern sieht Linux als ein schlechtes Betriebssystem. Und das ist richtig: Es erfüllt nicht ihre Bedürfnisse. Ihre Bedürfnisse sind ein Betriebssystem, das sehr einfach zu verwenden ist und alles tut, ohne dass sie etwas lernen müssten.

Windows wurde erschaffen für nicht technikkundige Nutzer. Unter diesen Nutzern ist die Wahrnehmung weit verbreitet, dass Linux schwer zu benutzen sei. Das ist nicht der Fall, aber es ist ein verständliches Missverständnis.

Linux ist tatsächlich erfreulich einfach zu benutzen. Wirklich. Es ist wirklich einfach zu benutzen. Der Grund, warum es nicht so wahrgenommen wird? Weil der Ausdruck *„einfach zu benutzen“* so weit deformiert wurde. Im alltäglichen Sprachgebrauch heißt *„einfach zu benutzen“* heute *„es ist einfach etwas zu tun, ohne vorher zu wissen, wie es zu machen ist“*. Aber das ist nicht wirklich *„einfach zu benutzen“*, oder? Das ist *„es ist einfach herauszufinden“*. Das ist wie der Unterschied zwischen:

- Ein Tresor mit einem Zettel darüber, auf dem steht „Sie öffnen diesen Tresor, indem Sie das Rad zur 32 drehen, dann 64, dann 18, dann 9 und dann den Schlüssel umdrehen und die Klinke nach unten drücken.“

und

- Ein Auto kann geöffnet werden, indem man auf der Fernbedienung den „Öffnen“-Knopf drückt.

Es ist weit einfacher, das Auto zu öffnen, richtig? Ein Knopf irgendwo in der Nähe des Autos im Gegensatz zu sehr spezifischen Drehungen. Aber es ist einfacher für jemanden, der nicht weiß, wie beide zu öffnen sind, den Tresor zu öffnen als das Auto: Der Tresor besitzt klare Instruktionen, während das Auto nur Knöpfe hat, **die nicht mal am Auto direkt zu finden sind**.

Mit Linux ist es genauso. Es ist einfach zu benutzen, wenn Sie wissen **wie**. Es ist einfach zu **benutzen**, aber nicht immer einfach zu **lernen**. Nur wenn Sie willens sind, Zeit in das Lernen von Linux zu investieren, werden Sie es einfach finden. Je weiter Sie eine Aufgabe in einfache Schritte zerlegen, umso mehr Schritte müssen Sie machen, um diese Aufgabe zu bewerkstelligen.

Nehmen wir als wirklich einfaches Beispiel, diese **beliebige** Übung: Sie wollen fünf Zeilen (Absätze) aus der Mitte eines Textdokuments ans Ende verschieben.

In MS Word, MS WordPad oder MS Notepad, alles „benutzerfreundliche“ Windows Texteditoren, der schnellste Weg, dies zu tun, ist:

```
· Strg-Shift-Runter
· Strg-Shift-Runter
· Strg-Shift-Runter
· Strg-Shift-Runter
· Strg-Shift-Runter
· Strg-X
· Strg-Ende
· Strg-V
```

(Angenommen, Sie verwenden die Tastatur, ansonsten diverse Mausklicks und ein verlässliches Autoscrolling, um den Text nach unten zu ziehen.)

Mit vi allerdings machen Sie:

```
· d5d
· Shift-g
· p
```

(Oder wenn Sie vi **wirklich** gut kennen, wird „:1,5m$“ auch funktionieren!)

Vi, der so *„benutzer-unfreundlich“* ist, wie es nur geht, schlägt Microsofts Produkte um Längen. Warum? Weil vi für Funktionalität entworfen wurde, während Microsoft versucht, *„benutzerfreundlich“* zu sein. Microsoft bricht alles in einfache Schritte runter und daher braucht es weit mehr Schritte, um die gleiche Aufgabe zu erfüllen.

Dadurch ist vi deutlich schneller und einfacher zu benutzen für fast alle Aufgaben zum Ändern von Text. Zumindest **so lange, wie Sie wissen, wie es zu benutzen ist**. Wenn Sie nicht wissen, dass *„d5d“* heißt *„Packe fünf Textzeilen in den Puffer und lösche sie aus dem Text“*, werden Sie Schwierigkeiten haben, vi zum Funktionieren zu bringen. Aber WENN Sie es wissen, dann kommen Sie mit vi am schnellsten zum Ziel.

Wenn irgendein Anfänger sieht, wie schnell und einfach ein erfahrener vi-Benutzer Dinge tun kann, wird er bereitwillig zustimmen, dass vi Word für das Ändern von Text überlegen ist. Anschließend versucht er es selbst. Er startet vi und erhält einen Bildschirm voller „~“s und wenn er tippt, erscheint nichts auf dem Schirm.

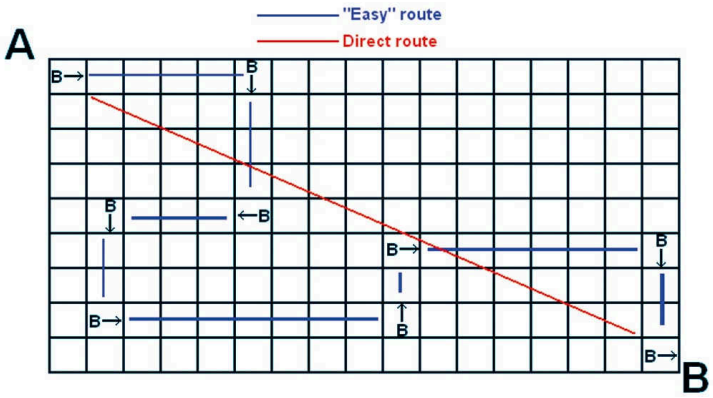
Er findet etwas heraus über die Texteingabe- und Kommando-Modi und versucht, vi mit einem begrenzten Wissen seiner Funktionen zu verwenden. Er kämpft, weil es so viele Dinge gibt, die er lernen muss, bevor er vi zum Funktionieren bringen kann. Dann beschwert er sich *„vi wäre so viel besser, wenn es so einfach zu benutzen wäre wie Word!“*.

Aber das tatsächliche Problem ist *„Ich weiß nicht, wie man vi benutzt und ich habe keine Lust, es zu lernen.“*. Aber das würde heißen, dass es sein Problem wäre, daher wirft er es der Software vor. Ungeachtet der vielen tausend Leute, die vi ohne jedes Problem nutzen: Es ist zu schwer zu benutzen und muss geändert werden!
Und glauben Sie mir, wenn er einen Texteditor schaffen könnte, der so *„benutzerfreundlich“* wie Word und so funktional wie vi ist, würde er nur mit Beifall empfangen werden. Vermutlich würde er sogar den Nobelpreis für Herausragende Intelligenz bekommen, weil niemand bisher dazu in der Lage war. Aber nur Jammern darüber, dass vi schwer zu benutzen ist, wird mit Hohn begegnet, weil es kein Problem mit vi ist, sondern ein Problem mit ihm.

Es ist, als ob man den Pinsel von Leonardo da Vinci kaufen und sich dann darüber beschweren würde, dass man immer noch nicht malen kann. Der Pinsel war es nicht, der die Mona Lisa gemacht hat, es war die Fähigkeit des Künstlers. Der Pinsel ist ein Werkzeug, das auf die Fähigkeiten des Benutzers angewiesen ist. Es gibt keinen anderen Weg, diese Fähigkeiten zu bekommen außer Übung.

Das gleiche gilt für vi. Das gleiche gilt für viele Linux-Programme, über die neue Benutzer sich beschweren, dass sie „zu schwer“ oder „nicht intuitiv genug“ sind, ganz gleich ob sie über einen Texteditor, einen Paketmanager oder die Kommandozeile selbst sprechen.
Bevor Sie darauf beharren, dass etwas in Linux verbessert werden muss, gibt es eine wichtige Frage: *„Haben erfahrene Benutzer ein Problem damit?“*
Wenn die Antwort *„Nein“* lautet, liegt das Problem bei Ihnen. Wenn andere Leute es erfolgreich verwenden, warum können Sie das nicht? Haben Sie sich die Zeit genommen, um die Benutzung zu lernen? Oder haben Sie erwartet, dass es für Sie funktioniert, direkt nach dem Wort „Start“?

„Benutzerfreundlichkeit“ und *„reine Funktionalität“* schließen sich aus. All die kleinen Knöpfe und Drop-Down-Menüs, die wichtig sind, um eine Software einfach benutzbar zu machen, sind nur Hindernisse, die dem erfahrenen Benutzer im Weg sind. Ein anderes Beispiel soll diesen Unterschied verdeutlichen: Man kann einen Weg von A nach B finden, indem man entweder Karte und Kompass benutzt oder einfach den Straßenschildern folgt. Letztlich wird man auch immer zum Ziel kommen, indem man den Schildern folgt, aber es wird doppelt so lange dauern, als wenn man den direkten Weg kennen würde.



Wenn ich den Wert einer Formel in Excel einfügen will, muss ich das über das Menü Bearbeiten ➜ Einfügen ➜ Spezial ➜ Wert einfügen tun. Ich **will mich nicht** durch all diese sch*** Menüs, Untermenüs und Dialogboxen durchklicken. Ich will einfach in der Lage sein, es zu tun. Um nicht ungerecht zu sein: Wenn ich die Tastenkürzel neu belegen und einige Makros aufzeichnen würde, **kann** ich Excel und Word dazu bringen, die meisten Dinge mit einem Mausklick zu machen.

Aber das ist nicht wirklich benutzerfreundlich, oder? Es zwingt den Benutzer immer noch dazu, eine Menge in die Software zu investieren. Linux fordert von Ihnen, dass Sie Zeit aufwenden, um die bestehenden Funktionalität zu lernen. Bei *„benutzerfreundlicher“* Software müssen Sie Zeit aufwenden, um die Funktionalität zu **erzeugen**.

Wenn das der Weg ist, den Sie bevorzugen, ist das in Ordnung, machen Sie weiter und tun Sie es auf diesem Weg. Aber verlieren Sie nie aus dem Auge, dass der Fehler in **Ihrer Ignoranz** liegt und nicht in der Software selbst. Alle Linux-Software ist höchst einfach zu benutzen, wenn Sie einmal wissen, wie sie zu verwenden ist. Wenn Sie es nicht wissen, wird es nicht einfach sein und das ist **nicht** der Fall, weil die Software fehlerhaft ist.

Jetzt könnten Sie ein Gefühl bekommen, dass Linux ein Gesinnungsproblem hat. Es will keine Benutzer, es will nicht das Leben für seine Nutzer einfacher machen... es ist nur für snobistische l33t h4xx0r5!

Nichts könnte ferner von der Wahrheit entfernt sein. Natürlich will Linux Benutzer haben! Und natürlich will es nicht Dinge schwierig machen. Ganz im Gegenteil: Schwierig zu bedienende Software ist per Definition schlechte Software.

Aber Sie müssen erkennen, dass Linux‘ Definitionen andere sein könnten als Ihre und auch andere als die der „traditionellen“ proprietären Softwarekultur.

Linux will Benutzer, **die Linux wollen**. Und das umfasst nicht nur den Namen. Es meint alles: Die freie Open-Source-Software, die Fähigkeit, an der Software zu basteln, die Möglichkeit, auf dem Fahrersitz zu sitzen, totale Kontrolle zu haben.

Das ist, was Linux **ist**. Das ist, worum es geht. Leute migrieren zu Linux, weil Sie Viren, BSODs [„Blue Screen of Death“, der blaue Bildschirm, wenn Windows komplett abgestürzt ist, Anm. von Felix Schwarz] und Spyware leid sind. Das ist verständlich. Aber diese Leute **wollen nicht** Linux. Sie wollen eigentlich nur **Windows ohne die Fehler**. Sie wollen nicht wirklich Linux. Warum sollte Linux also **sie** wollen?

Aber wenn sie Linux wegen der Viren und Spyware **ausprobieren** und **dann** entscheiden, dass **sie** die Idee eines Betriebssystems lieben, das sie kontrollieren... Dann wollen Sie Linux um seiner selbst willen. Und das ist der Punkt, an dem Linux auch sie will.

Bevor Sie entscheiden, ob Sie zu Linux wechseln wollen, fragen Sie sich *„Warum will ich wechseln?“*.

Wenn die Antwort ist *„Ich will ein Betriebssystem, das die ganze Macht in die Hände des Benutzers legt und von ihm erwartet, dass er weiß, wie sie zu gebrauchen ist.“*: Holen Sie sich Linux. Sie werden eine große Menge Zeit und Mühe investieren, bevor Sie bekommen, was Sie wollen, aber dann werden Sie schließlich belohnt mit einem Computer, der genauso so arbeitet, wie Sie es wollen.

ABER...

Wenn die Antwort lautet *„Ich will Windows ohne die Probleme“:* Machen Sie eine saubere Installation von Windows XP SP2, installieren Sie eine gute Firewall, installieren Sie ein gutes Anti-Viren-Programm, benutzen Sie niemals den IE zum Browsen im Web, aktualisieren Sie regelmäßig Ihr System, starten Sie nach jeder Software-Installation neu und lesen Sie etwas über gute Sicherheitsregeln. Ich habe Windows selbst benutzt von 3.1 über 95, 98, NT und XP und hatte **nicht einmal** einen Virus oder Spyware oder wurde gehackt. Windows **kann** ein sicheres und stabiles Betriebssystem sein, aber es ist auf Sie angewiesen, damit es so bleibt.

Wenn die Antwort lautet *„Ich will einen **Ersatz** für Windows ohne die Probleme“* : Kaufen Sie einen Apple Mac. Ich habe wundervolle Dinge über MacOS X „Tiger“ gehört und sie haben Hardware, die gut aussieht. Es wird Sie einen neuen Computer kosten, aber damit werden Sie bekommen, was Sie wollen.

In beiden Fällen **wechseln Sie nicht** zu Linux. Sie werden enttäuscht sein sowohl von der Software als auch von der Community. **Linux ist nicht Windows**.

