



embedded-projects.net

JOURNAL

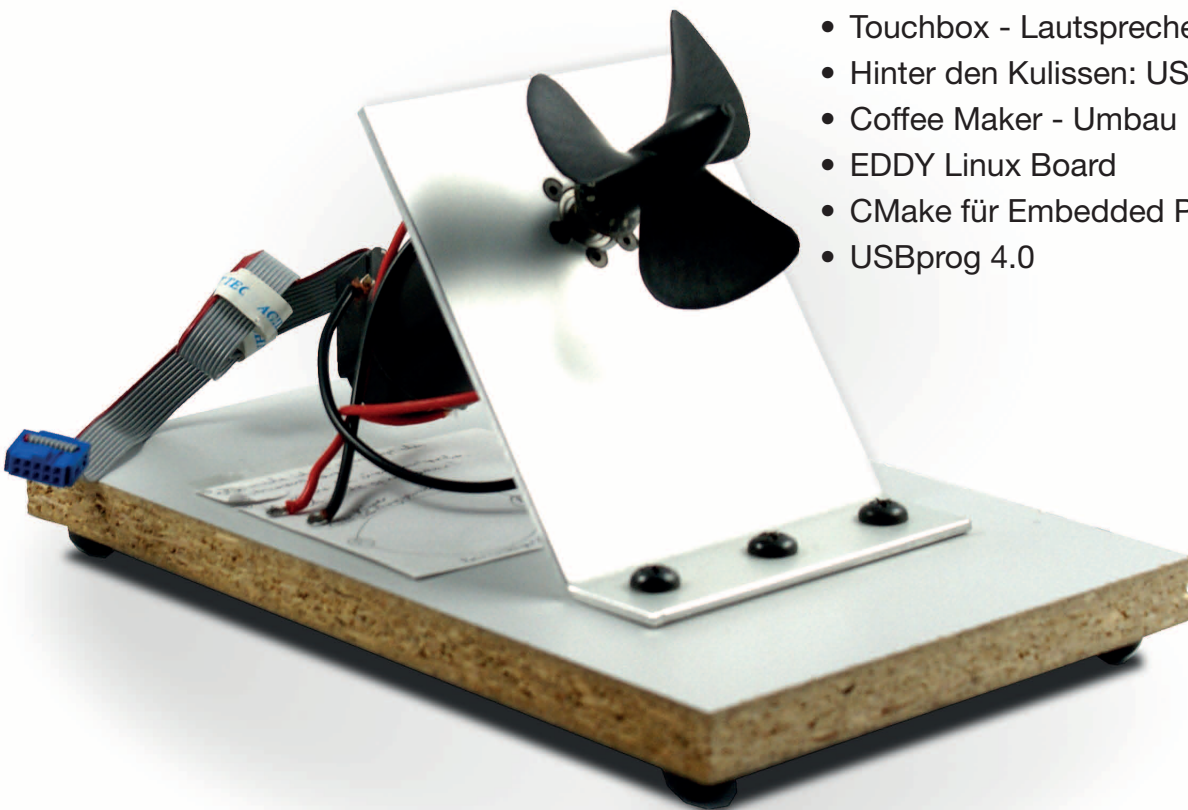
OPEN SOURCE SOFT-AND HARDWARE PROJECTS

Eine Open-Source Zeitschrift
zum Mitmachen!

TOUCH IT!

[PROJECTS]

- Vom Stromlaufplan zur Leiterplatte: Teil 2
- Touchbox - Lautsprecher zum Anfassen
- Hinter den Kulissen: USB Netzwerkkarte
- Coffee Maker - Umbau
- EDDY Linux Board
- CMake für Embedded Projekte
- USBprog 4.0



**Wussten Sie,
dass wir eine Firma
für kundenspezifische
Entwicklungen mit
Sitz in Augsburg sind?**



Wir bieten:

Hardware, Software,
Embedded, Software-
Entwicklung,
Mikrocontroller,
Anwendungsentwicklung,
Fachbeiträge/
Literatur, Schaltplan,
Webentwicklung,
Open-Source,
E-Commerce, Platinen-
layout, GNU/Linux

Kommen Sie vorbei!

Mehr über Augsburg

Römer, Fugger ...

Embedded Projects Journal - Ausgabe No. 10

In der letzten Einleitung habe ich angekündigt mehr über das schöne Städtchen Augsburg zu schreiben.

Augsburg wurde von Kaiser Augustus 15 v. Chr. als Castra gegründet. Eine Castra ist ein römisches Militärlager. Nach und nach wurde es immer bedeutender, bis es 95 n. Chr. zur Hauptstadt der römischen Provinz Raetien wurde.



Die nächsten größeren Ereignisse waren erst wieder im Mittelalter. Im Jahre 955 n.Chr. besiegte Kaiser Otto I mit Hilfe des Bischofs Ulrich von Augsburg die Ungarn auf dem Lechfeld. Im Jahre 1158 - jetzt erst - wurde München gegründet.

Die nächste prominente Epoche war die der Fugger. Die Fugger waren Kaufmänner und handelten seit dem Ende des 14. Jahrhunderts eigentlich mit allem, was damals benötigt wurde.

1555 gab es in Augsburg einen Tag über den sich heute noch jeder Augsburger ganz besonders freut. Das war der Religionsfrieden am 8. August. An diesem Tag schlossen die Katholiken gemeinsam mit den Protestanten einen Vertrag für das friedliche Zusammenleben.

Als Erinnerung an diesen Tag, hat die Stadt Augsburg hier einen eigenen Feiertag. Da Augsburg wie bekannt in Bayern liegt, ist es also die Stadt mit den meisten Feiertagen in Deutschland.

Natürlich darf man das Wahrzeichen von Augsburg nicht vergessen - das Rathaus - eine der bedeutendsten Renaissancebauten nördlich der Alpen.

So - ein Historiker werde ich wohl nicht, aber ich hoffe so dem einen oder anderen etwas unsere Geschichte näher gebracht zu haben.

Eine Sache, die zwar überhaupt nicht zu dieser Einleitung passt habe ich trotzdem noch:

Seit kurzem veröffentlichen wir unsere Neuigkeiten Rund um den Online-Shop und das embedded projects Journal bei Facebook [1] und Twitter [2]. Wenn jemand auf diesem Weg ebenfalls informiert werden will, kann man sich jetzt entsprechend verlinken.

[1] Facebook <http://www.facebook.com/embeddedpro>

[2] Twitter <http://twitter.com/#!/embeddedpro>



embedded projects GmbH
HARDWARE FOR PROJECTS

Holzbachstraße 4, D-86152 Augsburg
Tel +49 (0) 821 279599-0
Fax +49 (0) 821 279599-20
info@embedded-projects.net

wawision.embedded-projects.net

waWision - die Steuerzentrale für Ihre Firma



Verwal-
tung

Plug &
Play

Waren-
eingang

Marke-
ting

FiBu

Produktion

automa-
tisches
Lager

Online-
Shops

EIN SYSTEM AUS EINER HAND

- keine Installation
- Betriebssystem unabhängig
- Standardhardware Plug & Play
- mitwachsend

DEMOVERSION

weitere Infos
finden Sie auf
unserer
Internetseite



Vom Stromlaufplan zum Leiterplattenlayout

Fortsetzung: Teil 2

Bernhard Redemann <b-redemann@gmx.de>

Das Ethernetmodul

Das im ersten Teil (Ausgabe 7) besprochene Ethernetmodul wurde so gestaltet, dass sowohl bedrahtete als auch SMD-Bauteile eingesetzt werden. Alle SMD-Bauteile wurden auf der Unterseite platziert, während die bedrahteten Bauteile auf der Oberseite Platz finden. Aber es gibt einen Haken, wie man vielleicht schon erkannt hat: Da das Modul zur Verwendung in einem Steckbrett eingesetzt werden soll, müssen die beiden Stiftleisten (X1, X2) von der Unterseite eingesetzt und von oben angelötet werden.

Für den Herstellungsprozess hat das einen weiteren Arbeitsschritt zur Folge: Handbestückung (von unten) und Anlöten (von oben) per Hand (Abbildung 1 und Abbildung 2 verdeutlichen dies).

Man kann daraus schlussfolgern: Die Auswahl der Bauteiltechnologie (SMD,

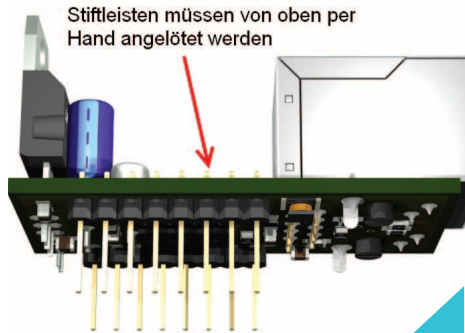


Abb. 1: Stiftleisten von unten montiert

bedrahtet oder beides) und das Platzieren der Bauteile im Layout (oben und unten) haben entscheidende Auswirkungen auf die Bestück- und Lötvorgänge der Fertigung und damit auch direkt auf die Kosten der Herstellung. Je weniger Arbeitsschritte bei der Herstellung eines Boards

notwendig werden, um so besser, da kostengünstiger.

Dies zeigt einmal mehr, welche Probleme beim Design einer Leiterplatte auftreten können und wie der Layouter diesen Anforderungen gerecht werden muss.

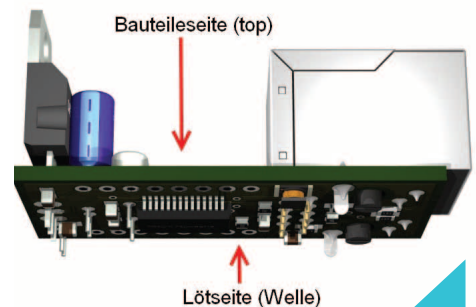


Abb. 2: Modul ohne Stiftleisten

Weitere Empfehlungen für das Platzieren von Bauteilen

Doch nun zurück zum Leiterplattenlayout. Generell gibt es natürlich keine Einschränkungen bei der Bauteileplatzierung, solange die Designrules, die mechanischen, thermischen, EMV- und generellen Vorgaben der Fertigung eingehalten werden. Oftmals ist es aber nicht ratsam, dass z.B. zentrale Bauteile (ICs) an der Leiterplattenkante liegen, vor allem dann, wenn viel Platz auf der Leiterplatte ist und es sich um eine einfache Schaltung handelt. Damit verbaut

man sich die Möglichkeit, von allen Seiten Leiterbahnen zu verlegen.

Auf der anderen Seite gibt es natürlich Bauteile, die durchaus am Rand der Leiterplatte liegen können und sollten, so z.B. Steckverbinder, Schraubklemmen, Schalter oder Bauteile mit Kühlkörpern. Damit schließt sich der Kreis zum Thema Servicefreundlichkeit (siehe Teil 1, „Zum Leiterplattenlayout“), denn durch z.B. außen-liegende Steckverbinder kann auf einfachste Weise die Schaltung an eine Peripherie angeschlossen werden. Dabei ist vom Layout her zu beachten, in welcher Richtung die Anschlüsse liegen. Gerade bei Schraubklemmen kann das dazu führen, dass man keine Leitung daran anschließen kann, wenn die Kabeleinführung auf der falschen Seite liegt.

auch die Verbindungen im Stromlaufplan geändert werden, was Nacharbeit bedeutet.

Bei Bauteilen, die sehr warm und/oder mit Kühlkörpern versehen werden, ist es ebenfalls meist notwendig und ratsam, diese eher am Rand als in der Mitte der Leiterplatte zu platzieren. Da alle Bauteile mehr oder weniger empfindlich auf Wärme reagieren und damit die Funktionalität einer Schaltung beeinflussen können, muss die Wärme dort abgeführt werden, wo die Bauteildichte gering ist. Am Rand wäre das also besser als in der Mitte. Besonders wärmeempfindliche Bauteile wie z.B. Elkos sollten mit einem gewissen Abstand zu einem Kühlkörper platziert werden.

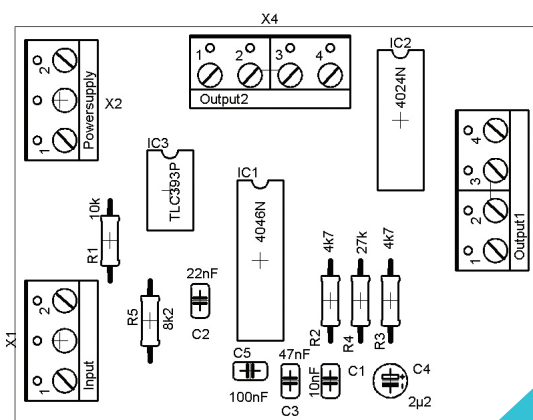


Abb. 3: Man achte auf die Lage der Schraubklemmen!

In der Abbildung 3 ist dem Layouter (war ich das etwa?) solch ein Fehler unterlaufen. Vergleicht man die Schraubklemmen mit dem realen Bauteil (Abbildung 4), so stellt man fest, dass die Kabeleinführung auf der Seite der glatten Fläche liegt. Somit müssen nicht nur X1, X2 und X4 gedreht, sondern

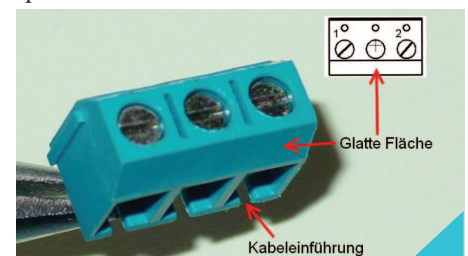


Abb. 4: Reale Klemme und Symbol

Ungenutzte Eingänge, Blockkondensatoren und Gate- Pinswap

Die folgende Schaltung mit den NAND-Gattern und dem Operationsverstärker sieht an sich harmlos aus. Dennoch gibt es hier Punkte, die zu beachten sind. Das betrifft zum Einen die unbenutzten Eingänge des HC132 und des LM393. In aller Regel sollten alle Eingänge von unbenutzten Gattern oder Operationsverstärkern auf ein festes Potential gelegt werden, z.B. auf GND oder VCC. Die Ausgänge dieser unbenutzten Gatter

bleiben natürlich offen! Die Schaltung ist demnach noch anzupassen.

Ein weiterer, derzeit noch nicht besprochener Punkt sind die sogenannten Blockkondensatoren, die direkt an jedem IC platziert werden sollten und zwischen GND und der Versorgungsspannung liegen. Diese stabilisieren die Versorgung, und zwar speziell dann, wenn es durch Schaltvorgänge an den ICs zu Störungen

in der Versorgungsspannung kommen kann.

Das Nichtbeachten der eben beiden genannten Punkte kann im Einzelfall zu einem seltsamen Verhalten der Schaltung führen, wenn nicht sogar zu einer nicht funktionierenden Schaltung. Gerade beim Einsatz von Mikrocontrollern sind die Blockkondensatoren Pflicht (Abbildung 5 und Abbildung 7).

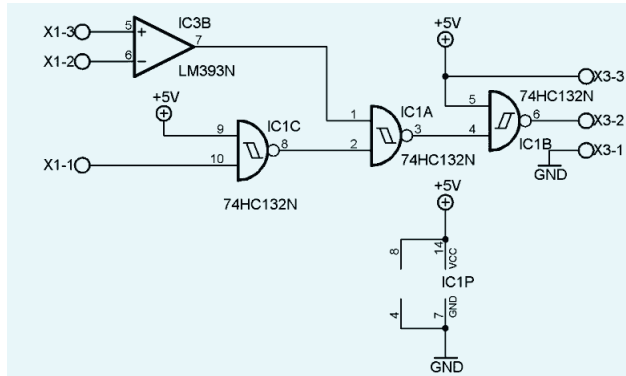


Abb. 5: Logikschaltung

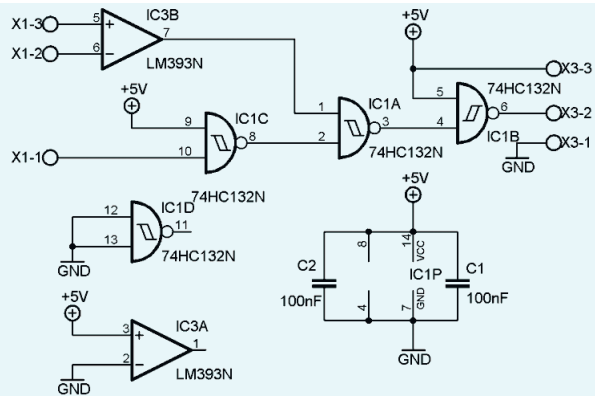


Abb. 7: Gatter angeschlossen, Blockkondensatoren

Eine weitere sehr nützliche Funktion beim Layouten ist das Gate- und Pinswapping, welches im obigen Beispiel erläutert wird. Das Board stellt sich nach Änderung des Stromlaufplans und nach einem „Ratsnet“, wie in Abbildung 6 zu sehen ist, dar. Vergleicht man nun beide Sichtweisen, so könnte man durch Vertauschen von Gattern einfachere, kürzere Verbindungen schaffen. Es lässt sich auch hier nicht generell sagen, wie vorzugehen ist. Nur durch aufmerksames oder geübtes Vergleichen beider Sichtweisen (Stromlaufplan – Board) wird man erkennen, welche Gatter miteinander ge-

tauscht werden sollten.

Sehr auffällig ist auf der Seite von X3 (X3-2) die Verbindung zu Pin 6 des HC132. Man könnte nun entweder das IC drehen oder einfach das Gatter (4,5,6) mit dem Gatter (11,12,13) vertauschen, da dies näher zu X3 ist:

Anschließend wird man auch sehen, dass auch das Gatter (8,9,10) auf der anderen Seite von X1-1 liegt, so dass dieses z.B. mit dem Gatter (4,5,6) getauscht werden kann. Auch was den Operationsverstärker betrifft, so kann hier durch Tausch die Verbindung zu X1 verkürzt werden.

Alle Pins, die mit GND verbunden sind, werden hier nicht angezeigt, sind aber verbunden. Achtung! Ein Pinswap kann natürlich nur zwischen zwei gleichwertigen Pins erfolgen. Ein Pinswap bei den Operationsverstärkereingängen oder bei (anscheinend gleichwertigen) Eingängen von Mikrocontrollern ist also nicht möglich. Die restlichen offenen Verbindungen können nun sehr einfach geroutet werden, das Layout sieht dann wie on Abbildung 10 aus.

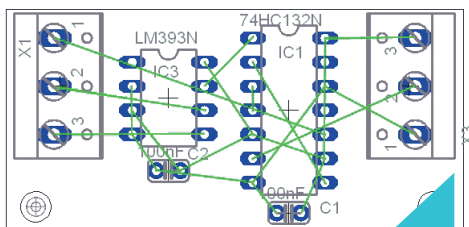


Abb. 6: Gate-Pinswap - Beispiel

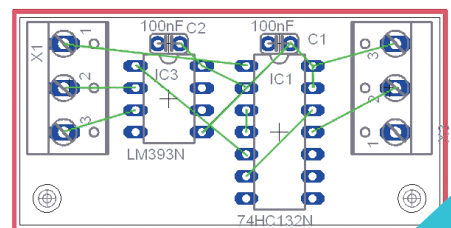


Abb. 9: Pinswap und GND-Fläche

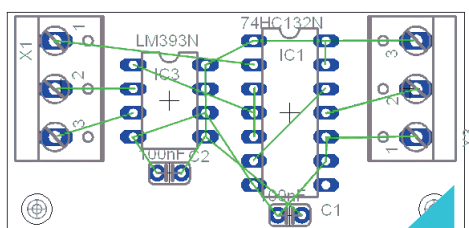


Abb. 8: Nach dem Gateswap

Nun gibt es nach wie vor eine lange Verbindung zwischen X1-1 und Pin 5 des HC132, so dass auch hier noch einmal getauscht wird: (4,5,6) gegen (1,2,3). Wurden diese Veränderungen durchgeführt, so gibt es schon ein etwas klareres Bild (Abbildung 8). Aber auch hier kann vielleicht noch verbessert werden, wenn man sich die Eingangspins der jeweiligen NAND-Gatter anschaut. Ein Pinswap (1-2) und (4-5) des HC132 wird zunächst ausprobiert.

Erstellt man nun, wie im ersten Teil erläutert, eine Massefläche um das Board und verbindet diese mit GND, so wird die Ansicht noch einfacher. Auch werden die beiden Blockkondensatoren direkt an die jeweiligen Versorgungspins gelegt und die ICs noch etwas verrückt, wie in Abbildung 9 zu sehen ist:

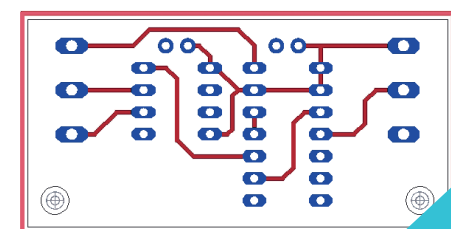


Abb. 10: Fertiges Layout

Spezielle Layouts für spezielle Bauteile

Oftmals sind für bestimmte Funktionalitäten einer Schaltung auch spezielle Bauteile mit besonderen Eigenschaften und daraus folgernd besonderer Leiterbahnführung notwendig. Für das hier genannte Beispiel wird ein Operationsverstärker vom Typ LMC6042 in Verbindung mit einer PH-Elektrode vorgestellt.