Links

[1] http://geekblog.oneandoneis2.org/index.php?title=who_do_i_think_i_am&more=1&c=1&tb=1&pb=1
[2] http://www.varbusiness.com/sections/movers/mover.jhtml;jsessionid=2DBOI2IWW3CNQQSNDBGCKH0CJUMEKJVN?articleId=18804216&_requestid=1323686
[3] http://de.wikipedia.org/wiki/FOSS
[4] http://de.wikipedia.org/wiki/Hacker_(Begriffskl%C3%A4rung)
[5] http://www.ccc.de/hackerethics
[6] http://geekblog.oneandoneis2.org/index.php?title=the_great_ui_debate&more=1&c=1&tb=1&pb=1
[7] http://linux.oneandoneis2.org/blog/2005/11/coders-interfaces.html
[8] http://www.felix-schwarz.name/files/opensource/articles/Linux_ist_nicht_Windows/

Nachwort von Felix Schwarz

In der Tat ist diese Übersetzung ein gutes Beispiel für den klassischen Open Source-Entwicklungsprozess. Eines Tages im Januar 2006 hatte ich nachmittags mal 3-4 Stunden Zeit und habe diese genutzt, um eine Übersetzung ins Deutsche anzufertigen. Natürlich war sie nicht perfekt (sie ist es immer noch nicht) und enthielt diverse Fehler, aber man konnte sie lesen. Ich stellte die deutsche Version online und Dominic Humphries verlinkte sie. Im Laufe der Wochen lasen mehrere Tausend Besucher den Artikel und ab und an bekam ich kurze oder längere E-Mails mit Anmerkungen und Verbesserungsvorschlägen, die ich dann auch eingebaut habe.
So wurde dieser Artikel im Laufe der Zeit immer wieder verbessert und der Dank muss auch an die unten aufgeführten Personen gehen, die mit ihren hilfreichen Kommentaren dazu beigetragen haben.

Nachwort von Benedikt Sauter

Mit kleinen Änderungen habe ich den Text nahezu 1:1 übernommen. Ich hoffe er gefällt den meisten so wie mir.

Änderungen

22.06.2008 kleinere Verbesserungen der Übersetzung (Dank an Karlheinz, Patrik Fimml und Heiko Hartmann), noch eine weitere vi-Ergänzung (Dank an Sebastian Schaetzl)
20.06.2008 Hinweis auf die zweite Version des Originalartikels (Dank an Jeremy Boy, Nikolaus Klepp, Matthias Kretz für die Hinweise)
08.12.2007 kleine Grammatikfehler ausgebessert (Dank an fuesika und Wilhelm Kweton), tote Links korrigiert (Dank an Marek Macioschek, David Sieg und Daniel Schmid für die Hinweise)
14.10.2007 kleine Satzkorrekturen (Dank an Thomas Fiedler und Edgar Hoffmann), Erklärungen englischer Akronyme wie CLI und BSOD auf Anregung von Thomas Fiedler, weitere Anmerkung zu vi-Befehlen hinzugefügt.
01.10.2007 Link auf deutsche Lizenzseite, korrigierter Link auf Original-Blog (Dank an Tobias Hobmeier)
26.09.2007 diverse Fehlerbeseitigungen und Verbesserungen der Übersetzung (Dank an Heiko Kempe, Ralph Jud, Ute Köster und Miriam Kuethe)
23.09.2007 Verbesserung der Übersetzung „gutaussehend“ (Dank an Sebastian Linke)
26.08.2007 Verbesserung der Übersetzung: Maschine → Motor (Dank an Nico)
03.07.2007 bessere Formulierung bei Motorrad-“Schieflage“ (Dank an Uwe Walter)
11.06.2007 weitere Fehlerbeseitigungen (Dank an Dominik Geyer und Wolfgang Seebacher), Anmerkung zu Hackern und Crackern auf Anregung von Dennis Brakhane hinzugefügt.
24.01.2007 richtiger vi-Befehl (Dank an Jürgen B., Karsten W.), bessere Formulierung (Dank an Peter Henning), Anmerkung zu Freibier
28.11.2006 diverse Korrekturen (Dank an Bruno Holliger, Thomas Schneider, Sebastian Restel)
27.08.2006 „Stück Software“ ist Denglisch (Dank an Danny Tank)
18.08.2006 Korrekturen im Satzbau (Dank an Jon Kowal)
05.08.2006 ein paar Fehler beseitigt (Dank an Norman Steinbach)
25.07.2006 diverse Rechtschreibfehler beseitigt (Dank an Jacqueline Dürr und insbesondere Andreas Drop)
17.06.2006 Tippfehler verbessert (Dank Sebastian Wagner)
13.06.2006 kleine Rechtschreibkorrekturen (Dank an anonymous und Timo Taglieber)
05.06.2006 kleiner Rechtschreibfehler beseitigt
04.06.2006 kleiner Rechtschreibfehler beseitigt (Dank an Robert).
03.06.2006 kleinere Rechtschreibfehler beseitigt (Dank an Stefan Schoenemann), Link auf Originalartikel angepasst.
17.05.2006 diverse englische Originalpassagen gelöscht, kleine Satzänderung auf Grund eines Hinweises von Christian Rumpf)
12.04.2006 zahlreiche Verbesserungen der Übersetzung (großen Dank an David Wührer)
12.04.2006 kleine Satzänderung (Dank an Edward von Flottwell)
22.03.2006 Link auf deutsche Wikipedia geändert (Dank an anonymous)
15.01.2006 kleine Korrekturen und Verbesserungen (Dank an Jessica Campbell)
12.01.2006 initiale Übersetzung

The Rundfunker

Rundfunker (WLAN MP3 Player)

Von Christian Leberfinger (Projektarbeit Hochschule Augsburg)

Der Rundfunker ist ein stationärer MP3-Player, der das WLAN nach lokalen Audioquellen durchsucht und dort freigegebene Audio-Dateien abspielt. Er besitzt ein eingebautes 2-Wege-Lautsprecher-System, ein LC-Display und ein überraschend simples und gleichzeitig mächtiges Bedienkonzept. Dabei benötigt das Gerät keinerlei externe Peripherie, ein einfacher Stromanschluss genügt - die komplette Hardware befindet sich bereits im optisch ansprechenden Designer-Gehäuse mit edler Aluminium-Frontplatte. Unser Fünf-Mann-Team (Mathias Bauer, Christoph Beckmann, Christian Leberfinger, Stefan Loibl, Jan Peuker, Projektarbeit an der FH-Augsburg) hat die komplette Software konzipiert, implementiert und als Open-Source zur Verfügung gestellt; außerdem haben wir zwei komplett funktionstüchtige Prototypen gebaut.

Abbildung 1: Zwei-Wege-Lautsprechersystem

Die Idee, ein WLAN-Radio zu entwickeln, entstand aus dem Wunsch, die Vorteile eines Radiogeräts mit denen einer personalisierten Musiksammlung auf einem PC zu kombinieren. Ein UKW-Empfänger ist in der Regel kompakt, transportabel und unkompliziert zu bedienen; leider jedoch entspricht das Programm der verschiedenen Sender nicht unbedingt jedermanns Geschmack. Die mp3s auf dem heimischen PC würden eigentlich eine perfekte Alternative zum Radio-Einheitsbrei darstellen, allerdings haben sie den großen Nachteil, dass sie in der Regel auf einem mehr oder weniger unbeweglichen Rechner liegen und damit kaum außerhalb der Zimmergrenzen gehört werden können (ohne die Nachbarn zu verärgern).

Der Rundfunker ist hier eine elegante Lösung. Er ist ein mp3-Player in Radioform, der drahtlosen Zugriff auf alle mp3s bietet, die im lokalen Rechnernetz freigegeben sind. Auf diese Weise unterliegt er nicht den Volumenbeschränkungen der beliebten mp3-Player in USB-Stick-Form, die heutzutage Platz für etwa ein Gigabyte Musikdaten bieten: der Rundfunker kann über WLAN theoretisch auf mehrere Terabyte Musik zurückgreifen. Damit übertrifft er selbst portable mp3-Player mit eingebauten Festplatten (mit etwa 60 Gigabyte) um Längen. Der Rundfunker ist zwar tragbar, aber nicht für den portablen Gebrauch wie bei einem Walkman gedacht. Sein Revier ist die Wohnung. Dies bringt diverse Vorteile mit sich. Das Gerät muss nicht mit einem winzigen Gehäuse à la iPod auskommen, sondern besteht aus einer kompakten Box von 21×21×18 Zentimetern, die genügend Volumen für das integrierte Zwei-Wege-Lautsprechersystem, eine edle Frontplatte mit leicht zu bedienenden Knöpfen und ein übersichtliches LC-Display bietet.

Ein typischer Anwendungsfall des Rundfunkers wäre folgender: im Arbeitszimmer im ersten Stock steht der PC mit einer Festplatte voller mp3-Dateien, die in der Küche im Erdgeschoss ohne Kabelverbindung wiedergegeben werden können. Dabei hat der Hörer zahlreiche Möglichkeiten, die Wiedergabe zu beeinflussen. Über das simple Bedienkonzept kann er beispielsweise sein Lieblingsalbum oder sein favorisiertes Genre suchen oder einfach durch die gesamte Musiksammlung blättern.

Abbildung 2: Zwei-Wege-Lautsprecher Hochtöner

Um den Rundfunker in Betrieb zu nehmen, muss er lediglich über die Steckdose mit Strom versorgt werden. Damit er auch mp3-Dateien zum Wiedergeben finden kann, muss der oder die Quell-PCs angeschaltet sein und die Ordner, die die Musikfiles beinhalten, im Dateisystem freigegeben werden. Die Konfiguration des Systems (z.B. Einstellen der IP-Adressen) erfolgt über ein übersichtliches Web-Interface, das als zusätzliches Feature auch die komplette Fernsteuerung des Rundfunkers ermöglicht.

Mainboard

Unser Anliegen war es, ein komplett passiv gekühlts Motherboard zu verbauen. Wir haben uns für das VIA Epia MS 10000E LVDS entschlossen, da es COM- und Audio-Ports als Steckbrückenverbinder, einen PCI-Slot (für die WLAN-Karte) und einen Compact-Flash-Einschub bietet. Von der Compact-Flash-Karte booten wir Troubadix, eine selbst angepasste Knoppix-Variante.

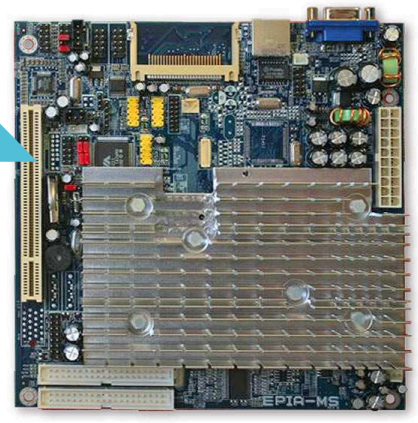


Abbildung 3: Mini-ITX Motherboard

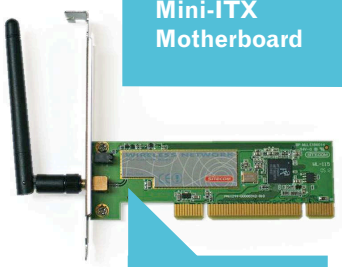


Abbildung 4: WLAN-Karte

Betriebssystem (Troubadix)

Als Betriebssystem kommt Troubadix, ein selbst gezüchtetes Knoppix-Derivat zum Einsatz, das im Gegensatz zu anderen Mini-Distributionen den 2.6er Kernel in der vollen Knoppix-Ausführung, MySQL 4.1, Apache2, PHP5, OpenSSH und die

Abb. 5: Troubadix on Compact-Flash card

Sun-JRE 1.4.2 beinhaltet - allerdings kein X und dementsprechend auch keinen Window-Manager. Da Troubadix im Rundfunker von einer Compact-Flash-Karte gebootet wird, sind keine beweglichen Teile im Gehäuse, die bei einem Transport beschädigt werden könnten.

Troubadix ist nur ungefähr 200 MB groß - der restliche Speicherplatz auf der CF-Karte steht für unser Java-Programm and die dahinterliegende Datenbank zur Verfügung.

LCD, Jog-Dial und LEDs an das Epia anschließen

Verbunden werden Java-Programm und Hardware über eine kleine Platine, die wir selbst entwickelt haben. Darauf sitzt ein von uns programmierter Microcontroller, der über die serielle Schnittstelle mit dem Motherboard kommuniziert. Der Microcontroller kann so die Benutzereingaben an unsere Software weiterleiten und empfängt im Gegenzug die Inhalte, die er an das LC-Display weiterleiten soll. Die Entwicklung dieser Platine und der Software, die auf dem Microcontroller läuft, war die Aufgabe, die uns anfangs am schwersten gefallen ist, da wir uns mit solchen Dingen noch nie zuvor beschäftigt hatten. Wir haben Eagle eingesetzt, um die Schaltkreise zu entwerfen und entwickelten die Microcontroller-Software für unseren Atmel Atmega 168 mit AVR Studio. Viel Zeit kostete die Fehlentscheidung, den Atmega 168 zur verwenden, da dieser leider nicht genügend Anschluss-Pins zur Verfügung stellt.

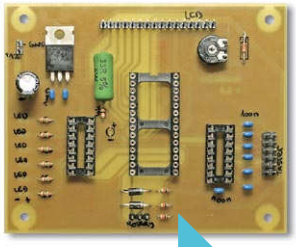


Abb. 7: Schaltung Mikrocontroller

Da wir uns außerdem für ein uC-Modul entschieden hatten, dass relativ teuer (ca. 50 EUR) war, wollten wir es nicht einfach durch einen größeren uC ersetzen (Atmega 32 oder ähnlich). So mussten wir die Anschlussmöglichkeiten mithilfe eines Schieberegisters und eines 8-to-3-Decoders „aufbohren“. Unglücklicherweise haben wir das Datenblatt des Schieberegisters anfänglich auch noch missverstanden, so dass wir auch hier nachträglich noch kleine Löt-Workarounds durchführen mussten ;)

Konfiguration

Grundsätzliche Einstellungen können komfortabel über ein in PHP umgesetztes Webinterface getätigt werden. Außerdem kann der Rundfunker über ein Java-Applet in diesem Webinterface komplett fernbedient werden.