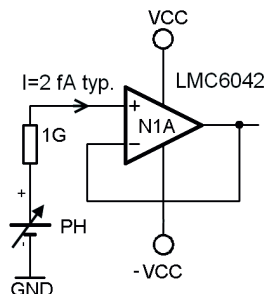


Abb. 11: Spannungsfolger

Die PH-Elektrode, die in Abhängigkeit des PH-Werts eine Spannung ($\pm 400\text{mV}$) erzeugt, hat einen sehr hohen Innenwiderstand, ca. $1\text{ G}\Omega$. Würde man die Spannung an der Elektrode mit einem Multimeter ($R_I = 10\text{M}\Omega$) messen, so würde diese sofort zusammenbrechen. Im Dauerbetrieb würde die Elektrode außerdem auch Schaden nehmen. Daher wird ein einfacher Spannungsfolger mit dem o.g. OP realisiert (Abbildung 11). Der typische Eingangsstrom, der in den

nichtinvertierenden Eingang dieses OP fließt, liegt bei 2 fA (femto Ampere). Somit wird gewährleistet, dass die Elektrode nicht belastet wird.

Dies birgt, was das Layout betrifft, jedoch ein Problem, und das sind eventuelle Querströme, die z.B. durch Feuchtigkeit, Staub oder einfach das Leiterplattenmaterial entstehen und somit den typischen Wert stark negativ beeinflussen können.

Um das Problem zu lösen, sind im Datenblatt des Herstellers (National Semiconductors) einige Hinweise zum Layout gegeben. Zum einen kann ein Leiterbahnring (auf dem Top- und Bottomlayer) um alle Eingangspins inkl. der dort angeschlossenen Bauteile gelegt werden, wie in Abbildung 12 auch realisiert.

Eine andere, aber etwas aufwändigere Variante lt. Datenblatt ist das Umbiegen der Beine des OP und das Anlöten der Bauteile mittels Draht. Da Luft ein sehr guter Isolator ist, umgeht man das Problem, was die Leitfähigkeit des Basismaterial betrifft. Vom Layout her muss dann sichergestellt werden, dass die Fertigung die notwendigen Bauteile nicht einfach bestückt. Dies kann mittels modifiziertem Bestückungsplan und einer angepassten Dokumentation zum Projekt geschehen.

Solche Herstellerempfehlungen sollten immer beachtet werden, um nicht mit unerwarteten Problemen und Nacharbeit konfrontiert zu werden.

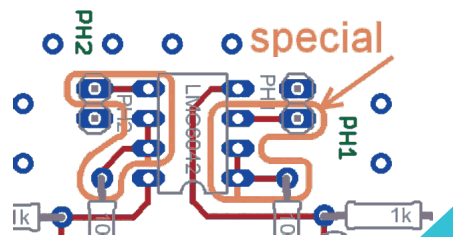


Abb. 12: Leiterbahnring

Man sieht hiermit auch, dass der Layouter nicht nur mit dem Layouten beschäftigt ist, sondern auch mit generellen, physikalischen Fragen in Zusammenarbeit mit der Entwicklung zu tun hat (Abbildung 13).

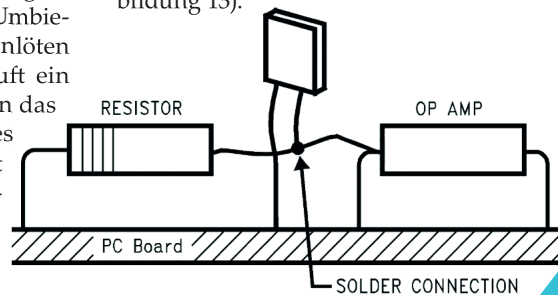


Abb. 13: Drahtverbindung (aus Datenblatt entnommen)

Autorouten kann jeder

...vernünftig layouts kaum einer. Auch wenn dies sehr provokant erscheint, so entspricht es doch noch am ehesten der Praxis.

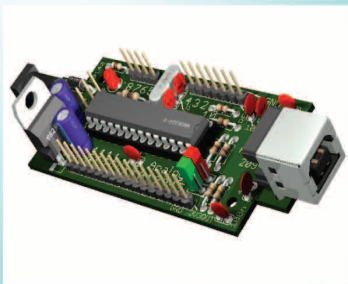
Aus den genannten Themen und Problemen dieses Artikels wird man sehr schnell feststellen, dass ein Autorouter praktisch nie das gewünschte Ergebnis

liefern kann. In einer professionellen Umgebung ist dieses Tool trotz Marketingversprechen nach wie vor kaum ein Thema.

Anzeige



Bausätze, Bücher und Komplettssets



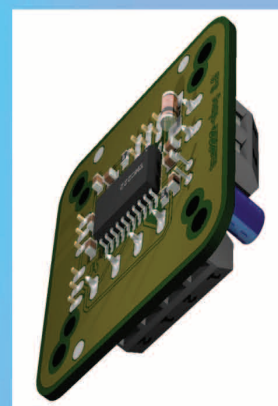
BS1005 - Bausatz Arduino-Clone mit Atmega 328
20,00 €*

Weitere Bausätze (u.a.):

- 06103 - Bausatz USB-Modul mit dem FT2232D 40,00€
- 06104 - Bausatz USB-Modul mit dem FT232RL 20,00€
- 07103 - Bausatz AN910 Programmer für AVR 18,00€



NEU! BS1007 - Bausatz all-in-one AVR Programmer
 - Attiny, Atmega, Atmega - Lieferbar ab April
30,00 €*



OB102 - Bausatz Schrittmotor-modul mit dem TMC222
18,00 €*

* Alle Preise inkl. MwSt. zzgl. Versandkosten

Ing.-Büro B. Redemann
 Mahlower Str. 204
 14513 Teltow

Hier im Shop: www.b-redemann.de

Folgende Punkte zum Thema Autorouter seien angemerkt:

- 1) Ohne eine sinnvolle Platzierung der Bauteile und ohne zu wissen, was man am Autorouter einstellen kann (oder sollte), dürfte das Ergebnis eine Katastrophe werden, und zwar in jeder Hinsicht.
- 2) Nur durch gute Vorarbeit eines Layouters kann der Autorouter eine Hilfe sein.
- 3) Da ein Autorouter sich generell nur um die Leiterbahnen kümmert, kann er nicht sehen, ob z.B. durch Verdrehen eines Bauteils eine einfachere Leiterbahnführung möglich wäre.
- 4) Auch einen Pin- oder Gateswap kann der Autorouter i.d.R. nicht erkennen.
- 5) Ein Autorouter kann nicht wie ein

Mensch „von oben“ sein Layout betrachten, darüber einmal schlafen und Verbesserungen in der Platzierung vor, während und nach dem Layouten durchführen.

6) Bedingt durch die Punkte 3) bis 5) wird der Autorouter i.d.R. mehr Durchkontaktierungen benötigen, als wenn das Board durch einen Layouter kreiert wird.

7) Ein Autorouter ist zwar schneller als ein Layouter, aber das Design wird nicht den Erwartungen entsprechen.

8) Für sehr einfache Layouts oder Testaufbauten kann der Autorouter eine Hilfe sein. Er kann auch eine Hilfe sein, wenn man einen kleinen, unkritischen Bereich einer Leiterplatte automatisch routen lassen möchte.

9) Nach Beendigung des Designvorgangs durch den Autorouter sind praktisch immer (zeitaufwändige) Nacharbeiten notwendig.

Zu Punkt 8) sei allerdings anzumerken, dass einfache Layouts von einer geübten Fachkraft auch schnell und besser erstellt werden können und dass solch ein Fachkraft dann auf automatische Tools verzichten kann.

Fazit: Ein selbst erstelltes Layout, auch im Hobbybereich, und die positive Resonanz von anderen Fachleuten („Oh, das sieht wirklich gut aus“) steigern das Selbstbewusstsein. Ich kann daher nur empfehlen, beim Leiterplattendesign kreativ und flexibel ein eigenes Layout zu erstellen, auch wenn dies etwas Zeit braucht.

Negativbeispiel eines Layouts

Am Ende kann ich es mir nicht verkneifen, eine verkorkste Leiterplatte zu zeigen, die von einem Unerfahrenen ohne Vorkenntnisse layoutet wurde. Leider wurde auch nicht nachgefragt, ob man bestimmte Dinge doch hätte anders machen können, und leider trifft man manchmal auch auf beratungsresistente Menschen... (siehe Abbildung 14)

Folgende Dinge sind zu bemängeln:

- Es gibt keine Befestigungsbohrungen, um die Leiterplatte zu montieren (war notwendig)
- Die Leiterbahnen für die Signalführung und im Leistungsteil sind zu schmal
- Im Leistungsteil wurden die Leiterbahnen mit blankem Kupferdraht verstärkt (Kurzschlussgefahr)

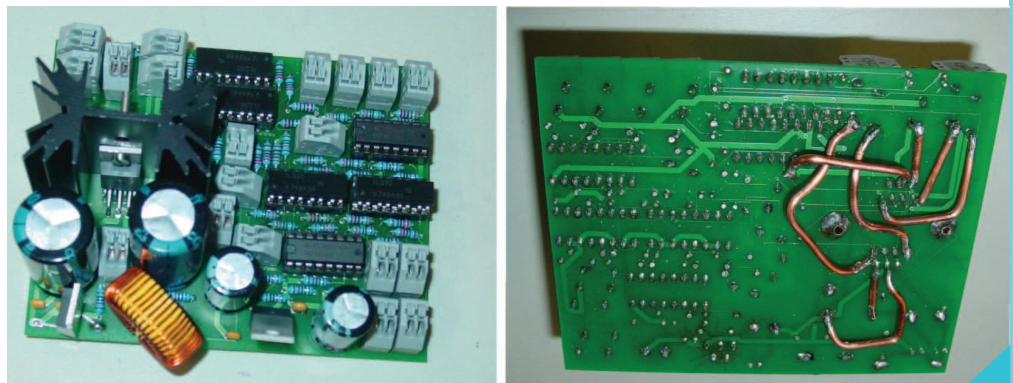


Abb. 14: Verkorkste Leiterplatte - Ein Produkt für den Elektroschrott

- Sehr oft 90° Winkel im Layout
- Die Schraubklemmen sind teilweise nicht zu erreichen
- Die Elkos liegen direkt neben einem Kühlkörper, der auch warm wird
- Das Rastermaß für die Widerstände passt nicht zum realen Bauteil
- Beim Spannungsregler und der Drosselspule unten links wurde ein falsches Bauteil ausgesucht

Es gäbe noch mehr zu sagen

Die beiden Teile dieses Artikels zu diesem Thema sollten dazu anregen, sich auch im Hobbybereich über die Fragen und Probleme zum Leiterplattenlayout nicht nur Gedanken zu machen, sondern auch bestimmte Richtlinien umzusetzen.

Es konnte allerdings aufgrund der Kom-

plexität dieses Themas nicht alles gesagt werden, es sollte aber ausreichen, um ein Gefühl für dieses Thema zu bekommen.

Neben den genannten Links im ersten Teil möchte ich auf das Buch „Leiterplattendesign“ von Jürgen Händschke verweisen, welches sich mit weiteren The-

men, u.a. mit EMV, Bestückung und Test rund um das Design beschäftigt. Es ist zu finden im Eugen G. Leuze Verlag und hat die ISBN 3-87480-219-1.