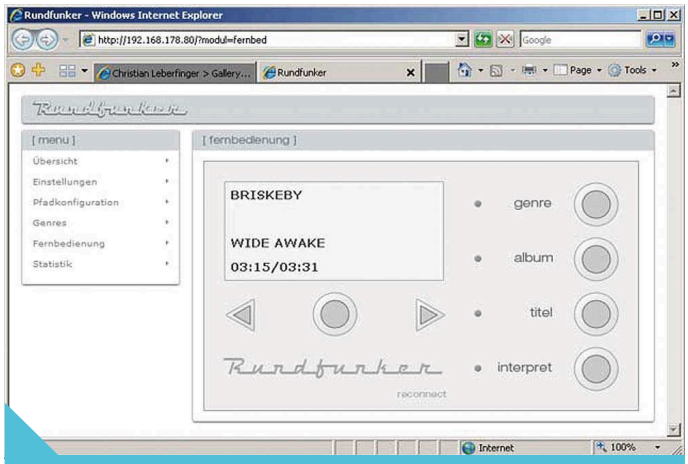


Abb. 6: Configuring the Rundfunker via Webinterface

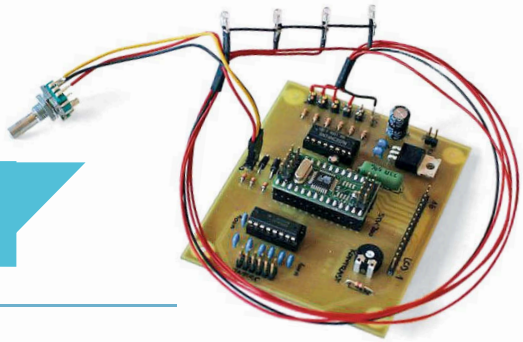


Abbildung 8: Bedienung

Benutzer-Anwendung

Auf diesem System aufgesetzt ist ein selbst konzipiertes und implementiertes Java-Programm, das sowohl das Suchen und Abspielen der Audiodateien aus dem WLAN als auch die Bedienung des Geräts ermöglicht. Bei der Konzeption dieser Software wurde besonders auf eine sauber objektorientierte Struktur und leichte Erweiterbarkeit geachtet. Normalerweise würden wir jetzt ein paar Screenshots unseres Programms herzeigen; in diesem speziellen Fall gibt es natürlich keinen „Screen“ im herkömmlichen Sinne - deswegen einige Photos von Inhalten unseres LC-Displays:

- **standard-screen during playback**
  Der Standard-Bildschirm während Liedwiedergabe (vorheriger/nächster Song bei Betätigung des Jog-Dials)
- **list of artists**
  Interpreten-Liste - durchblättern und auswählen mittels Jog-Dial (ALPS rotary encoder)
- **Rundfunker speller**
  Der komfortabelste Weg, einen Interpreten aus einer großen Liste zu wählen: Rundfunker Speller

Hardware/Software

Software:

- die Microcontroller-Software zur Ansteuerung der Hardware (in C implementiert)
- das Java-Programm zum Handling des Bedienkonzepts und Abspielen der Audio-Dateien
- die PHP-Webanwendung für Fernbedienung und Konfiguration des Rundfunkers
- Troubadix, unsere selbst entwickelte Knoppix-Variante

Hardware:

- ein VIA Epia Mini-ITX-Motherboard mit 1GHz CPU und 256 MB RAM
- ein 4 x 20 Zeichen LC-Display
- eine selbst entwickelte Platine mit elektronischen Bauteilen zum Anschluss von Display, Tastern, LEDs und Jog-Dial
- ein selbst programmierter Microcontroller zur Ansteuerung der Hardware
- ein 2-Wege-Lautsprecher-System aus dem Car-Hifi-Bereich, das für gute Klangqualität sorgt

.

Gehäuse

Das Gehäuse des Rundfunkers haben wir aus MDF-Platten (mitteldichte Faserplatte) aus dem Baumarkt gebaut. Wir hatten es zuvor in einem 3D-Programm entworfen, dann gebaut und anschließend seidenmatt-weiß lackiert. Die edle Aluminium-Frontplatte, die wir entworfen hatten, wurde von einer Spezialfirma aus Berlin gefertigt.
Zwei Werkzeuge, die wir beim Bau des Gehäuses eingesetzt haben: Handoberfräse und Stichsäge.

Abb. 9: Schraubzwingen

Abb. 10: Stichsäge in Aktion

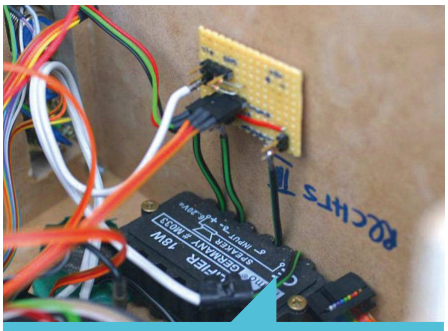


Abbildung 11: 18-Watt Verstärker

Wir haben einen kleinen 18-Watt-Verstärker für die Ansteuerung unserer 2-Wege-Lautsprecher von Pioneer eingebaut.

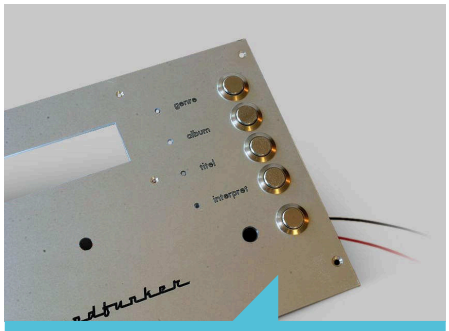


Abb. 12: Frontplatte des Players

Abb. 13: Ein edles blaues Display

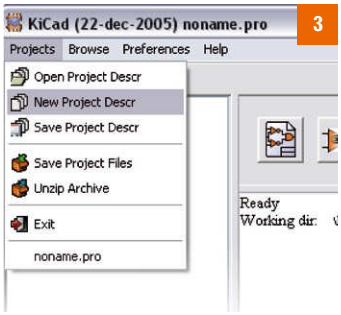
Projekt-Management

Alle Entwicklungsschritte wurden mit einem Ticketsystem (Trac) koordiniert, vom Versionierungssystem Subversion verwaltet und im angebundenen Wiki dokumentiert. Alle Quelltexte, Schaltpläne und Dokumente, die während der Entwicklungsphase angefallen sind, stehen auf unserem Projekt-Server für jeden zum Einblick:
https://developer.berlios.de/projects/rundfunker/

KiCad Step by Step Tutorial

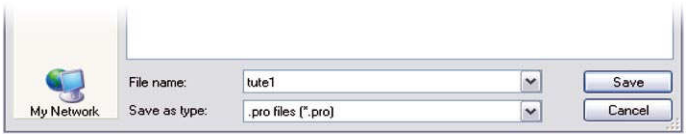
David Jahshan, <kicad@iridec.com.au> ©2006

KiCad is a open source (GPL) integrated package for schematic circuit capture and PCB layout.
Before you start, you will need an installed copy of KiCad.
This tutorial assumes that KiCad is installed at **C:\Kicad**.
You can download a copy from http://www.lis.inpg.fr/realise_au_lis/kicad/
Installation instructions are available on the website under Infos: Install

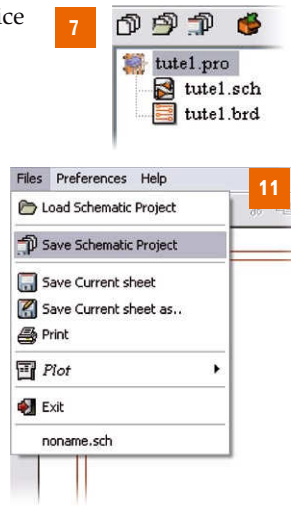


1. Run "**KiCad.exe**".
2. You are now in the Main Window.
3. Create a new project: "**Projects**" → "**New Project Descr**".
4. Click on "**Create New Folder**" button, and name the new folder "**tute1**".

5. Open the new directory by double clicking on it.
6. Enter the name of the project in "**File Name**", in this tutorial we will call it "**tute1**".

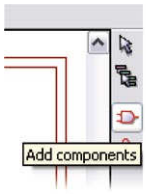


7. Click on "**Save**". You should notice the project name should change to "**tute1**".
8. Double click on "**tute1.sch**".
9. An "Infos" window will appear informingyou it is a new project. Click "**OK**".
10. You are now in the "**EESchema**" window.This window is used for entering schematics.
11. You should first save the schematic project: "**Files**" → "**Save Schematic Project**".
12. Click the "**page settings**" button on the top toolbar.

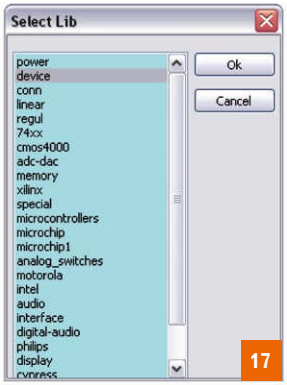


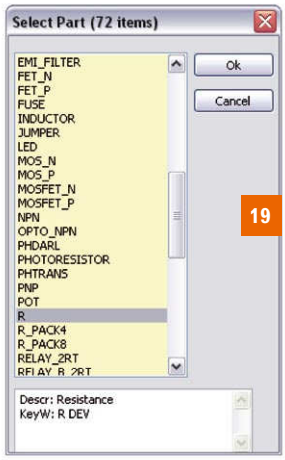
12

13. Select the "page size" as "**A4**" an the "**Title**" as "**Tute 1**".
14. Click on the "**Add components**" button found in the right toolbar.

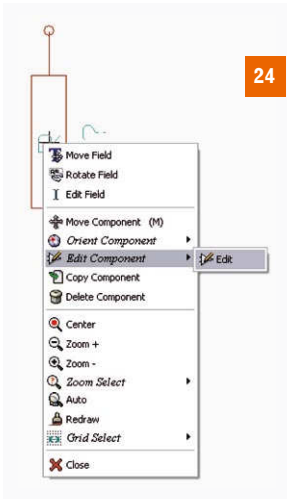


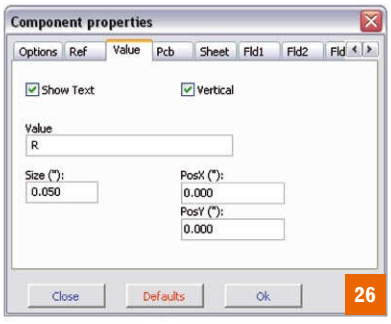
15. Click in the middle of the screen where you want to place your first component.
16. A "**Component selection**" window will appear.
17. Click on "**List All**". The "**Select Lib**" window will appear.
18. Double click on "**device**".
19. The "**Select Part**" window will appear.



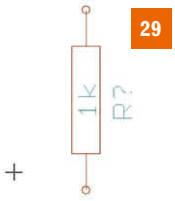


20. Scroll down and double click on "**R**".
21. Press ‚**r**' on the keyboard. Notice how the component rotates.
22. Place the component in the middle of the sheet by left clicking where you want it.
23. Click on the magnifier twice to zoom in on the component.
24. Right click in the middle of the component.
25. Select: "**Edit Component**" → "**Edit**".
26. The "**Component properties**" window will appear.





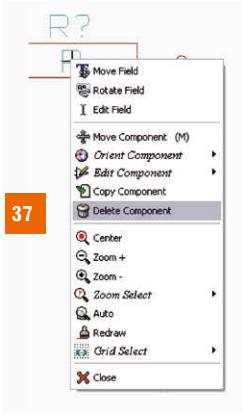
27. Select the "**Value**" tab.
28. Replace the current "**Value**" "**R**" with "**1k**".
29. Click "**OK**".



29

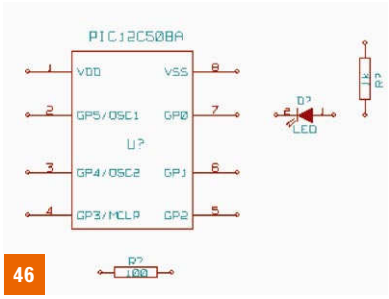
33

30. The value inside the resistor should now be "**1k**" .
31. Place another resistor by clicking on the place you want the resistor to appear.
32. The "**Component selection**:" window will appear.
33. The resistor you previously chose is now in your history list appearing as "**R**".
34. Click on "**R**".
35. Place the resistor on the page
36. Repeat and place a third resistor on the page.
37. Right click on the second resistor
38. Click on "**Delete Component**". This should remove the component.
39. Right click on the third resistor. Select "**Move Component**".



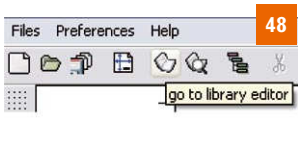
37

40. Reposition the component left click to drop.
41. Repeat steps 24 to 27 on the third resistor and replace "**R**" with "**100**"
42. Repeat steps 14 to 20, however this time select "**microcontrollers**" instead of "**device**" and "**PIC12C508A**" instead of "**R**".
43. Press ‚**y**' and ‚**x**' on the keyboard. Notice how the component is mirrored on it's x and y axis. Press ‚**y**' and ‚**x**' again to return to it's original orientation.
44. Place the component on the page.
45. Repeat steps 14 to 20, however this time choose "**device**" and "**LED**".
46. Organise the components on the page in the following way:
47. We will now add a component to the library.



46

48. Click on the "**go to library editor**" button on the top toolbar.
49. This will open the "**Libedit**" window.
50. Click on the "**Select working library**" button.
51. In the "**select lib**" window click on "**conn**".
52. Click on the "**New part**" button.
53. Name the new part "**MYCONN3**".

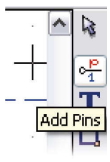


48

49

52

54. Enter the prefix as "**J**", and number of parts as "**1**".
55. If the warning "**has a convert drawing**" appears click "yes".
56. In the middle of the screen the name of the component should appear.
57. Click on the magnifier twice to zoom in on it.
58. Click on the "**Add Pins**" button on the right toolbar.
59. Left click on the screen where you want the pin.
60. In the "**Pin Properties**" box enter pin name as "**VCC**", enter pin number as "**1**".