In diesem Sinne: Frohes Layouten!

Links / Datenblätter

<http://www.national.com/ds/LM/LMC6042.pdf>

Links: Siehe Teil 1, Ausgabe 7

TouchBox

Lautsprecher zum Anfassen

Tobias Göpel <mail@tobiasgoepel.de>

Elektronische Musik

Mit Instrumenten verändert sich die Musik. Technische Erkenntnisse führten zum Theremin, benannt nach dem russischen Wissenschaftler Lev Termen [1]. Es ist eines der ersten elektrischen Klangerzeuger, entwickelt im 20. Jahrhundert. Im Laufe der darauf folgenden 90 Jahre bis heute entstanden viele elektrische und elektronische Klangerzeuger [2]. Völlig unbekannte Klänge entwickelten sich daraus. Andererseits wurden Instrumente wie das Klavier, die Orgel, Trommeln oder Streichinstrumente kopiert. Die Kopie wird kreativ genutzt und günstiger verkauft als viele Originale. Laptops und Smartphones (siehe z.B. [3]) haben Konzertpotential, nicht nur über mp3-Aufnahmen. Es existiert eine Unzahl von Instrumenten als Plugins. Die virtuellen Klänge sind heute ebenso vielschichtig wie die Mechanik, mit der sie erzeugt werden. Tastaturen, Joysticks oder Handys mit Beschleunigungssensoren triggern Klänge. Über die MIDI-Schnittstelle [4] oder Erweiterungen wie das Open Sound Control (OSC) [5] kann der Klang unabhängig von der Mechanik gewählt werden. Mit der Taste ‚Play‘ lassen sich Musikstücke starten, über das MIDI-Protokoll sogar ganze

Melodien, Rhythmen, einzelne Noten oder Akkorde. Durch das OSC, welches UDP/IP zum Datenversand nutzt, können technisch zunehmend komplexere Instrumente auch über das Internet gesteuert werden.

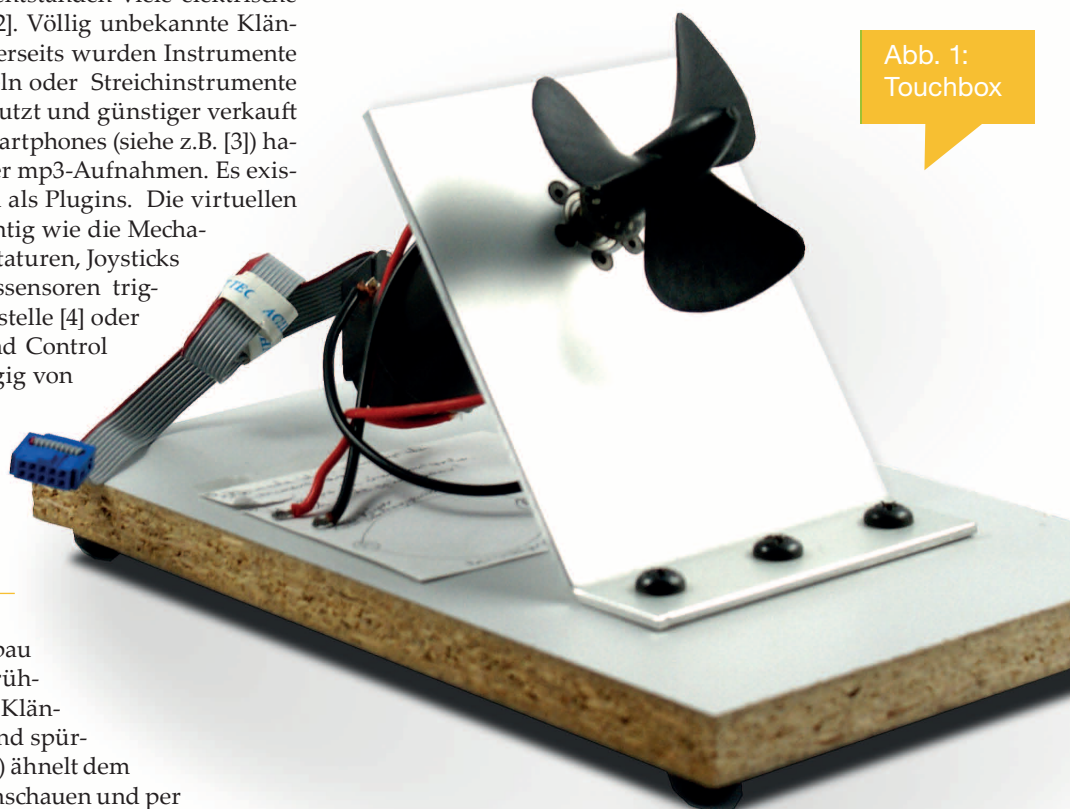


Abb. 1:
Touchbox

TouchBox

Dieser Artikel beschreibt den Aufbau eines Lautsprechers, der durch Berührung Musik erzeugen kann. Er gibt Klänge wieder und macht sie für die Hand spürbar. Diese „TouchBox“ (Abbildung 1) ähnelt dem Touchscreen, mit dem man Bilder anschauen und per Hand verändern kann; sie dreht sich im Kreis, klingt im Ohr und vibriert in der Hand, gibt also akustische und haptische Reize wieder. Bei Haptik handelt es sich um Reize, die man mit Fingern spüren kann [7]. Mit der TouchBox können Klänge getriggert werden, die diese Reize erzeugen: Je nach dem Winkel, in den der Motor über einen Griff vom Musiker gedreht wird, können Klänge erzeugt werden. Auch Tonhöhe oder Lautstärke können über den Winkel verändert werden. Der Motor kombiniert

zwei Eigenschaften:
Er simuliert eine Taste auf einer Tastatur, da er Klänge triggert, wenn er auf einen bestimmten, festgelegten Winkel gedreht wird. Diese Position soll „Tawie“ (Tastenwinkel) genannt werden. Wird der Griff auf den Tawie gedreht, wird der Klang getriggert, und auf dem Griff gespürt.

Aufbau

Auf der Motorwelle des Prototyps ist ein Schiffspropeller angebracht. Er wird als Griff in die Hand genommen. Seine Flügel geben Klänge über ihre Großflächigkeit an die Luft ab, sobald der Motor beginnt, zu vibrieren. Es gibt am Griff also einen Teil, der den Klang hochfrequent an die Luft abgibt, und einen anderen, der fest in die Hand genommen werden kann. Wie bei vielen Musikinstrumenten existieren beide getrennt voneinander, damit die Berührung hohe Frequenzen aus dem Klang nicht filtert.

Elektromotoren und Lautsprecher sind in ihrer physikalischen Modellierung ähnlich: Sie wandeln elektrische in mechanische Energie um. Elektromotoren übertragen Kräfte an ihre Umgebung, wenn sie fest verankert sind. Auch Lautsprecher sind fest verankert: Sie bringen Luft in Schwingung, ohne sie vollständig von sich zu stoßen. Die TouchBox vereint beide Eigenschaften in einem einzigen Motor.

Der Anschluss

Mit einem Klinkenkabel kann man den Motor von einem Mp3- Player, einem Smartphone oder einem Laptop aus ansteuern. Das Kabel wird mit einem Verstärker verbunden, der den Klang mit weitaus mehr Energie an den Motor weiterleitet. Da es sich um einen einzelnen Motor handelt, reicht ein Monoklinkenkabel.

Der Motor ist mit einem Encoder ausgestattet, der dessen Position misst. Die Positionsdaten werden durch eine Schaltung von Doepfer (CTM64) elektronisch über MIDI an einen Laptop übermittelt. Die MIDI-Schnittstelle ist im musikalischen Bereich Standard für den Datenaustausch. Jeder moderne, nach 1980 gebaute Klangerzeuger kann über MIDI dazu angeregt werden, Klänge zu erzeugen. MIDI wird nicht genutzt, um Aufnahmen zu machen, da die Bandbreite des MIDI-Protokolls recht gering ist.

Auf vielen modernen Klaviaturen gibt es ein Rad, mit dem die Tonhöhe eines Klanges verändert werden kann. Dreht man an dem Rad, werden MIDI-Daten erzeugt. Sie ändern die Tonhöhe, Lautstärke und diverse andere Eigenschaften des Klanges.

Im Falle der TouchBox kann nicht nur der Musiker den Klang steuern, indem er an dem Griff dreht. Auch der Klang selbst bewegt den Motor. Diese Bewegung ändert die Position des Motors und erzeugt Resonanz, einen geschlossenen Kreislauf des Klangs. Resonanz ist eine Schwingung, die sich selbst erhält. Die TouchBox ist so programmiert, dass der Motor vibriert, wenn man den Griff in den Tawie dreht. Bewegt die Schwingung danach den Motor wieder in den Tawie zurück, erzeugt sie weitere Klänge. Die Resonanz kann bewusst musikalisch genutzt werden.

Code

Für Musikinstrumente gibt es eine Landschaft von Programmiersprachen. Ihre Möglichkeiten sind auf die Erzeugung und Verarbeitung von auditiven und visuellen Reizen spezialisiert.

Die Open-Source Lösung Pd [8] oder dessen kommerzieller Vater Max/MSP [9] sind grafische Programmieroberflächen. Die folgenden Kapitel beschreiben, wie Max/MSP funktioniert, und wie die grafische Benutzeroberfläche für die TouchBox aufgebaut ist. Für Interessierte der Open-Source Landschaft lässt die Software sich relativ schnell in Pd umsetzen.

Der Code soll folgende Aufgaben erfüllen:

1. Die Positionsdaten werden durch den Encoder erzeugt und triggern Klänge am Tawie.
2. Lautstärke, Tonhöhe und viele andere Eigenschaften des Klanges werden durch die Position des Griffes gesteuert.
3. Das Instrument, das den Klang erzeugt, soll frei wählbar sein

Wie diese Aufgaben umgesetzt werden, wird in den folgenden Absätzen beschrieben.

MIDI In

Abbildung 2 ist ein typischer Auszug aus Max/MSP. Die Daten, die am Motor abgelesen werden, kommen über ein MIDI-Eingangsgesetz, hier den IAC-Driver, in Max/MSP an. Der Block [r midiinfo] erhält die Namen aller möglichen MIDI-Eingangsgesetze und versorgt das Menü, das darunter liegt. Man kann mit dem Menü wählen, aus welchem Gerät die Daten des Potentiometers kommen werden. Dann werden die MIDI-Daten verarbeitet (siehe Abbildung 5):

1. Der Block [p motorPosition] liest aus den MIDI-Daten die Position des Motors aus. Die Positionsdaten werden dort in eine Auflösung von 6 Bit bereitgestellt, das heißt, sie können Werte von 0 bis 127 annehmen.
2. Über den Block [s position] werden die Positionsdaten innerhalb von Max/MSP überall hingeseendet, wo sie gebraucht werden. In Max/MSP bedeutet

OCTAMEX
electronics for professionals

10%
Rabatt

Gutscheincode
FE 7211

Jetzt neu!
Günstige RF-Transceiver
433 MHz & 868 MHz

Anzeige



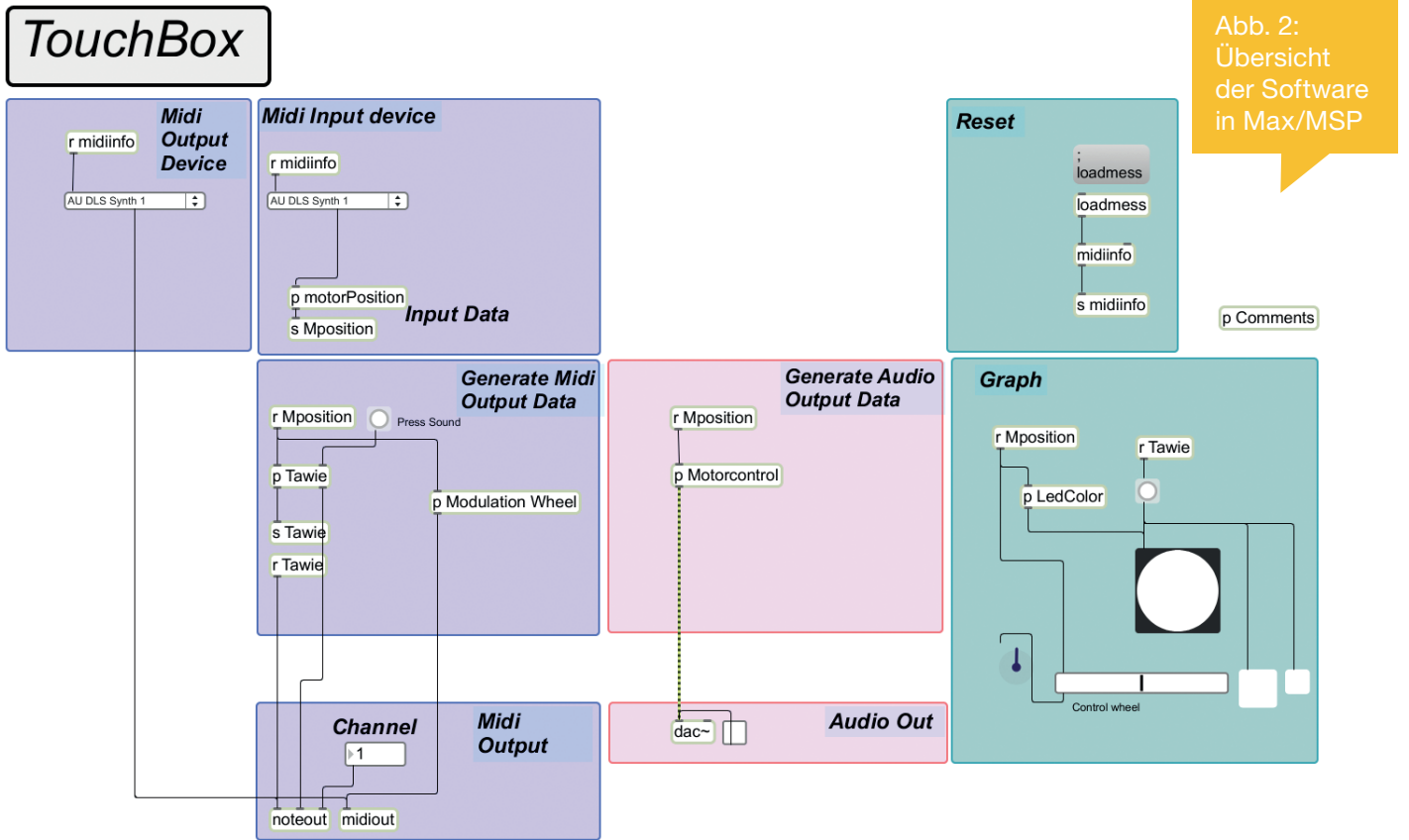
- Basismaterialien
- PCB Chemikalien
- Speziallamine
- FSK-Funkmodule
- Sensoren & ICs
- GPS & GSM Module
- Steckverbinder
- Löttechnik

www.octamex.de

Der Rabatt wird nur einmalig bei Onlinebestellung gewährt. Es gelten die Allgemeinen Geschäftsbedingungen der Hillemann & Schöne GbR 01157 Dresden.

[s xyz] stets, dass die Variable xyz versendet wird. Mit diesem Block lassen sich unübersichtliche Verkabelungen auf dem Bildschirm vermeiden.

3. Der Block [print input] gibt, wenn man das will, auch in einem Terminal außerhalb von Max/MSP an, welches MIDI-Eingangsgerät gewählt wurde.



MIDI Out

Abbildung 4 zeigt, wie die Daten an einen Klangerzeuger weitergesendet werden. Wie beim MIDI-In kann auch beim MIDI-Out der Klangerzeuger, der genutzt werden soll, über ein Menü ausgewählt werden. Hier handelt es sich um ein einfaches Instrument, den AU DLS Synth, einem Teil von Max/MSP. Mit ihm können beispielsweise Klavierklänge wiedergegeben werden.

Alle Boxen auf der Programmieroberfläche, die mit r beginnen, also zum Beispiel [r zyx], sind Daten, die von anderswo mit einer [s zyx] Box hergesendet werden:

1. [r NoteOn] liefert immer dann ein Signal, wenn die Position des Motors genau in der Mitte liegt, also auf dem oben beschriebenen Tawie.
2. [r PosCC] gibt die Positionsdaten auf einem typischen MIDI Datenkanal, nämlich MIDI ControlChange 1, aus. Dieser Kanal wird am häufigsten benutzt, um Klänge in elektronischen Musikinstrumenten zu beeinflussen. Dadurch kann Vibrato eingeleitet werden. Im Falle des Motors wird das Vibrato erzeugt, wenn der Griff weit nach rechts gedreht wird.
3. Mit [r Program] kann das Instrument ausgewählt werden, das von dem AU DLS Synth genutzt werden soll.
4. [makenote 100 3000] erzeugt jedesmal eine Note, wenn es von [r NoteOn] ein Signal gesendet bekommt. Darüberhinaus

schickt es 3000ms später ein NoteOff, damit der Ton auch wieder beendet wird. Beides sendet er weiter an...

- 5....[noteout], das die Noten im Synthesizer triggert.
6. [midiout] gibt die ControlChange-Daten an den Synthesizer weiter, um dort beispielsweise Vibrato zu erzeugen.

Auf Abbildung 3 ist eine grafische Benutzeroberfläche zu sehen, die für die TouchBox zur Verfügung steht. Auf ihr ist zu sehen,

1. wo der Griff sich gerade befindet,
2. welche Tonhöhe für den Klang eingestellt wurde (Pitch),
3. welcher Klang, zum Beispiel Klavier oder Geige, gespielt werden soll (Program),
4. und welcher MIDI- Eingang und welcher Ausgang benutzt werden.

Die Benutzeroberfläche (siehe Abbildung 3) kann all die Daten live darstellen, die im Programm vorkommen. Der lilane Drehregler stellt die Position der TouchBox dar. Wenn der Regler nach oben schaut, triggert er den Klang und die Benutzeroberfläche blitzt auf. Der Klang ist gleichzeitig am Griff hör- und fühlbar.

...ihr Klang

Die TouchBox ist eine Chimäre [10], also ein Mischwesen, das unterschiedliche Reize in sich vereint.

Akustische und haptische Reize fließen ineinander über. Taube

Menschen spüren Klänge, wenn sie sie anfassen können- was mit der TouchBox möglich ist.

Jede Waschmaschine soll möglichst unhörbar sein, während sie

sich mit großen Geschwindigkeiten bewegt. Im Gegensatz dazu will jeder Rocker, dass sein Motorrad überall zu hören ist. Motoren sind überall zu hören und Lautsprecher können viel bewegen. Diese Eigenschaft, zwei Reize zu vereinen, sind ein Grundstein vieler Musikinstrumente bis hin zur menschlichen Stimme, die Menschen über Atemmuskulatur spüren und steuern können. Werden beide Reize genutzt, verdoppeln sich die Möglichkeiten, den Klang wahrzunehmen. Der Bogen, den Streicher in der klassischen Musik nutzen, um den Klang zu beeinflussen, macht ihr Instrument vielseitiger. In jedem Klavier steckt eine komplexe Mechanik, um das Gefühl der Finger für den Anschlag der Tasten zu verfeinern.

Die TouchBox realisiert diese Idee elektromechanisch für virtuelle Instrumente. Da sie keinen festen Körper hat, ist Ihr Klang nicht festgelegt. Nicht nur seine Akustik ist frei programmierbar, auch das Gefühl, das er abgibt, ist es.

Jeder, den die Erweiterung des Instruments, seiner klanglichen oder physikalischen Programmierung reizt, ist herzlich dazu eingeladen, es weiter zu entwickeln oder vor dem Team ein Konzert zu halten.

Links / Literatur

[1] <http://de.wikipedia.org/wiki/Theremin>

[2] <http://www.synthzone.com/>

[3] <http://hexler.net/software/touchosc>

[4] http://de.wikipedia.org/wiki/Musical_Instrument_Digital_Interface

[5] http://de.wikipedia.org/wiki/Open_Sound_Control

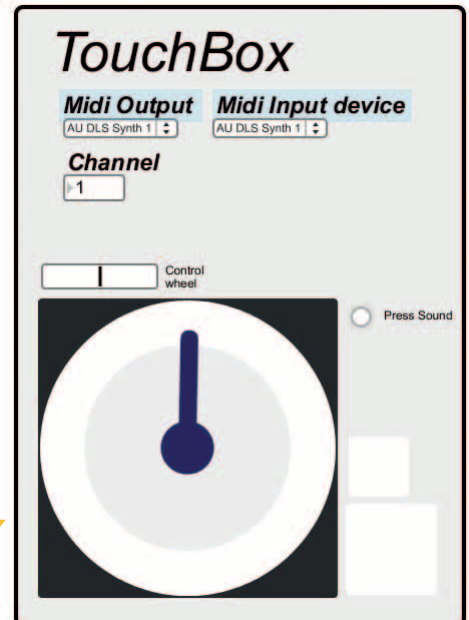
[7] <http://de.wikipedia.org/wiki/Haptik>

[8] <http://puredata.info/>

[9] <http://cycling74.com/>

[10] [http://de.wikipedia.org/wiki/Chimäre_\(Mythologie\)](http://de.wikipedia.org/wiki/Chimäre_(Mythologie))

Abb. 3: Benutzeroberfläche der Touchbox



Midi Out

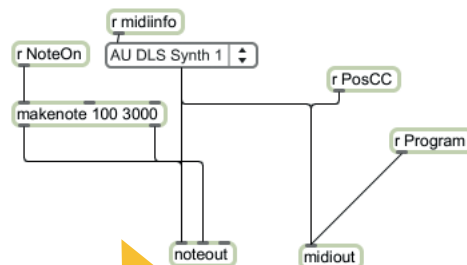


Abb. 4: Die Positionsdaten des Motors werden genutzt, um über das MIDI-Out Noten zu triggern und den Klang zu beeinflussen.