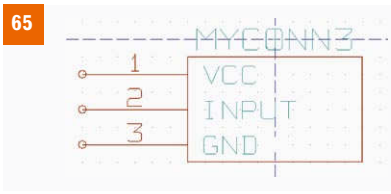


58

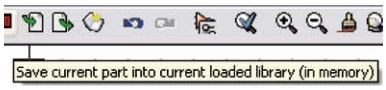
61. Select "**Electrical Type**" as "**Power Out**" then click "OK". Then place the pin by clicking at the location you would like it to appear.
62. Repeat steps 59 to 61, this time "**Pin Name**" should be "**INPUT**" and ‚**Pin Number**" should be "**2**". **Electrical Type**" should be "**Input**".
63. Repeat steps 59 to 61, this time "**Pin Name**" should be "**GND**" and "**Pin Number**" should be "**3**". "**Electrical Type**" should be "**Power Out**".
64. Arrange the pins and label as shown below step 65.
65. Click on the "**Add rectangle**" button. By left clicking and holding the button down place a rectangle around the pin names.
66. Click on "**Save current part into current loded library (in memory)**" on the top toolbar.
67. Click on "**Save current loaded library on disk (file update)**" on the top tool bar.
68. Click "**yes**" on the confirmation message.
69. You can now close the "**Libedit**" window.
70. Return back to the "**EeSchema**" window.
71. Repeat steps 14 to 20, however this time choose "conn" and "**MYCONN3**".
72. Your newly created part will appear. Choose a location near the second resistor to place this component. Press the ‚**y**' key to mirror it on the y axis.
73. The component identifier "**J?**" will appear under the "**MYCONN3**" label. Right click on "**J?**" and click on „**move field**". Reposition "**J?**" to under the pins.
74. Click on the "**Add powers**" button on the right toolbar.
75. Click above the pin of the 1k resistor.
76. In the "**Component Selection**" click on list all.
77. Scroll down and select "**VCC**" in the "**Select Part**" window.
78. Click above the pin of the 1k resistor to place the part.
79. Click above the VDD pin near the microcontroller.
80. In the "**Component Selection history**" select "VCC" and click again next to the VDD pin.
81. Repeat again and place the VCC pin above the VCC pin of "**MYCONN3**".
82. Repeat steps 74 to 76 but select GND this time.
83. Place the GND pin under the GND pin of "**MYCONN3**".
84. Place the GND symbol little to the right and below the VSS pin of the microcontroller.

61

64

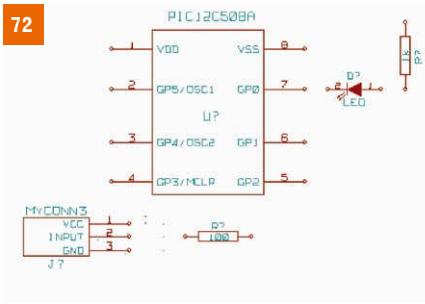


65



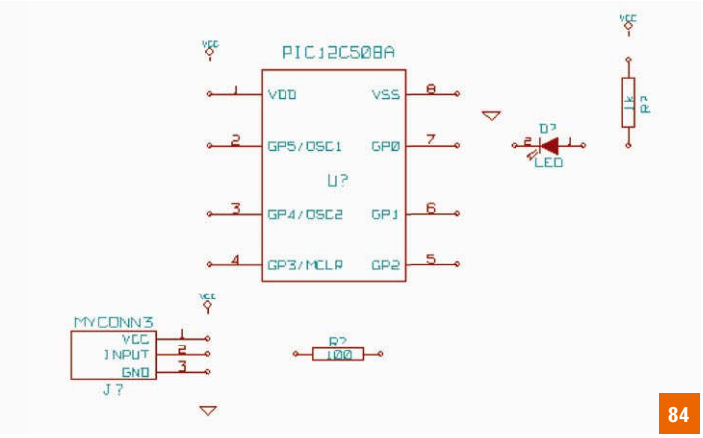
66

67



72

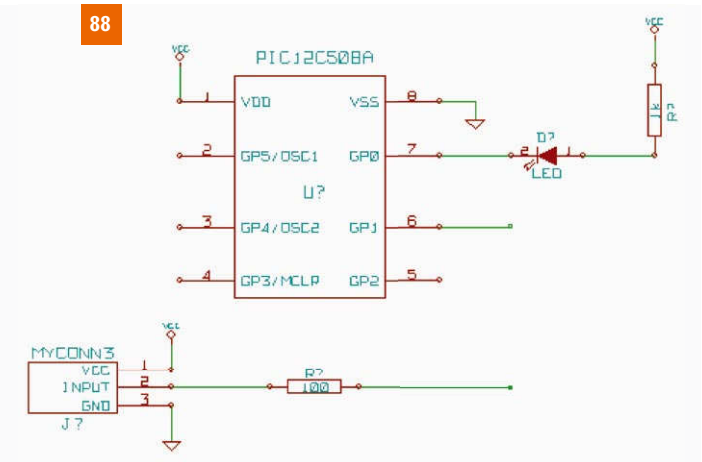
74



84

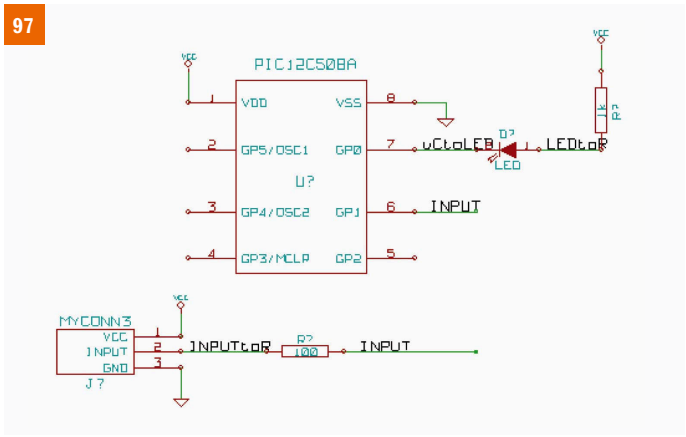
85. Click on the "**Add wires**" on the right toolbar **Careful not to pick "**Add bus**" which appears directly beneath but has thicker lines**.

86. Left click on the little circle on the end of pin 7 of the microcontroller and then on the little circle on pin two of the LED.
87. Repeat process to wire up the other components as below.
88. When wiring up the VCC and GND symbols, the wire should touch at the bottom of the VCC symbol, and in the middle top of the GND symbol.



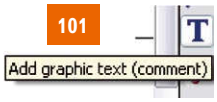
89. Label the nets by clicking on the "**Add wire or bus label**" button on the right toolbar.

90. Click in the middle of the wire between the microcontroller and the LED.
91. Enter the name "**uCtoLED**".
92. Click near the circle (little to the right) of pin 7 to place the net name.
93. Name the wire between the resistor and the LED to "**LEDtoR**".
94. Name the wire between "**MYCONN3**"to the resistor as "**INPUTtoR**".
95. Name the line on the right of the 100 ohm resistor as "**INPUT**".
96. Name the line from pin 6 as"**INPUT**".This creates an invisible connection between the two pins labelled "**INPUT**". This is a useful technique when connecting wires in a complex design where drawing the lines wouldmake the drawing very messy.
97. You do not have to label the VCC and GND lines, the labels are implied from the power objects they are connected to.

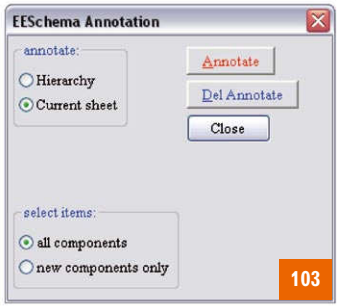


98. The program automatically checks for errors therefore any wires that are not connected may generate a warning. To avoid these warnings you can instruct the program that the unconnected wires are deliberate.

99. Click on the "**Add no connect**" flag button on the right toolbar.
100. Click on the little circle at the end of lines 2, 3, 4 and 5.
101. To add comments on the schematic use the "**Add graphics text (comment)**" on the right toolbar .



102. The components now need to be given unique identifiers. To do this clickon "**Schematic Annotation**" button.
103. In "**EESchema Annotation**" select "**Current Sheet**" and "**all components**".
104. Click on "**Annotate**".
105. Click "**yes**" for the confirmation message.
106. Notice how all the "**?**" on the components have been replaced with a number. Each identifier is unique. In our example "**R1**", "**R2**", "**U1**", "**D1**" and"**J1**".

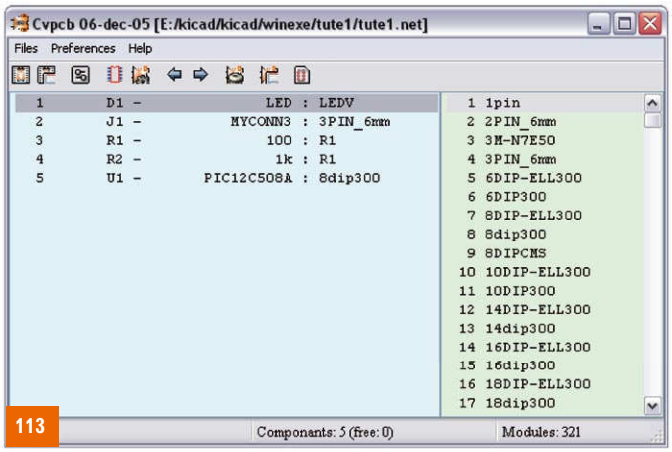


107. Click on the "**Schematic Electric Rules Check**" button. Push the "**Test ERC**" button.

108. This will generate a report to inform you of any errors or warnings such as wires being disconnected. You should have 0 Errors and 0 Warnings. A small green arrow will appear in the location of the error if you have made a mistake. Check "**Write erc report**" and press the "**Test ERC**" button again to receivemore information about the errors.

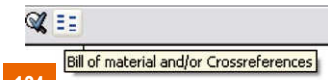
109. Click on "**Netlist generation**" on the top toolbar.
110. Click "**Netlist**" then on "**save**" to the default file name.
111. Click on "**Run Cvpcb**" on the top toolbar.
112. Cvpcb permits you to link footprints to components.

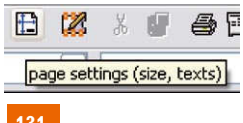
113. In the light blue screen select "D1" and scroll down in the light green screen to"**LEDV**" and double click on it.
114. For "**J1**" select the "**3PIN_6mm**" footprint.



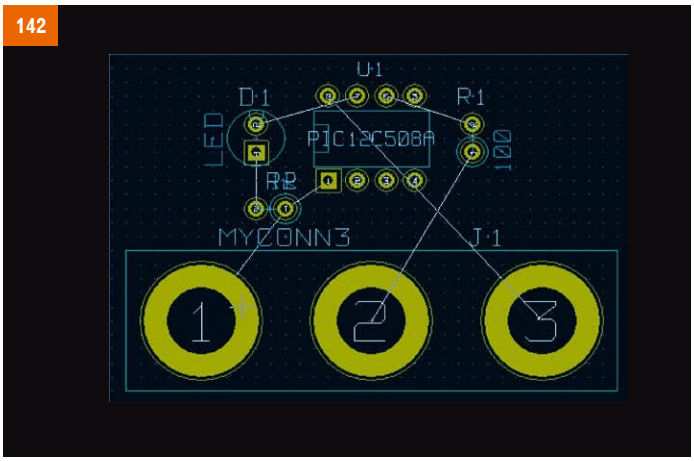
115. For "**R1**" and "**R2**" select the "**R1**" footprint from the light green screen.
116. Select 8dip300 for "**U1**".
117. Click on "**files**"➔ "**Save netlist**". The default "**tute1.net**" is fine therefore click save.
118. Save the project by clicking on "**files**"➔ "**Save Schematic Project**".
119. Switch to KiCad main window.
120. Select "**Browse**"➔ "**Browse Files**".
121. If an error message appears, choose your text browser. Most computers have one at "**c:\windows\notepad.exe**".

122. Select the "**tute1.net**" file. This will open your netlist file. It describes which components and which pins are connected to which pins.
123. Now return back to the "**EeSchema**" window.
124. To create a bill of materials click on the "**Bill of materials**" button on the top toolbar.
125. Click on "**Create List**" and the on "**Save**".
126. To view the file repeat step 120 and select "**tute1.lst**".
127. Now click on the "**Run Pcbnew**" button on the top toolbar.
128. The "**Pcbnew**" window will open.
129. Click "**OK**" on the error message for the file not existing.
130. Click on "**files**" ➔ "**Save board**".
131. Click on "**page settings**" button on the top toolbar.
132. Select "**paper size**" as"**A4**" and enter "**title**"as "**Tute 1**"
133. Click on "**Dimensions**" ➔ "**Tracks and Vias**".
134. Set the settings so that they correspond to your PCB manufacturing capabilities.(consult your PCB manufacturer for this information.) For our example increase clearance to „**0.0150**".
135. Click on the "**Read Netlist**" button on the top toolbar.

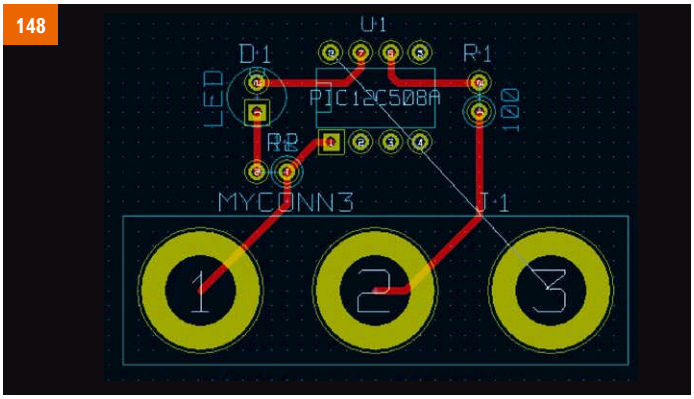




136. Click the "**Select**" button select "**tute1.net**" and click on "**open**" and then click the "**Read**" button. Then click the "**Close**" button.
137. The components will be placed in the top left hand corner just above the page, scroll up to see the components.
138. Right click on a component select "**move component**" and position it in the middle of the page.
139. Repeat previous step till all the components are in the middle of the page.
140. Make sure that the "**General ratsnest not show**" button is on.
141. This will display the ratsnest, which is a set of lines showing which pin connects to which other pin.
142. Move the components around till the ratsnest until you minimise the number ofcrossovers.



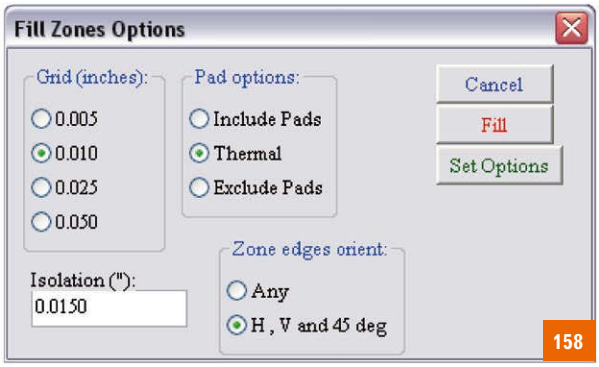
143. If the ratsnest disappears or the screen gets messy right click and click "**redraw**".
144. Now we will connect up all except the ground wires on the "**component side**" (top layer).
145. Click on the "**Add Tracks an vias**" button on the right toolbar.
146. Select "**Component**" out of the drag down menu on the top toolbar.
147. Click in the middle of pin 1 of "**J1**" and run a track to the pad "**R2**".
148. Repeat this process till all wires except pin 3 of J1 is connected.
149. In the drag down menu on the toptoolbar select Copper (bottom layer).
150. Click on "**Add tracks and vias**" button (step 145).
151. Draw a track between pin 3 of J1 and pin 8 of U1.



152. Click on "**Net highlight**" button on the right toolbar.
153. Click on pin 3 of J1. It should turn yellow.
154. Click on "**Add Zones**" button on the right toolbar.
155. Trace around the outline of the board.
156. Right click inside the area you have just traced.
157. Click on "**Fill Zones**".
158. Select "**Grid**" "**0.010**", "**Pad options:**" "**Thermal**", "**Zone edges orient:**" "**H,V**" and then click on "**Fill**".

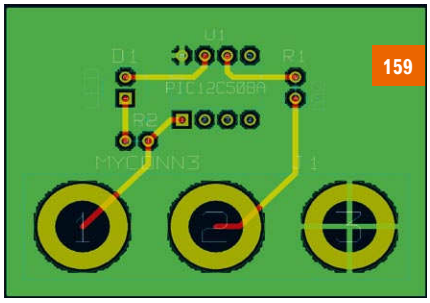
152

154



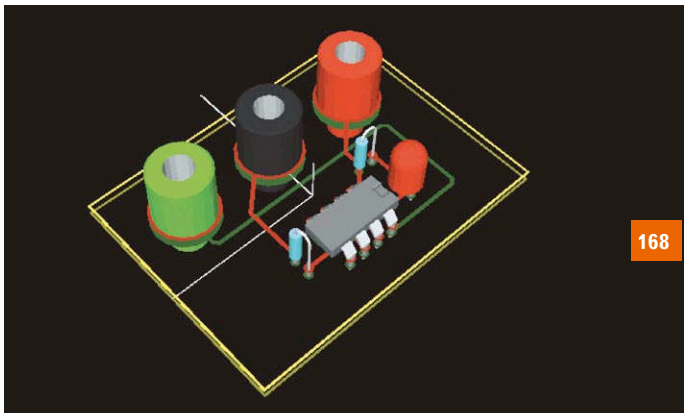
158

159. Your board should look like this.
160. Now select "**Edges Pcb**" from the dragdown menu in the top toolbar.
161. Select the "**Add graphic line or polygon**" button on the right toolbar.
162. Trace around the edge of the board but remember to leave a small gap between the edge of the green and the edge of the PCB.
163. Run a design rules check by clicking on "**Pcb Design Rules Check**".
164. Click on "**Test DRC**". There should be no errors.
165. Click on "**List Unconn**". There shouldbe no unconnected.
166. Save your file by clicking on "**files**" → "**Save board**".


159

161

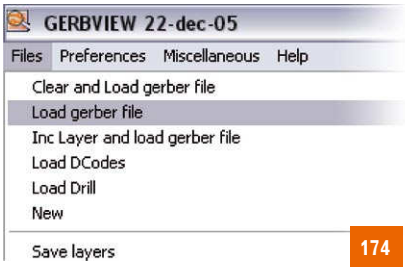
163


168

167. To view your board in 3d click on "**3D Display**" → "**3D Display**".
168. You can drag your mouse around to rotate the PCB.
169. Your board is complete, to send it off to a manufacturer you will need to generate a GERBER file.
170. Click on "**files**" → "**plot**".
171. Select GERBER as the "**plot format**"and click on plot.
172. To view GERBER files go to the main KiCad window.
173. Click on the"**Gerb View**" button.
174. Click on "**files**" → "**Load GERBER file**".
175. Select the file named "**tute1_Copper.pho**" and then on "open".
176. On the drag down menu select"**Layer2**".
177. Repeat steps 174 and 175 but this time load "**tute1_component.pho**".

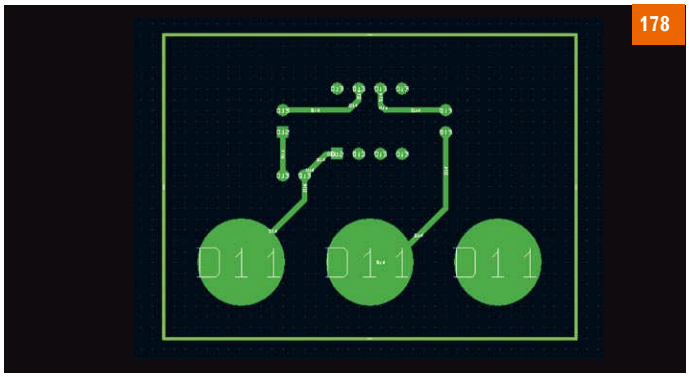
167

173



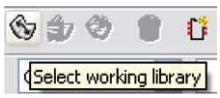
174

178. Repeat steps 176 but choose **Layer3**"then steps174 and 175 but this timeload "**tute1_SlkSCmp.pho**".


178

179. This way you can examine the layers that will be sent to production. There is a extensive footprint library with KiCad, however on occasion you might find that the footprint that you need is not in a KiCad library. Follows are some steps for creating a surface mount footprint in KiCad.
180. To create a new PCB footprint switch back to "**PCBnew**".
181. Click on "**Open Module Editor**" on the top toolbar.
182. This will open the "**Module Editor**".
183. Click on "**select working library**" on the top toolbar.
184. For this exercise select the "**connect**" library.
185. Click the "**New Module**" button on the top toolbar.
186. Enter "**MYCONN3**" as the "**modulereference**".
187. In the middle of the screen a "**MYCONN3**" label will appear.
188. Under the label will be "**VAL****".
189. Right click on "**MYCONN3**" and move above "**VAL****".
190. Right click on "**VAL****", select "**Edit Text Mod**" and rename it to "**SMD**".
191. Check the "**no display**" box.

181

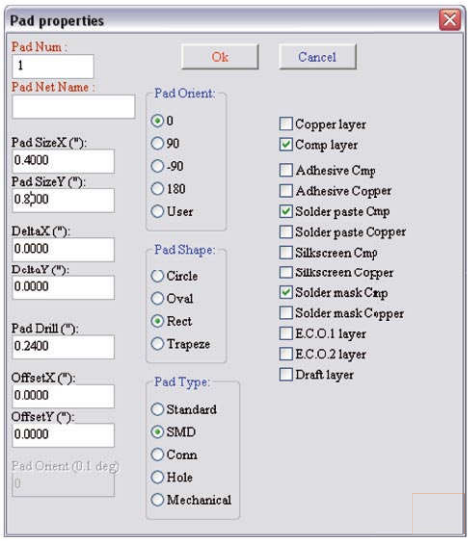


183

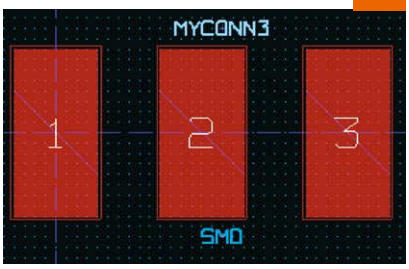
185

192. Select the "**Add Pads**" on the right toolbar.
193. Click on the screen to place the pad.
194. Right click on the new pad and click "**edit**".
195. Set the "**Pad Num**" to "**1**", "**Pad Size X**" to"**0.4**","**Pad Size Y**" to "**0.8**", "**Pad Shape**" to "**Rect**"and "**Pad Type**" to "**SMD**". Click "**Ok**".
196. Click on "**Add Pads**" again and place two more pads.

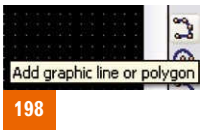
192



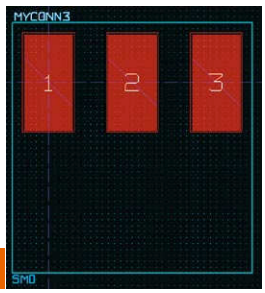
196



197. Move the "**MYCONN3**" and "**SMD**" labels out of the way so it looks like above.
198. Click on "**Add graphic line or polygon**" button in the right toolbar.
199. Draw an outline of the connector around the component.
200. Click on "**Save Module in working directory**" on the top toolbar.



198



199

200

201. You can now return to PCB new and click on "**Add modules**" button on the right toolbar.
202. Click in the screen, and the module name window will pop-up.
203. Select the module "**MYCONN3**" and place it on your PCB design.

201

This has been a quick tutorial on most of the features in KiCad. For more detailedinstructions there is a detailed help file which can be accessed through all KiCad modules. Click on help → help.

Warpzone

Der Hackerspace in Münster

Hauptautoren Marcel Wunderlich & Philipp Neuhaus | www.warpzone.ms

Schon seit Anfang des Millenniums wurde in Münster aktiv die Hackerkultur, also der kreative Umgang mit Technik, gepflegt. Dies geschah im „Chaostreff Münster / Osnabrück“ [1], der dem Chaos Computer Club nahe steht. Nun existiert seit August 2009 in Münster noch eine weitere hackernahe Vereinigung: die Warpzone2) - Münsters erster Hackerspace.

Was ist ein Hackerspace und ist das nicht illegal?

Diese Frage stellen sich sicherlich viele, besonders die, die wenig bis keinen Kontakt mit der Hackerszene haben. In dieser bezeichnet man mit „Hacker“ nämlich nicht jemanden, der in fremde Computersysteme einbricht, um sich persönliche Vorteile auf Kosten anderer zu verschaffen, sondern jemanden, der hohe Fähigkeiten in einem Gebiet besitzt und diese kreativer einsetzt, als die meisten Anderen. So lässt sich beispielsweise Albert Einstein als einer der prominentesten Hacker der Physik bezeichnen, da er mit seiner Relativitätstheorie auf völlig neue Art und Weise tiefe Erkenntnisse über das Wesen unserer Welt brachte.

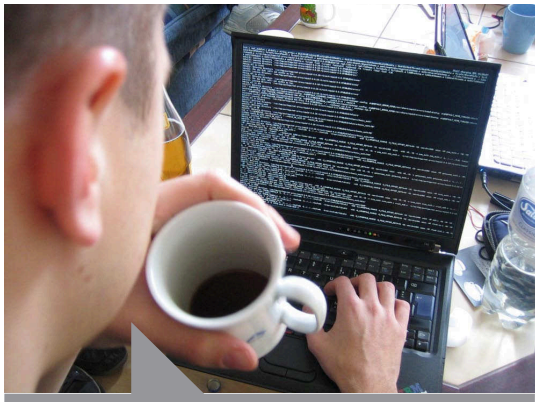


Abb. 2: Hacking mit Kaffee in der Chillzone.

So soll auch der Hackerspace den Raum und die Möglichkeiten bieten, sich kreativ mit Technik und Naturwissenschaften auseinanderzusetzen, um dadurch den eigenen Horizont zu erweitern und natürlich auch Spaß daran zu haben. Für die Warpzone ist hierbei insbesondere der Bildungsauftrag in der Vereinssatzung verwurzelt, welchem durch regelmäßige Vorträge und Workshops zu technischen Themen nachgekommen werden soll. Hierbei ist es weniger wichtig, ob die Teilnehmer bereits Hacker nach obiger Definition sind, sondern eher, ob sie die dafür nötige Offenheit für Neues und den Willen etwas zu lernen mitbringen. Um dies zu ermöglichen, gibt es eine Werkstatt, in der die Mitglieder mit Lötkolben, Dremel und Oszilloskop ihren technischen Projekten nachgehen können, sowie eine Lounge, in der Programmierer eine anregende Atmosphäre für ihre Softwareentwicklung finden.

Die Entstehung der Warpzone

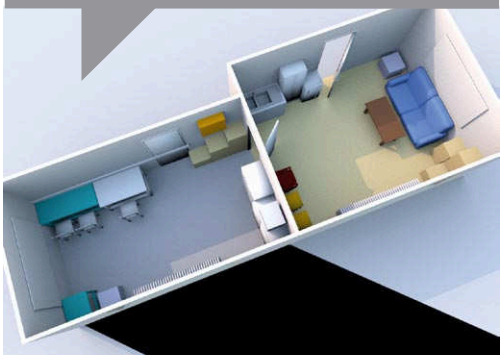
Die Geschichte ist schnell erzählt: am 15. August 2009 hatten zwei Nerds aus Münster die Idee, einen Hackerspace in Münster zu gründen. So wurde schnell ein Blogeintrag [3] verfasst und kurzerhand an alle bekannten technikaffinen Mailinglisten Münsters geschickt. So geschah es dann auch, dass sich am folgenden Dienstag zwölf Sympathisanten in einer von Münsters Lokalitäten trafen. Für eine Idee, die zu dem Zeitpunkt keine halbe Woche alt war, eine großartige Bilanz. So war klar: Das Potential zu „etwas Größerem“ existiert und muss genutzt werden. Aber was nutzen die Motivation und Kompetenz, wenn nicht einmal ein Ort zum Treffen zur Verfügung steht?

Doch bereits nach einem Monat wurde am Hawerkamp, dem Viertel der alternativen Szene Münsters, eine geeignete Räumlichkeit gefunden: zwei Räume mit insgesamt 35m². Alles in Bahnhofsnähe und einer Lage, die auch nächtliche Aktivitäten erlaubt. Schnell stand fest, dass es sowohl einen gemütlicheren (Vortrags-)Raum und eine Werkstatt geben soll. Jeder steuerte etwas zur Startfinanzierung bei und die Renovierung der Räume konnte beginnen. Diese war bitter nötig, denn zum einen ist der Hawerkamp nicht für den guten Zustand der Gebäude bekannt, zum anderen haben die Mitglieder der Warpzone naturgemäß überdurchschnittliche Ansprüche an die technische Infrastruktur. Doch nichts, was sechs Wochen harte Arbeit, viel Mate und geschicktes Budgethandling nicht schaffen können.