Input

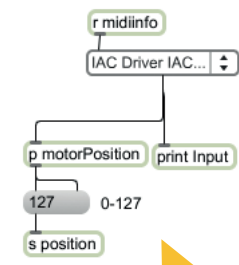


Abb. 5: Die MIDI-Daten werden über Max/MSP aufgenommen und so verarbeitet, dass die Position des Motors ausgelesen werden kann.

Anzeige

Sie wollen mehr?

Mehr Verantwortung, mehr Leistung, mehr Erfolg? Würth Elektronik eiSos ist der größte europäische Hersteller von passiven elektronischen Bauelementen & Elektromechanik für Anwendungen in fast jeder Branche. Durch unser jährliches Wachstum im zweistelligen Bereich, sind wir eines der erfolgreichsten Tochterunternehmen der Würth-Gruppe. Mit unseren ca. 4.800 Mitarbeitern bieten wir unseren Kunden einen Service, der uns klar vom Markt abhebt. Dabei ist das Rückgrat das dichte Beratungsnetz im Innen- und Außendienst.



Für die Stärkung unseres Service am Kunden suchen wir Sie als

**(Junior-) Vertriebsmitarbeiter (m/w)
für PLZ 85, 88 & 89**

Haben wir Ihr Interesse geweckt?

Dann besuchen Sie unsere Homepage www.we-online.de. Dort erfahren Sie mehr über die Aufgaben und die benötigten Qualifikationen.

Ihre Chance

Durch unser rasantes Wachstum bieten wir Ihnen sehr abwechslungsreiche Tätigkeiten, die Ihnen schnell die Übernahme und den Ausbau von Verantwortung ermöglichen. Vor allem aber eine Atmosphäre, die garantiert Spaß bei der Arbeit verspricht. Ihren Einsatz belohnen wir mit vielfältigen Möglichkeiten sich persönlich sowie fachlich weiterzuentwickeln.

Ihre Bewerbung

Nicole Adelman (nicole.adelman@we-online.de · Tel: 07942/945-5425) freut sich auf die Zusendung Ihrer Bewerbungsunterlagen mit Angabe Ihres frühestmöglichen Eintrittstermins und Ihrer Gehaltsvorstellung.

GNUBLIN Peer-Revier

Fehlersucher und Verbesserungsvorschläge

<http://elk.informatik.fh-augsburg.de/hhwiki/Gnublin>

Status Quo

In den letzten Ausgaben haben wir bzw. Hubert Hoegl regelmäßig von unserem Open Source Projekt GNUBLIN berichtet. Für die Ausbildung soll ein kleines günstiges Open-Source Board für embedded GNU/Linux entstehen.

Mittlerweile sind wir soweit, dass der Schaltplan und das Layout für die erste Version steht. An dieser Stelle wollen wir den Schaltplan und das Layout gerne zur Verfügung stellen, um es von anderen Leuten prüfen lassen:

Wir freuen uns über jeden, der Lust hat den Schaltplan bzw. das Layout zu prüfen und Fehler zu suchen bzw. einfache Verbesserungsvorschläge zu bringen, so dass wir vielleicht im ersten aufgebauten Musterso wenig wie möglich an Änderungen benötigen. Geprüft werden kann entweder in den PDF Dateien [1], die es als Download gibt. Besser ist es natürlich wenn man direkt mit KiCAD den Schaltplan und das Layout prüft.

Installation Ubuntu:

```
sudo apt-get install kicad
```

Installation Windows:

Es gibt einen Installer für Windows unter [2].

Ist KiCAD installiert lädt man sich das Downloadpaket [3] herunter, entpackt es auf dem lokalem Computer und öffnet mit KiCAD die Projektdatei Gnublin.pro. Es öffnet sich die Projektübersicht von KiCAD (siehe Abbildung 1). Die Datei Gnublin.brd ist die Platine. Mit Doppelklick auf Gnublin.sch öffnet man den Schaltplan. Bei mir kam eine Fehlermeldung, welche ich aber erstmal weggeklickt hatte.

Der Schaltplan besteht aus mehreren Seiten. Leider ist die Navigation etwas komplizierter. Rechts oben gibt es in der rechten Leiste das zweite Icon „Navigation in der Schaltplanhierarchie: Das muss aktiviert werden. Jetzt kann man auf der ersten Schaltplanseite rechts unten die Sheets (blauer Rahmen) auf einen Unterplan klicken. Will man aus einem Unterplan wieder zurück auf die Hauptseite zurück, einfach nochmals sicherstellen, dass rechts das Icon noch aktiv ist und dann außerhalb des Schaltplans auf die Arbeitsfläche klicken.

Im Layout bzw. Board (Gnublin.brd) kann man rechts die einzelnen Layer ein- und ausblenden. Mit der Maus kann in die Schaltung gezoomt werden bzw. wieder die Standardansicht erzeugt werden.

Gerne prüfe ich die Layouts noch mit einem Gerberviewer. Hier sieht man „komische Muster“, falsche Leitungen, etc. die oft auf Fehler hindeuten. In KiCAD kann man die Gerber Daten über Datei Plotten erzeugen. Über die Oberfläche kann man sich die gewünschten Layer auswählen und als Ausgabeformat Gerber einstellen. Anschliessend öffnet man diese Dateien mit dem Programm gerbv. Unter Ubuntu kann man dies wieder einfach mit sudo apt-get install gerbv installieren. Hat man die Daten in einen eigenen Ordner exportiert, kann man direkt von der Kommandozeile aus alle Dateien wie folgt öffnen:

```
gerbv *
```

Wir freuen uns über Eure Rückmeldungen. Am besten schickt Ihr die Daten an die Mailingliste [4]. Sollte dem einen oder anderen dies zu aufwändig sein, reicht auch eine einfache E-Mail an sauter@embedded-projects.net bzw. hubert.hoegl@fh-augsburg.de.

Vielen Dank für eure Mithilfe.

Links

[1] http://elk.informatik.fh-augsburg.de/gnublin-cdrom/eda/Gnublin_2011-06-25/ (PS Dateien)

[2] Installer <http://kicad.sourceforge.net/wiki/Downloads>

[3] <http://elk.informatik.fh-augsburg.de/gnublin-cdrom/eda/>

[4] <https://www.rz.fh-augsburg.de/sympa/info/elinuboard>

Abb. 1: KiCAD Projektübersicht



Abb. 1: KiCad

Hinter den Kulissen

USB Netzwerkkarte von Innen

Benedikt Sauter <sauter@embedded-projects.net>

„Hinter den Kulissen“ soll einen Einblick Produkte aus dem Laden geben. Wie sind die Schaltungen aufgebaut, welche Bausteine sind integriert, wie sind mechanische Einbauten bzw. Gehäusekonstruktionen gelöst? Diese Artikel entstehen immer parallel während dem systematischen Zerlegen. Erstes „Opfer“ für diese Artikelreihe ist ein etwas ins Alter gekommene USB zu Netzwerkadapter von der Telekom.

Abb. 1: USB Netzwerkadapter



Erster Funktionstest

Mit einem USB-Kabel an einen GNU/Linux PC verrät sich der Chip mittels dem Aufruf des Befehls `lsusb` als D-Link Gerät. Auf dem Gehäuse steht genau: Teledata Fast Ethernet USB (siehe Abbildung 3).

```
sauter@meinpc:~$ lsusb
```

```
Bus 006 Device 002: ID 2001:400b D-Link Corp. 10/100 Ethernet
```

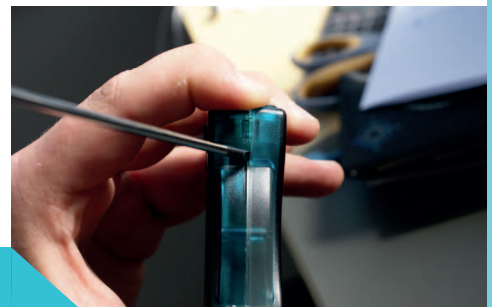


Abb. 2: Gehäuse öffnen

Knackt das Gehäuse

Meine erste Vermutung, dass sich unter den kleinen Gummifüsschen Schrauben befinden, hat sich leider nicht bewahrheitet.

Dann Hilft wohl nur die Schraubenzieher-Methode (Abbildung 2). Nach dem Drücken auf die entsprechenden Plastik-

nasen fiel das Gehäuse auseinander und die Platine war relativ schnell befreit.

Die Schaltung

Als erstes fällt auf, dass der Adapter im Wesentlichen aus einem großen TQFP Chip besteht, welcher mit dem USB- und Netzwerkstecker verbunden ist. Was mir außerdem gleich auffiel, auf der Rückseite der Platine ist ein Platzhalter für einen weiteren TQFP Chip, welcher gar nicht bestückt ist. Es könnte für die sogenannte Second-Source vorgesehen sein. In kommerziellen Schaltungen wird oft für wichtige Bauteile eine zweite Alternative definiert und mit in die Schaltung integriert. Kann ein Chiphersteller einmal nicht liefern kommt es beim Produkthersteller nicht zu Lieferengpässen, er kann einfach auf die Alternative umschwenken. Was sich aber dahinter bei dem USB-Netzwerkadapter verbirgt lichtet sich eventuell noch später.

Im Wesentlichen sieht die Schaltung wie in Abbildung 4 dargestellt aus.

Der ADM8511 von ADMtek ist die Basis der Schaltung. Mittels Internet findet man das Datenblatt sehr schnell. Im Datenblatt steht: Pegasus IIP USB/Fast Ethernet with MII Interface (siehe Abbildung 6).