Die Möglichkeiten

Die Werkstatt bietet einige Tische, die über eine raumfüllende Brüstungsleiste mit normalem Stromnetz, Kleinspannungen und Netzwerk versorgt werden. Der Boden ist außerdem gefliest, so dass auch schwerere Arbeiten problemlos erledigt werden können. Alles hängt natürlich an einem extra gesicherten Stromkreis, so dass bei Arbeitsunfällen einer der beiden Not-Aus-Schalter betätigt werden kann, ohne dass die Programmierer in der Softwarelounge um ihre Daten fürchten müssen. So steht umfangreichen Lötarbeiten nichts mehr im Weg!

Abb. 3: Der Grundriss der Warpzone.



Damit jedes Mitglied auch die Möglichkeiten hat, seine Projekte zu verwirklichen, kann es sich bis zu zwei Kunststoffboxen kaufen, um seine Lötarbeiten verstauen zu können. Dazu haben wir uns gemeinsam im Baumarkt auf eine stapelbare Box geeinigt, so dass keine wüste Boxsammlung entsteht.

Die Lounge bietet einem alles, um seine Akkus wieder aufzuladen: Steckdosen für die Laptops und ein vielfältiges Angebot koffeinhaltiger Getränke - Kaffee, Club-Mate und diverse Colasorten stehen für ein kleines Entgelt zur Verfügung. Natürlich gibt es auch koffeinfreie Getränke, deren Absatz unerklärlicherweise jedoch wesentlich geringer ist. Außerdem gibt es genügend Sitzmöglichkeiten, so dass man sich gemütlich unterhalten kann oder auch in motivierender Gesellschaft seinen persönlichen Projekten nachgehen kann.

Beide Räume verfügen über ein Whiteboard, so dass man auch schnell gemeinsam Lösungen entwickeln oder anderen Mitgliedern Notizen hinterlassen kann. Ein Beamer rundet die Vortragsmöglichkeiten ab. Sollte das Publikum einmal die Räumlichkeiten der Warpzone überlasten, besteht auch die Möglichkeit eine größere Halle des Hauses zu nutzen.

Abb. 4: Ein Arbeitsplatz in Benutzung.

Aktivitäten

Als Anfang November die Räume vorhanden waren, konnte direkt durchgestartet werden. Als Idee entstand das sogenannde „Hack'N'Breakfast", also Hacken bis zum Frühstück. Dieses wurde auch gleich auf das erste Monatswochenende terminiert und schon die erste Veranstaltung war ein durchschlagender Erfolg. Durch die „Laufkundschaft", die sich an den Tagen der offenen Ateliers im Szeneviertel durch die Gebäude des alten Industriekomplexes gezwängt hat, waren unsere Vorträge auch gut besucht. Kein Wunder, denn von „Open Access", „Medienkunst" über „Die Demoszene" zu „Creative Commons" waren auch für jeden Themen dabei - sogar ohne technischen Hintergrund.

Abb. 5: Ein Vortrag in der Werkstatt

Das nächste Hack'n'Breakfast fand am ersten Dezemberwochenende statt - diesmal auch mit technisch orientierteren Vorträgen, wie einem Vortrag zu dem Debuggingwerkzeug „DTrace". Zudem wird parallel zum 26. Chaos Communication Congress im Dezember an der Aktion Dragons everywhere [4] teilgenommen, um auch Münsteranern, die sich die Fahrt und Übernachtung in Berlin zeitlich oder kostentechnisch nicht leisten können.

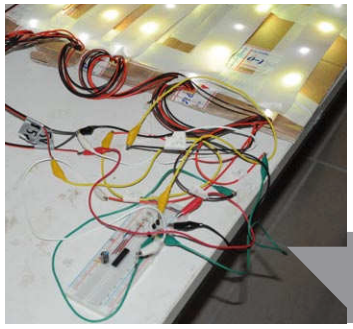


Abb. 6: Der LED-Prototyp auf einem Breadboard.

Aussichten & Kontakt

Naturgemäß ist so ein Hackerspace natürlich nie fertig gestellt, er lebt und entwickelt sich mit den Aktivitäten seiner Mitglieder. Jeder bringt seine Ideen und Projekte ein, so befindet sich zum Beispiel gerade die Deckenbeleuchtung der Lounge im Aufbau, die aus mehreren per Bussystem angesteuerten LED-Panelen besteht. Auch kleine Basteleien auf Basis des Etherrape / Ethersex [5] sollen schon gesichtet worden sein…

Natürlich ist jeder eingeladen, uns mal auch abseits unserer Veranstaltungen zu besuchen! Wir freuen uns auf euch!
Ihr findet Kontaktmöglichkeiten auf unserer Homepage [6]. Tragt euch doch auf unserer Mailingliste ein, indem ihr eine Mail an warpzone-request@warpzone.ms mit dem Betreff „Subscribe" schickt.

Sonderkasten: Hackerspaces mit Werkstatt

Neben der relativ jungen Warpzone gibt es natürlich schon länger andererorts Hackerspaces mit Werkstatt, zu nennen ist z.B. „Das Labor" in Bochum [7] oder das „Metalab" in Wien [8]. Wer auf der Suche nach einem Hackerspace in seiner Nähe ist oder Gleichgesinnte zur gemeinsamen Gründung sucht, wird eventuell auf Hackerspaces.org [9] fündig, das Wiki fasst eine Liste sämtlicher Hackerspaces. Eine allgemeine Einführung ist auch in dem Podcast „Chaosradio Express" [10] zu hören.

Links

[1] http://cmos.name
[2], [6] http://warpzone.ms
[3] http://raichoo.blogspot.com/2009/08/ein-hackerspace-fur-munster-und.html
[4] http://events.ccc.de/congress/2009/wiki/index.php/Dragons_everywhere#Nordrhein-Westfahlen
[5] http://www.ethersex.de
[7] http://www.das-labor.org/
[8] http://metalab.at/
[9] http://hackerspaces.org/wiki/
[10] http://chaosradio.ccc.de/cre134.html

MP32F103-Stick

Ein Mini-Mikrocontroller-Board mit USB und bis zu 4MB Datenspeicher

Mario Pieschel

Einleitung

Inspiriert vom verstärken Interesse an Cortex-M3-Mikrocontrollern und nach Einarbeitung in diese Materie veröffentliche ich hier ein kleines Board mit dem Mikrocontroller STM32F103CBT6 von STMicroelectronics (www.st.com). Dieser Chip hat 128KB Flash, 20KB RAM, SPI, I2C, USART, USB, CAN, ADC und läuft bis 72 MHz. Das Bord wird genauso aufgebaut sein wie mein MP2103-Stick [1].

Also sehr minimalistisch - alles was nötig ist um den Mikrocontroller zu betreiben, die Programmierung ohne Programmiergerät gewährleisten und möglichst viele Pins nach außen führen. Den MP32F103 an eine USB-Schnittstelle eines PCs anschließen und loslegen. Viele Mikrocontroller-Anwendungen sind als Stand-Alone-Anwendungen konzipiert – der MP32F103-Stick nicht!!!

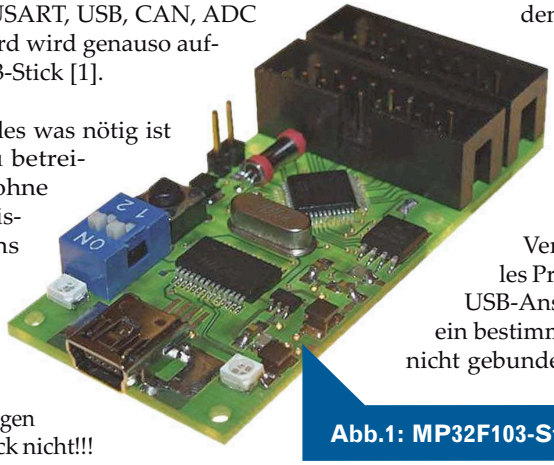


Abb.1: MP32F103-Stick

Er soll mehr als Interface-Baustein für den PC gesehen werden. Anwendungsbereiche sehe ich überall dort, wo irgendetwas mit dem PC elektrisch verbunden werden soll, was sich sonst nicht mit diesem über die herkömmlichen PC-Schnittstellen verbinden lässt. Überall wo der PC steuern, regeln und/oder messen soll, könnte der MP32F103-Stick zum Einsatz kommen. Die Daten feuern vom und zum PC mit einer Datenübertragungsgeschwindigkeit von fast bis zu 1 Megabit pro Sekunde! Für eventuelle Kalibrierdaten oder andere individuelle Daten für das zu betreibende Gerät steht ein bis zu 4 Megabyte großer Datenspeicher zur Verfügung. Für den MP32F103-Stick wird kein spezielles Programmiergerät benötigt – er wird einfach an einen USB-Anschluss eines PCs angeschlossen und das war's. An ein bestimmtes Betriebssystem des Ziel-PCs ist man ebenfalls nicht gebunden, da für den eingesetzten USB-Baustein (FT232) Treiber für alle gängigen Betriebssysteme vom Hersteller FTDI-Chip angeboten werden.

Funktionsmodell

LINK: Datasheet vom STM32F103CBT6 [2]
LINK: st.com-Page vom STM32F103CBT6 [3]
LINK: Datasheet und Anwendung vom AT45DBxxx [4]
LINK: Datasheet vom TPS62203 [5]

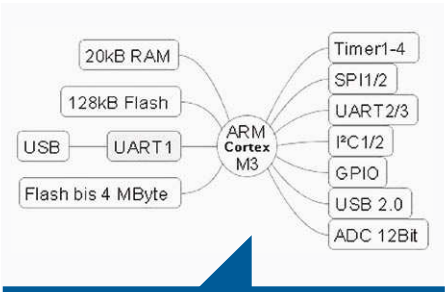


Abbildung 2: Funktionsmodell

Schaltplan

Herzstück des Boards ist der Mikrocontroller STM32F103CBT6 (IC1) mit 128kB Flash und 20 kB RAM. Die Anschlüsse von Port A und Port B sind mit den Steckleisten CON1 und CON2 zur Außenwelt verbunden, mit Ausnahme von PA9 und PA10, welche zur Kommunikation mit einem Hostrechner an das USB-IC FT232RL (IC3) gehen. Der Mikrocontroller benötigt drei Versorgungsspannungen mit je 3.3V (VDD,

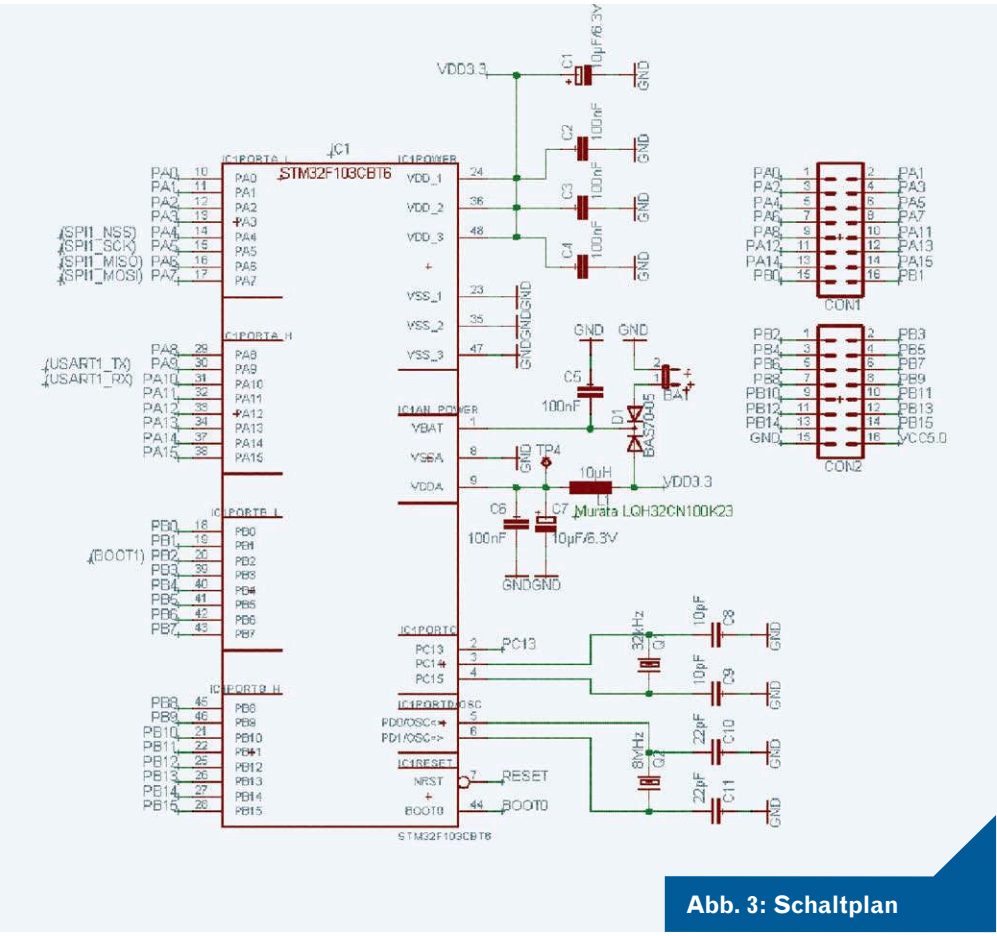


Abb. 3: Schaltplan

VDDA und VBAT). Der digitale Teil wird über VDD versorgt, der analoge über VDDA und die interne Uhr über VBAT. C1 bis C7 sind Abblockkondensatoren, L1 fungiert als Eingangsfilter gegen Störungen der analogen Komponenten und D1 dient der Entkopplung einer an den Stecker „BAT“ angeschlossene Backupbatterie für die interne Uhr (1 an +3,3V, 2 an 0V). Die beiden Quarze Q1 (32kHz) und Q2 (8MHz) werden für die externe Taktversorgung des Mikrocontrollers benötigt.

Der Mikrocontroller kann in drei Boot-Modi gestartet werden, welche mit S1 eingestellt werden können.

1=ON (2: ON oder OFF) ➜ der Mikrocontroller startet aus dem Flash (Normalfall).

1=OFF und 2=ON ➜ der interne Bootloader wird gestartet (zum downloaden der Firmware).

1=OFF und 2=OFF ➜ der Mikrocontroller startet aus dem RAM (für Tests).

R1 und R2 sind Pullup-Widerstände, R3 und R4 begrenzen den Stom im Fehlerfall und links neben S1 ist eine optionale Brücke, für den Fall, dass booten vom Flash unabhängig von S1 erwünscht wird (z.B. Auslieferungszustand).

S1 ist der RESET-Taster und D2, R5, C12 lösen einen RESET-Impuls nach Einschalten der Betriebsspannung aus. LED1 ist für universelle Zwecke an Port-Pin PC13 mit R6 als Strombegrenzung angeschlossen.

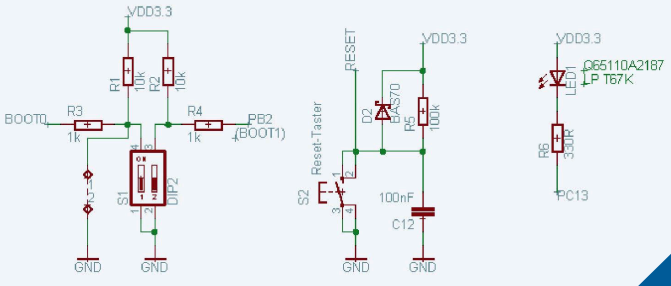


Abb. 4: MP32F103_Logik

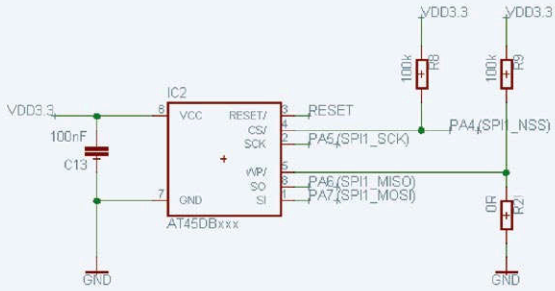


Abb. 5: Flash Datenspeicher

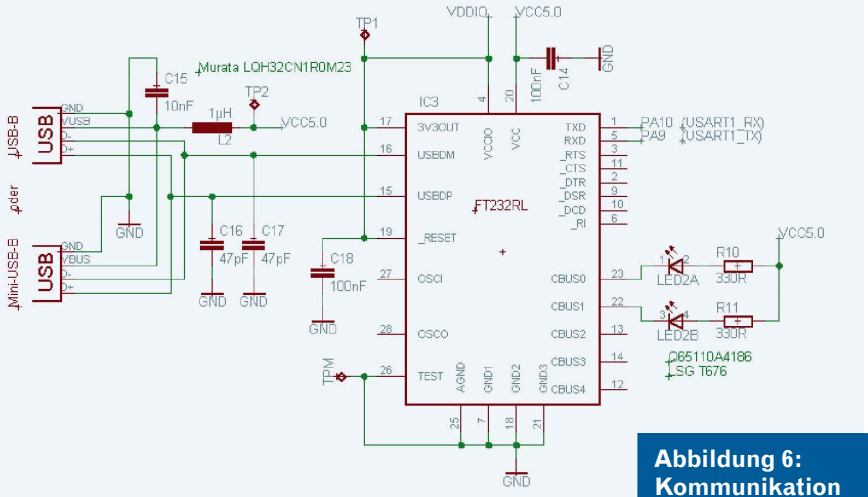


Abbildung 6: Kommunikation

Das Board verfügt über einen Datenspeicher von bis zu 4MB (Mega Byte) Flash, IC2 AT45xxx von Atmel. Dieser ist per SPI mit dem Mikrocontroller verbunden. C13 ist ein Abblockkondensator, R8 und R9 Pullup-Widerstände, und R2! Nur bestücken, wenn das Flash schreibgeschützt sein soll (!!! im Normalfall nicht bestücken !!!).

Die Kommunikation mit einem Hostrechner erfolgt per seriellem Port UART1 (PA9 und PA10) über den USB-zu-seriell-Umsetzer FT232RL (IC3). Dort gehen die beiden Signale UART1_RX und UART1_TX an Pin 1 und 5. Die beiden Datenleitungen USBDM (Pin 16) und USBDP (Pin 15) sind direkt mit der USB-B-Buchse (Pin 2 und 3) verbunden. Es kann wahlweise eine Mini-USB-B-Buchse oder eine normale USB-B-Buchse eingesetzt werden. VDD5.0 (5 Volt) kommt von der USB-B-Buchse über L2 (Entstörung) und geht an Pin 20 von IC3. C14 bis C18 sind Abblockkondensatoren für die Versorgungsspannungen. LED1 und LED2 signalisieren den seriellen Datentransfer.

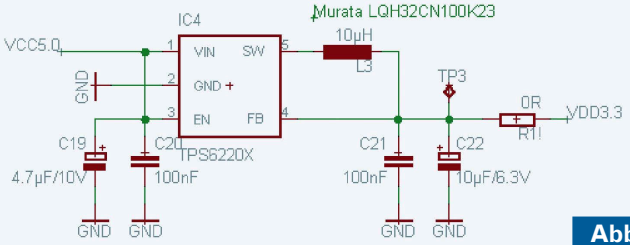
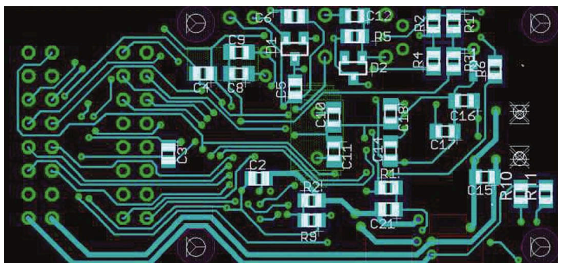


Abb. 7: Versorgungs-spannung

Der Mikrocontroller STM32F103CBT6 (IC1) benötigt eine Versorgungsspannung von 3,3V, welche von IC4 (TPS62203) erzeugt wird. L3 wird für den internen Schaltregler benötig und C19 bis C22 sind Abblockkondensatoren. R1! (Lötbrücke) erst bestücken wenn die elektrische Prüfung erfolgt ist.

Layout



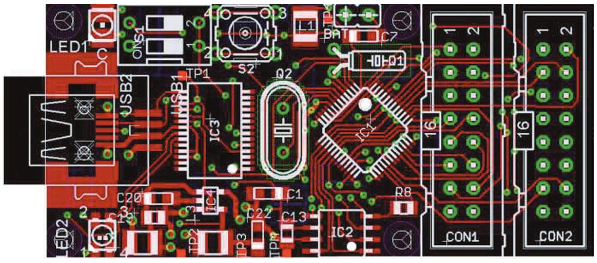


Abb. 8: Stückliste

Eagle-Files

MP32F103Stick V1.3.zip [6]

Die Leiterplatte sollte von einem Leiterplattenhersteller gefertigt werden. Ich habe die Erfahrung gemacht, dass Anbieter mit Preis-Berechnungen auf Quadratdezimeter-Basis egal mit welchem Inhalt die günstigsten sind. Wichtig ist, dass die Leiterplatten mit Durchkontaktierungen gefertigt werden. Aus Erfahrung rate ich die Leiterplatten mit Lötstopplack fertigen zu lassen. Einfach die Datei „MP32F103_Stick_V1.3.brd“ dem Hersteller schicken.

Qty	Value	Package	Parts	Shop	Bestellnummer	gewähltes Bauteil
1	2pol 2,54mm	JP1_MP	BAT	reichelt.de	SL 1X36G 2,54	auf 2pol kürzen
3	10µF/6.3V	A/3216-18R	C1, C7, C22	secor.de	TA10u-16A SMD	10µF / 16V
2	22pF	C0805K	C10, C11	reichelt.de	NOP-G0805 22P	
1	10nF	C0805K	C15	reichelt.de	X7R-G0805 10N	
2	47pF	C0805K	C16, C17	reichelt.de	NPO-G0805 47P	
1	4.7µF/10V	A/3216-18R	C19	secor.de	TA10u-16SMD	10µF/16V
11	100nF	C0805	C2, C3, C4, C5, C6, C12, C13, C14, C18, C20, C21	reichelt.de	X7R-G0805 100N	
2	10pF	C0805K	C8,C9	reichelt.de	NPO-G0805 10P	
2		ML16	CON1, CON2	reichelt.de	WSL 16G	
1	BAS70	SOT23	D2	reichelt.de	BAT 54 SMD	
1	BAS70-05‘	SOT23	D1	reichelt.de	BAT 54C SMD	
1	STM32F103 CBT6	TQFP48	IC1	sander-electronic.de	STM32F103CBT6	
1	AT45DBxxx	SOIC8	IC2	watterrott.de	2009572	AT45DB321D-SU
1	FT232RL	SSOP28DB	IC3	reichelt.de	FT232 RL	
1	TPS62203	SOT23-5L	IC4	hbe-shop.de	8458049	
2	10µH	R1210	L1, L3	reichelt.de	LQH3C 10µ	
1	1µH	R1210	L2	reichelt.de	LQH3C 1,0µ	
1		P-LCC-2-MPI	LED1	reichelt.de	LP T67K	
1		P-LCC-4-2-MPI	LED2	reichelt.de	LSG T676	
1	32kHz	TC26H	Q1	reichelt.de	0,032768	
1	8MHz	HC49U-V	Q2	reichelt.de	8,0000-HC49U-S	
2	10k	R0805	R1, R2	reichelt.de	SMD-0805 10,0K	
2	1k	R0805	R3, R4	reichelt.de	SMD-0805 1,00K	
3	100k	R0805	R5, R8, R9	reichelt.de	SMD-0805 100K	
3	330R	R0805	R6, R10, R11	reichelt.de	SMD-0805 330	
1	DIP2	DS-02	S1	reichelt.de	NT02	
1	Reset-Taster	B3F-10XX	S2	reichelt.de	TASTER3302	
1	SUB-B-S	USB-B-SMT	USB1	reichelt.de	USB BW SMD	Alternativ-bestückung
1	USB-MB-S2	USB-MP-S2	USB2	reichelt.de	USB BWM SMD	

Bestückung der Leiterplatte

Elektronische Bauteile sind durch elektrostatische Ladungen gefährdet. Deshalb ab jetzt mit ESD-Armband und ESD-Unterlage arbeiten!!! Beim Arbeiten mit den kleinen Bauteilen hilft auf jeden Fall ein Briefmarken-Lupen-Headset oder ein Stereo-Mikroskop mit Verstärkungsfaktor 10 (in der Bucht ca. 200,- EUR). Zuerst die beiden großen ICs, IC1 (STM32F103CBT6) und IC3 (FT232RL), bestücken. Da es sich um ziemlich kleine und mit hoher Pin-Anzahl versehene elektronische Bauteile handelt, empfehle ich folgende Vorgehensweise: Die Lötpads auf der Leiterplatte für das entsprechende IC reichlich mit dem Lötkolben und Lötzinn (<= 0,5 mm Durchmesser) verzinnen. Die Lötpads sollten glänzen und um ca. 0,1 bis 0,2 mm erhaben sein. Nun die Leiterplatte mit handelsüblichem Brennspiritus reinigen. Jetzt das IC mit ein wenig Klebstoff auf der Leiterplatte fixieren. Ich nehme dafür handelsüblichen Klebestift (Pr…-Stift). Eine kleine Messerspitze voll vom Klebestift auf die Mitte der IC-Position auf die Leiterplatte auftragen, das IC auf die Leiterplatte aufsetzen und positionieren. Es darf auf keinen Fall Klebstoff bis zu den Lötanschlüssen des ICs quellen. Erst mit dem Festlöten beginnen, wenn das IC korrekt auf den Lötpads sitzt und etwas angetrocknet ist. Ab jetzt wird nicht mehr mit Lötzinn gearbeitet – dieses am besten so weit wie möglich weglegen. Die Kapillarwirkung der IC-Anschlüsse ist viel zu groß, diese würden eine weitere Zugabe von Lötzinn sofort dieses zwischen die Lötpins ziehen und damit für reichlich Kurzschlüsse sorgen. Ab jetzt wird nur noch mit Flussmittel gearbeitet, entweder handelsübliches Flux oder Kolofonium (in Brennspiritus aufgelöst) nehmen. Ist das IC korrekt positioniert und fixiert mit dem Festlöten beginnen. Jeden einzelnen IC-Anschluss mit dem heißen Lötkolben und wenig Druck in Richtung Leiterplatte anlöten. Wichtig ist wirklich, sich damit Zeit zu nehem und darauf zu achten, dass das Lötzinn richtig fließt. Sollte bei einem der hier beschriebenen Arbeitsschritte etwas schief gehen, unbedingt die Arbeiten abbrechen, alles reinigen und von Anfang beginnen. Sind IC3 und IC4 erfolgreich auf die Leiterplatte gelötet können alle anderen Bauteile bestückt und gelötet werden. Wichtig: „R1!“ und „R2!“ nicht bestücken!!! Die voll bestückte Leiterplatte gründlich mit Brennspiritus

und einem kleinen Pinsel reinigen. Die Leiterplatte wegen der Buchsen und Schalter/Taster nicht im Reinigungsmittel baden. Das Reinigungsmittel nicht abtrocknen lassen sondern mit Druckluft entfernen. Es werden so unschöne Flecken vermieden. Sollte der Stick in rauer Umgebung betrieben werden ist das Lackieren mit farblosem Leiterplattenlack sinnvoll.

Optische Prüfung

Jetzt sollte eine ausgiebige Fehlersuche auf Kurzschlüsse, kalte Lötstellen, Verpolung von Bauteilen und richtiger Bestückung erfolgen. Entweder mit einer großen Lupe oder einem Stereo-Mikroskop mit Vergrößerungsfaktor 10.
Alle weiteren Arbeitsschritt erst nach vollständiger Trocknung des Stick durchführen.