Exkurs: Hier ein Auszug aus Wikipedia + Abbildung (auch zu finden im Bereich Links):

„Der Ausdruck Media Independent Interface (MII) wurde bei Fast Ethernet für Netzwerkkomponenten eingeführt. Er teilt Fast-Ethernet-Chipsätze in zwei Komponenten auf: Das Media Dependent Interface (MDI) und das MII. Die Schnittstelle zwischen MII zu MDI ist eine Teilkomponente des Ethernet-Chipsatzes und bei allen Herstellern und allen Medien (Kabel, Fiber) identisch. Das MDI hingegen ist der Teil des Chipsatzes der spezifisch für das jeweilige Medium (Kabel, Fiber) (Transceiver) ausgelegt werden muss. Er beinhaltet auch die physikalische (elektrische, optische) und mechanische Schnittstelle zwischen



Abb. 3: Teledata von Unten

dem Physical Medium Attachment (PMA) und dem Medium (Kabel). Bei klassischem 10-Mbit-Ethernet nannte man die Schnittstelle zwischen MII und MDI auch AUI-Schnittstelle, der Transceiver entspricht dem MDI, der Rest der Karte beginnt mit dem MII und endet am jeweiligen Systembus (PCI, EISA usw.).“

Der Controller ist also dafür gemacht, genau solche Adapter zu bauen. Primärer Fokus der Hersteller ist bei solchen Chips immer, dass die BOM (Bill of Material oder zu deutsch Stückliste) sehr gering ausfällt. Jeder zusätzliche Baustein kostet bei Auflagen > 1.000.000 Stück richtig Geld. Kurz mal ins Datenblatt geschaut ob ich etwas spannendes entdeckte:

Der integrierte USB-Controller bietet 4 Endpunkte für die Kommunikation und unterstützt die Standby-Funktion von USB vollständig. Warum dies explizit erwähnt ist? Eventuell war das damals nicht „Standard“. Wenn das USB-Gerät von keinem Treiber aktiv verwendet wird, kann es der USB-Host schlafen legen und so den Stromverbrauch extrem reduzieren. Über eine Signalierung per USB kann das Gerät wieder geweckt werden. Intern befinden sich an den Kommunikationsschnittstellen - egal ob USB oder Netzwerk - FIFOs für eine effektive Abarbeitung der Transfers. Für USB sind diese 24K, 64 Byte oder 2K groß. Für das Netzwerk wird mit 24K ein Puffer für den flüssigen Datenstrom bereitgestellt. Jetzt könnte man kurz überlegen: wie schnell muss die Logik im Kern des Chips sein, um 100 MBit/Sekunde ohne Stau abzuwickeln? 100 MBit/s sind 12.5 MByte/s d.h. pro Byte: 12.5 MByte/s sind 131072 Byte pro Sekunde. Das sind dann 76.29 nS pro Byte. Bei 24K können so 1,875 mS gepuffert werden. So schnell muss die interne Logik gar nicht mal sein. Ein AVR 8 Bit Prozessor kann z.B. bei 16 MHz Befehle mit 62.5 nS ausführen. Es gibt natürlich auch Befehle, die mehr als ein Takt brauchen. Aber mit einem Prozessor der etwas über dieser Liga liegt (Richtung ARM7 oder UC3 von Atmel) könnte man bereits den Datenstrom verkraften.

Soviel aber genug zu dem Controller. Auf der Schaltung gibt es noch ein EEPROM von ATMEL 93C64. Dies ist ein serielles EEPROM. Hier kann man eine eigene USB-Hersteller- und -Produkt-ID hinterlegen, so dass sich das Gerät beim Anstecken an den Computer als das von der Firma, die als Hersteller auf der Verpackung steht ausgeben kann. Der Baustein in der Schaltung bietet 64 16 Bit

Platinenlayout

Ingesamt kamen beim Layout (siehe Abbildung 5) wohl drei Netzklassen zum Einsatz. Eine Netzklasse beschreibt in einem Platinenlayout-Programm primär die Leiterbahnstärke. Für Stromführende Leitungen werden in der Regel bereitere Bahnen als für einfache Steuerleitungen benötigt. Für die Berechnung der Breite der Leiterbahn gibt es viele kostenfreie Rechner im Internet (abhängig von Strom und Kupferdicke auf der Platine).

Schön sind die Übergänge an den Lötdurchkontaktierungen beim Netzwerkstecker. Hier sieht man wie es mit einem „Tropfen“ vom Pad zur Leitung geht. Dadurch hat man eine maximale Übergangsfläche und die Chance, dass die Leiterbahn vom Pad durch das Löten getrennt wird, wird so minimal. Die Massefläche verbindet sich zu Durchkontaktierungen mit Thermal-Pads. Das sieht man am typischen Muster.

Dazu gibt es wieder einen kurzen Auszug aus Wikipedia (siehe Abbildung 7):

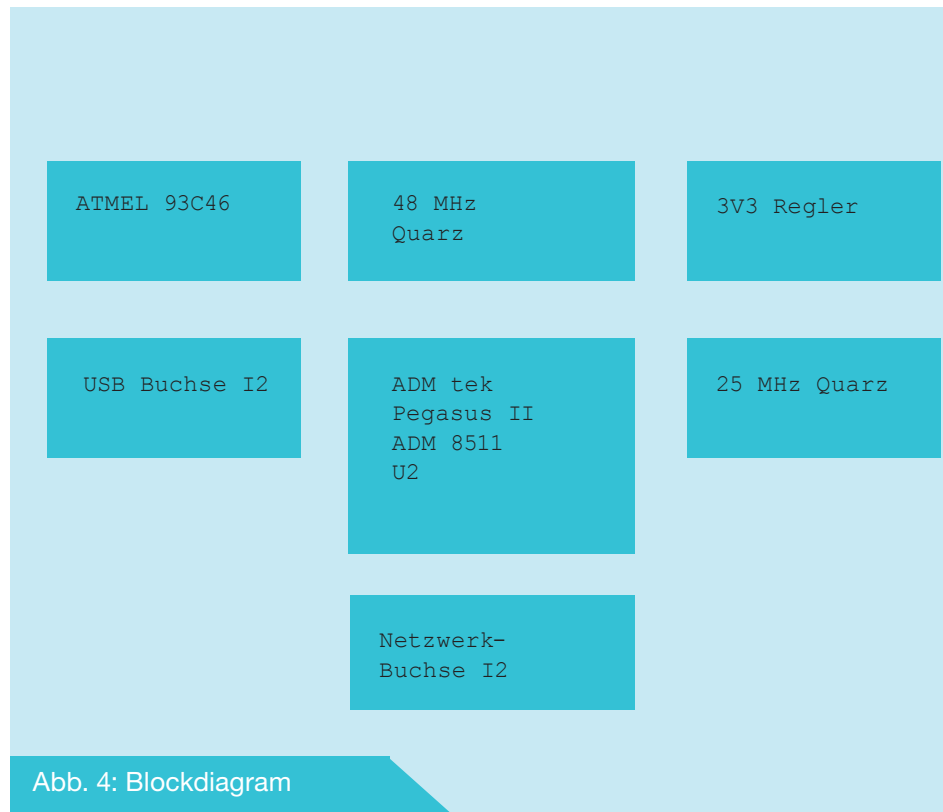


Abb. 4: Blockdiagramm

breite nichtflüchtige Register für die eigenen Daten an.

Eine Sache fällt mir noch ein. Wie ist es gelöst, dass die Netzwerkkarte eine eindeutige MAC-Adresse besitzt? Ein Blick ins Datenblatt:

Diese wird wohl auch im EEPROM abgelegt. Beim Lesen des Datenblatts habe ich gesehen, dass das EEPROM über ein vom Hersteller mitgeliefertes Programm programmiert werden kann. Die Programmerroutinen sind im ADM8511 integriert. Es gibt eine sogenannte „System Register Table“ in der mit Offsets an-

gegeben Register beschrieben sind, auf die man lesend und schreibend zugreifen kann. Im Anhang 4 des Datenblatts ist beschrieben, wie man mit „Vendor-Requests“ über die USB-Schnittstelle auf diese Register zugreifen kann. Dem einen oder anderen werden bestimmt auch einige Ideen soeben kommen, was man da alles schönes machen könnte :-). Aber jetzt sei das mal genug zu den Bausteinen.

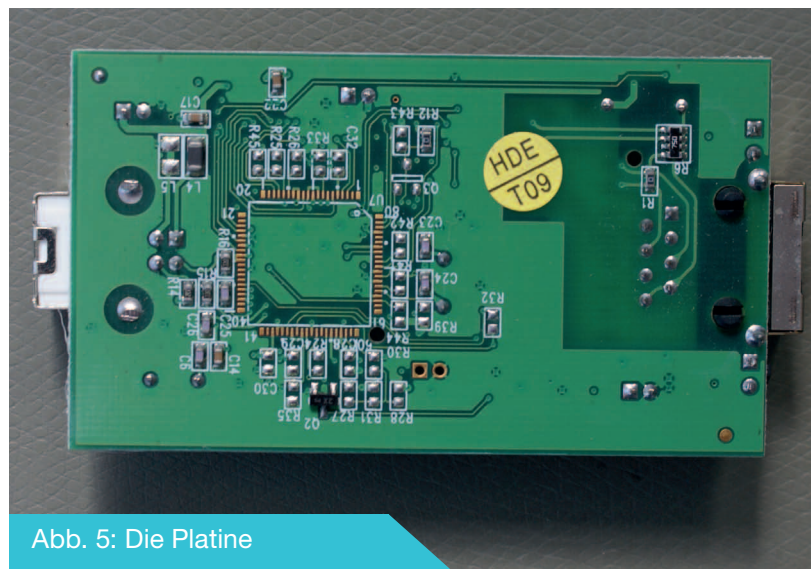


Abb. 5: Die Platine

„Als Thermal Pad oder Wärmefalle bezeichnet man auf Leiterplatten solche Lötflächen von elektronischen Bauteilen, die nicht vollflächig, sondern nur mit dünnen Stegen an größere Kupferflächen angeschlossen werden. Die Verkleinerung des Querschnitts verhindert, dass bei punktuellen Erwärmung wie bei Handlötungen, die Wärme über die an die Lötflächen elektrisch angeschlossene Kupferfläche, z. B. eine Masselage, abfließt und der Lötprozess dadurch beeinträchtigt wird. Bei großflächiger Erwärmung der Leiterplatte, beispielsweise bei maschinellen Lötungen wie dem Reflow-Löten wo die komplette Leiterplatte einheitlich aufgewärmt wird, sind keine Thermal Pads notwendig, stören aber im Regelfall nicht. Bei gesteigerten Anforderungen an geringe Wärmeübergangswiderstände im Bereich der Leistungselektronik, wo unter anderem auch über die Lötstellen in die metallischen Flächen Verlustwärme abgeführt wird, sind Thermal Pads hinderlich. Weitere Beispiele wo Wärmefallen störend wirken sind thermisch optimierte Leiterplatten mit einem Kern aus Aluminium, wie sie beispielsweise als Träger bei LED-Scheinwerfern eingesetzt werden.“

Die Platine ist mindestens eine vier Lagen Platine. Das sieht man daran, dass manche Durchkontaktierungen auf den Außenlagen keine weiterführende Leiterbahnen haben. Diese sind dann wohl in Innenlagen verlegt. Wenn man

mit dem Mikroskop von der Seite auf die Platine sieht, kann man die Lagenzahl genau nachzählen. Unter Hochdruck und viel Hitze werden die Platinen in der Herstellung zusammengepresst. Die einzelnen Lagen kann man noch erkennen. Was noch ein unscheinbares Detail ist: die kleinen goldenen Punkte in den Ecken der Platine (zwei auf der obersten Lage und einer auf der untersten Lage). Das sind Fiducial bzw. auf deutsch Passermaken. Diese helfen dem automatischen SMD-Bestückungsautomaten die Position der Platine genau auszumessen. Dadurch können sehr genau die Bauteile platziert werden. Wesentlich genauer, als wenn die Platinenkante als Basis verwendet werden würde.