Elektrische Inbetriebnahme

Für erste Tests ist es am Besten den Stick mit einem Strom geregelten Netzteil über die USB-Buchse (Pin1 +5V, Pin4 Masse) zu verbinden; den Strom auf 50 mA begrenzen und 5 Volt einstellen. Nach Einschalten sollte ein Strom von ca. 10 bis 20 mA fließen und die anliegende Spannung weiterhin 5 Volt betragen. Bricht die Spannung zusammen auf keinen Fall die Strombegrenzung erhöhen sondern auf Kurzschlusssuche gehen. Fließt zu wenig Strom oder keiner kalte Lötstellen suchen. Ist kein Netzteil vorhanden, einfach drei kleine billige Knopfzellen á 1,5 Volt in Reihe schalten und den Strom mit einem Multimeter messen. Stimmt der Strom und liegen die 5 Volt an, die Spannungen an den Testpunkten TP1 bis TP4 überprüfen. Bezugspunkt ist TPM (Masse). Folgende Spannungen sollten gemessen werden:

TP1 = 3,3 Volt
TP2 = 5 Volt
(oder 4,5V bei der Variante mit den drei 1,5V Knopfzellen)
TP3 = 3,3 Volt
TP4 = 3,3 Volt

Wenn diese Spannungen OK sind kann der Stick an den PC/Notebook über ein USB-Kabel angeschlossen werden. Nun sollte das Betriebssystem (Windows, für WinCE, Linux und Mac weiß ich es nicht) melden, dass neue Hardware erkannt wurde. Jetzt kann der Treiber für den FT232RL installiert werden. Den Treiber (Virtual COM Port, VCP) von der Herstellerseite www.ftdichip.com downloaden – die Vorgehensweise ist dort ausführlich beschrieben. Nach erfolgreicher Installation des Treibers sollte im Hardwaremanager ein neuer COMx-Port zu sehen sein. Die COM-Nummer sollte man sich merken. Windows registriert für jede USB-Schnittstelle eine andere COM-Port-Nummer, was zu Verwirrung beim Umstecken führen kann. Damit sind aber auch mehrere Sticks anschließbar.

LINK: die Treiber für den FT232RL [7]
LINK: Driver Installation Guides [8]

Sind alle Tests und Arbeitsschritte bis hier OK kann „R1!“ (Lötbrücke) eingelötet werden um den Mikrocontroller mit Strom zu versorgen. Es sollte ein Strom zwischen 20 und 40 mA fließen, wenn nicht: Fehlersuche nach Kurzschlüssen, kalten Lötstellen, verpolten Bauteilen und falscher Bestückung.

Firmware Upload

Zum Hochladen der Firmware benötigt man das Programm „mcuisp_stm“. Das Download-Programm von st.com geht natürlich auch. LINK: mcuisp_stm [9]

Jetzt am Besten erst einmal testen, ob sich der Mikrocontroller meldet. Schalter S1 ➜ 1=OFF und 2=ON und RESET-Taster drücken das Programm „mcuisp_stm“ starten und den Button „Read Chip Info(R)“ klicken.

Für Download der Firmware den Button „Start ISP(P)“ klicken. [Abbildung 9] Programm starten mit Schalter S1 ➜ 1=ON und 2=egal und RESET-Taster drücken. [Abbildung 10]

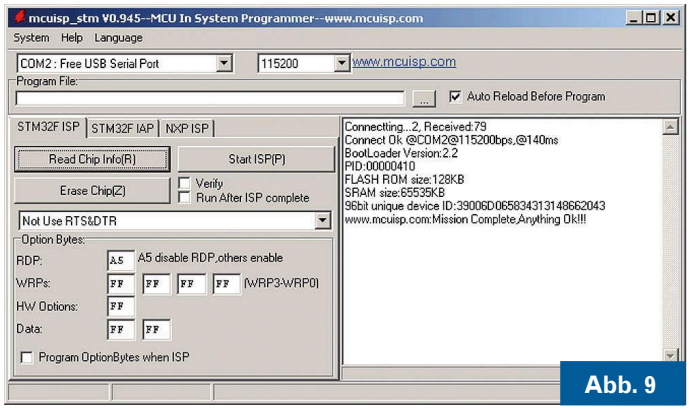


Abb. 9

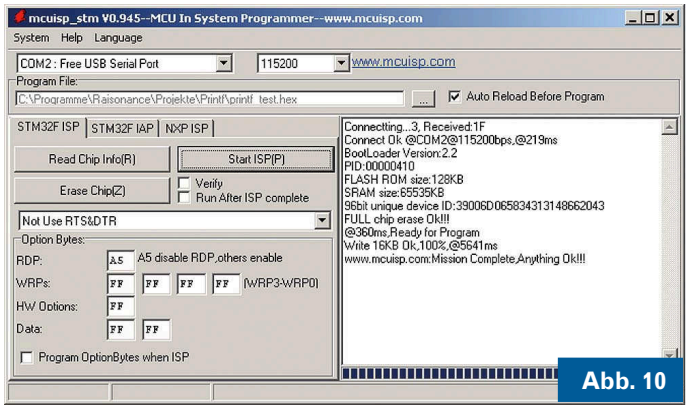


Abb. 10

Compiler und IDE

Die Integrierte Entwicklungsumgebung von Raisonance, Ride7 und RKit-ARM, umfasst einen voll funktionsfähigen und Code unbegrenzten GNU C/C++-Compiler und das Raisonance Integrated Development Environment (RIDE). Über eine grafische Benutzeroberfläche wird die Bedienung der Softwareentwicklungs-Tools (Compiler, Assembler, Linker und Simulator) ermöglicht. Für die Softwareentwicklung und Kompilierung vom C-Source-Code bis hin zum endgültigen HEX-File besteht keine Codegrößenbegrenzung, lediglich das Debugging ist bis 32kByte begrenzt. Das ist aber hier nicht von Belang, da mit dem Bootloader des Mikrocontrollers geflasht wird.

Auf der Page von Raisonance.com anmelden und die Files „RKit-ARM“ (GCC) und „Ride7“ (IDE) downloaden und installieren. Zuerst RKit-ARM dann Ride7. Der Umgang mit diesen Programmen ist super einfach - keine Rumquälerei mit Makefiles und solcherlei Teufelszeug - alles ist bereits Mundgerecht von Raisonance zugeschneidert. Sehr empfehlenswert!

LINK: Compiler und IDE Ride7 von Raisonance.com [11]
LINK: Getting Started with ARM & Ride7 User manual [12]

Und wie einfach alles ist zeigt dieses Beipiel-Projekt:
LINK: Hello-World-Projekt mit Ride7 [13]

Da in Ride7 kein serielles Upload-Tool für den MP32F103-Stick integriert ist, kann zum Hochladen der Firmware (Hello_World.hex) das Programm „mcuisp_stm“ genommen werden.

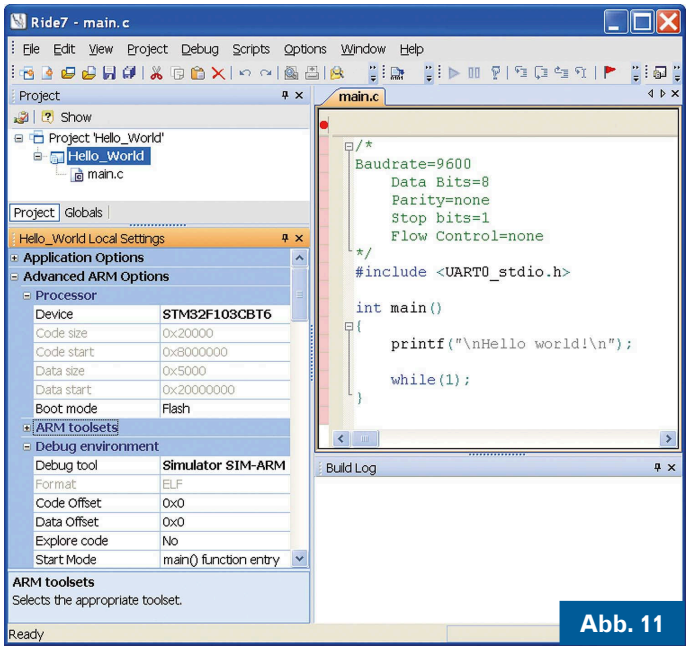


Abb. 11

Links

[1] http://www.mikrocontroller.net/articles/MP2103-Stick:_Ein_Mini-Mikrocontroller-Board_mit_USB_und_bis_zu_4MB_Datenspeicher
[2] http://www.st.com/stonline/products/literature/ds/13587/stm32f103cb.pdf
[3] http://www.st.com/mcu/devicedocs-STM32F103CB-110.html
[4] http://atmel.com/dyn/products/devices.asp?family_id=616#1802
[5] http://focus.ti.com/lit/ds/symlink/tps62203.pdf
[6] http://www.mikrocontroller.net/wikifiles/b/b6/MP32F103Stick_V1.3.zip
[7] http://ftdichip.com/Drivers/VCP.htm
[8] http://ftdichip.com/Documents/InstallGuides.htm
[9] http://www.mikrocontroller.net/wikifiles/5/55/MP32F103Stick_mcuisp_stm.zip
[10] http://www.mikrocontroller.net/articles/MP32F103-Stick:_Ein_Mini-Mikrocontroller-Board_mit_USB_und_bis_zu_4MB_Datenspeicher
[11] http://www.mcu-raisonance.com/mcu_downloads.html
[12] http://www.mcu-raisonance.com/tzr/scripts/downloader2.php?filename=T020/file/d8/cd/datj7g5d1wr&mime=application/pdf&originalname=GettingStartedARM_Ride7.pdf
[12] http://www.mikrocontroller.net/wikifiles/4/44/MP32F103Stick_Hello_World.zip

UNTERSTÜTZER GESUCHT

Das Gemeinschaftsprojekt

http://www.embedded-projects.net/journal

embedded-projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

embedded-projects.net
JOURNAL
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Werdet aktiv!

**Das Motto: Von der Community für die Community!
Das Magazin ist ein Open-Source Projekt.**

**Falls Du Lust hast, Dich an der Zeitschrift durch einen Beitrag zu beteiligen, würden wir uns darüber sehr freuen. Schreibe Deine Idee an:
sauter@embedded-projects.net
Wir werden dann gemeinsam sehen, was wir daraus machen können.**

Regelmäßig lesen!

Die Zeitschrift wird über mehrere Kanäle verteilt. Der erste Kanal ist der Download als PDF-Datei. Alle Ausgaben sind auf der Internetseite [1] verfügbar. Diejenigen, die lieber eine Papierversion erhalten möchten, können den zweiten Kanal wählen. Man kann sich dort auf einer Internetseite [1] in eine Liste für die gesponserten Abos eintragen. Beim Erscheinen einer neuen Ausgabe wird dank Sponsorengeldern an jeden auf der Liste eine Ausgabe des aktuellen Journal versendet. Falls man den Versandtermin verpasst hat, kann man das Heft auch zum Preis von einem Euro über einen Online-Shop [2] beziehen.

**1. Internetseite (Anmeldeformular gesponserte Abos)
http://www.embedded-projects.net/journal
2. Online-Shop für Journal (Preis 1 EUR + Versand)
http://www.eproo.de/journal**

Sponsoren gesucht!

Damit wir weiterhin diese Zeitschrift für jeden frei bereitstellen können, suchen wir dringend Sponsoren für Werbe- und Stellenanzeigen. Bei Interesse meldet Euch bitte unter folgender Telefonnummer: 0821/27 95 990 oder sendet eine E-Mail an die oben genannte Adresse.

Bitte frei machen

Embedded Projects
Holzbachstraße 4
D-86152 Augsburg

Bitte in Druckbuchstaben gut lesbar ausfüllen, damit wir Ihre Bestellung bearbeiten können.

Name / Firma

Straße / Hausnummer

PLZ / Ort

E-Mail / Telefon / Fax

☐ **Bestellung des Embedded Projects Journal:**
Ich möchte jede zukünftige Ausgabe erhalten

☐ **Bestellung des Embedded Projects Journal für Hochschulen/ Ausbildungsbetriebe**
Wir möchten jede zukünftige Ausgabe zu Ausbildungszwecken bestellen
Anzahl der Hefte pro Ausgabe [5] [10]
(zutreffendes bitte ankreuzen)

☐ **Anforderung Infomaterial zur Anzeigenschaltung:**
Wir sind eine Firma und sind an Werbe- und Stellenanzeigen interessiert.
Bitte schicken Sie uns kostenlos und unverbindlich Infomaterial, Preisliste, ect. zur Anzeigenschaltung zu.

[IMPRESSUM]

embedded projects GmbH
Holzbachstrasse 4
D-86152 Augsburg
Telefon: +49 (0) 821 / 27 95 99 - 0
Telefax: +49 (0) 821 / 27 95 99 - 20
Mail: journal@embedded-projects.net

Anzeigemöglichkeiten und Preisliste auf Anfrage via Mail.

Herausgeber: Benedikt Sauter
Gestaltung/Satz:Das-Medienkollektiv.de

Veröffentlichung: 4x / Jahr

Ausgabeformate: PDF / Print
Auflage Print: 2500 Stk.
Einzelverkaufspreis: 1,00 EUR

Dies ist ein Open-Source Projekt.
Informationen zum ABO - www.embedded-projects.net/journal

Titelfoto: Sebastian Kaulitzki, Viorel Sima, dextroza,
Fotocomposing: Das Medienkollektiv
EPJ Anzeige: Ramona Heim, Nicemonkey, dip,...@fotolia.com

SPEZIALISTEN GESUCHT

Dein Spaß an High-tech
Deine Herausforderungen an Bits und Bytes
Deine Sendung mit der Maus - für Erwachsene

Du willst kreativ Entwicklungsprozesse mitgestalten -
werde Teil der Elektronikwelt.

Software-Ingenieure (m/w) Echtzeit und Embedded Systeme

Zur Verstärkung unseres Entwicklerteams in Königsbrunn
bei Augsburg suchen wir ab sofort deutschsprachige
Informatiker, Ingenieure oder Naturwissenschaftler mit
abgeschlossenem Hochschulstudium und Erfahrung
in der Entwicklung technischer Software.

Infos: http://www.ibv-augsburg.net/jobs

IBV - ECHTZEIT- UND EMBEDDED GMBH & CO. KG

Keltenstraße 2 D-86343 Königsbrunn
Fon +49 (0) 82 31.95 86 - 041 Fax +49 (0) 82 31.95 86 - 049
info@ibv-augsburg.net www.ibv-augsburg.net

ibv
Realtime is BLUE