Noch zu sehen ist auf der Platine, dass sich unter der einen Hälfte der Netzwerkbuchse und ebenfalls der einen Hälfte des Übertragers in Richtung Netzwerkbuchse keine Massefläche oben unten bzw. in Zwischenlagen befindet. Dies sieht man, wenn die Platinen gegen eine Lichtquelle gehalten wird. So fängt sich die Schaltung an den kritischen Leitungen keine Störungen durch Nachbarleitungen ein. Der Adapter kann ja eigentlich 100MBit/s Übertragungsraten erreichen. Bei diesen Geschwindigkeiten beginnen so langsam die ganzen Tricks der HF-Technik.

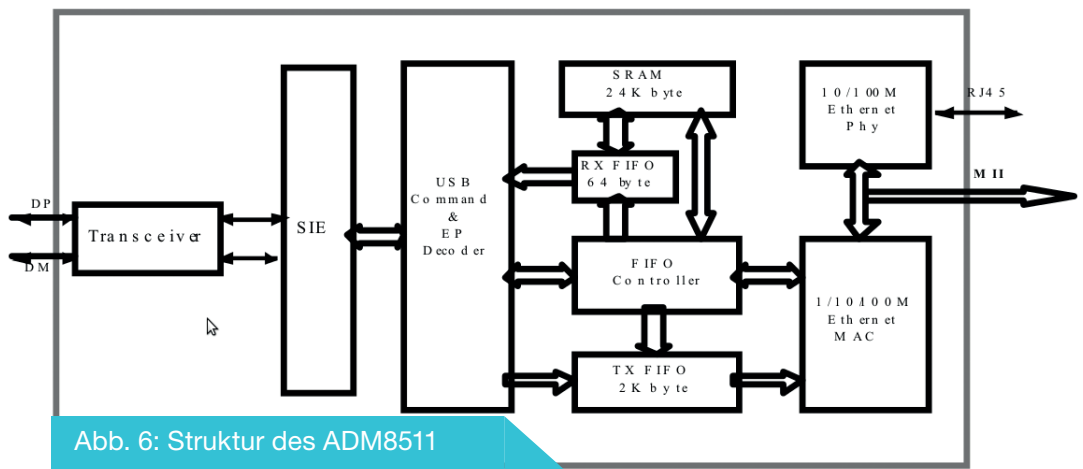


Abb. 6: Struktur des ADM8511

Zusammenfassung

Im Wesentlichen bin ich mit dem Ergebnis dieser Recherche zufrieden. Es konnte einiges an Wissen aufgefrischt werden.

Was noch offen geblieben ist, ist der Bestückungsplatz für das IC auf der Rückseite. Das soll wohl das Geheimnis des

Herstellers der Platine bleiben?

Literatur / Links

[1] http://de.wikipedia.org/wiki/Media_Independent_Interface

[2] Datenblatt ADM8511 (Abbildung 6)

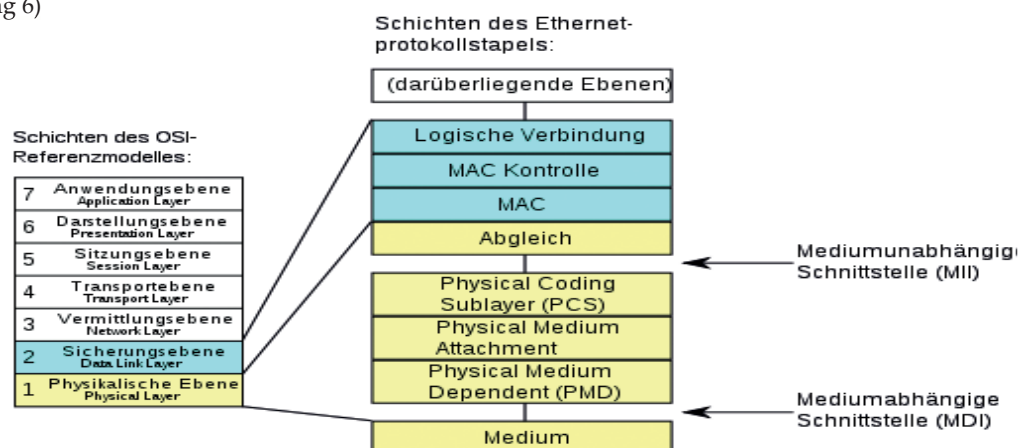


Abb. 7: Auszug aus Wikipedia

Espressomaker

Leckerer Kaffee

Knut Jacobs

Einleitung

Durch Zufall hatte ich vor einiger Zeit einige MEDION Padmaschinen bekommen. Bei ersten Tests stellte ich fest, dass der Espresso nicht so toll ist wie im Vergleich zur Philips Senseo, also beschloss ich dieses zu ändern, da ich gerade eh

kein Projekt hatte. Bei der Suche im Netz fand ich nur eine umgebaute Senseo [1] also musste doch etwas Eigenes gemacht werden (siehe Abbildung 1).

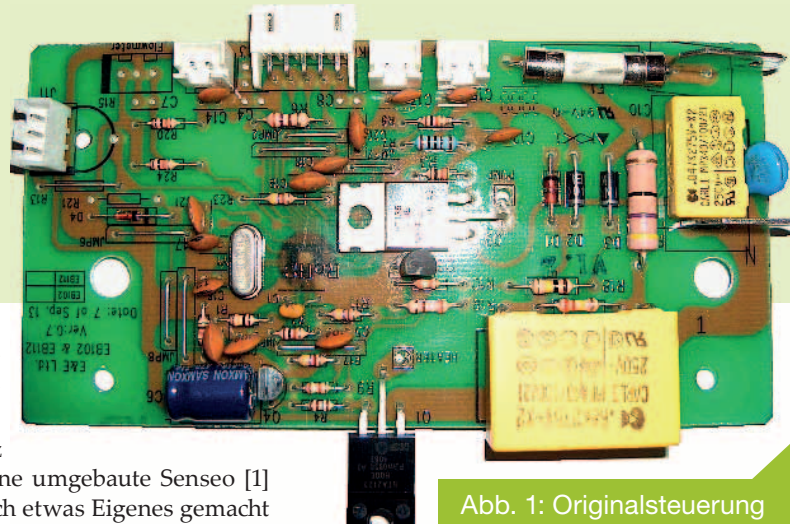


Abb. 1: Originalsteuerung

Bedienpanel, Sensoren und Leistungsteil

Die Technik ist erstaunlich einfach, es gibt einen Reed-Schalter [2] für den Wasserstand, zwei in Reihe geschaltete Schliesserkontakte für Brüheinheit und Verschluss, als Letztes ist noch ein Temperatursensor im Boiler, den ich leider nicht ausbauen konnte (Werte siehe Abbildung 4).

elementen und einen Übertemperatur-Sicherheitsschalter; leider ist dieser so verbaut, dass ich nicht schauen konnte, wann dieser auslöst. Beim Testen hat dieser bei 120° noch nicht ausgelöst (Abbildung 2).

Das Bedienpanel besteht aus drei Tastern und zwei LEDs für die Betriebsanzeige (Abbildung 3).

Der Leistungsteil besteht aus einer Pumpe [4], die über einen Triac auf der Steuerung geschaltet wird, der Boiler wird ebenfalls per Triac geschaltet und besitzt eine Little-Fuse (10A/216°) vor den Heiz-

Die Bauteile sind bis auf Boiler und Steuerung die gleichen wie in dem ersten Senseo Modell.

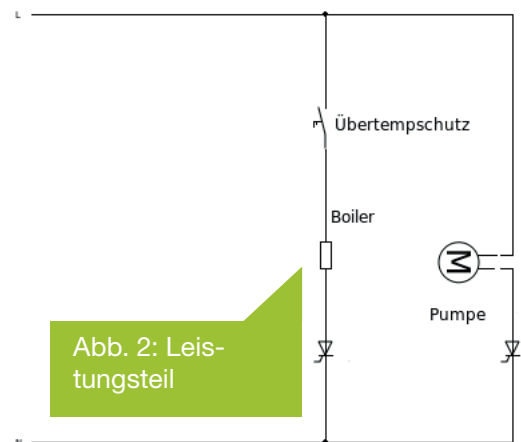


Abb. 2: Leistungsteil

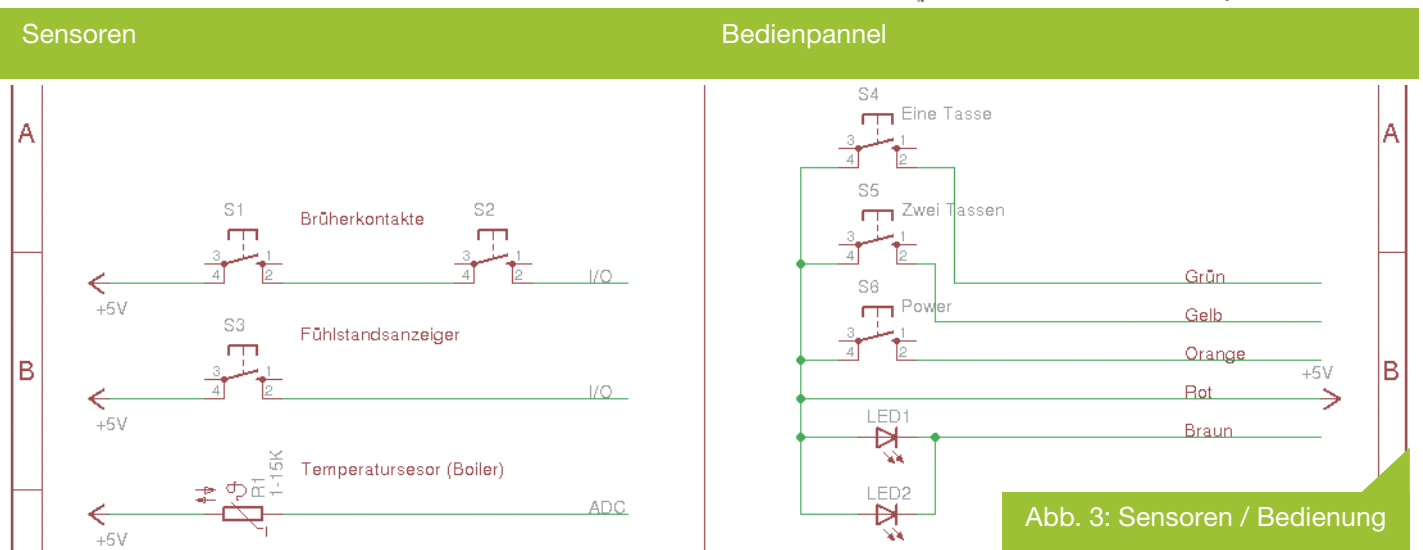


Abb. 3: Sensoren / Bedienung

Prototyp und Entwicklung

Als nächstes habe ich eine der Maschinen zerlegt um einen Prototypen zum Testen und Programmieren zu haben (siehe Abbildung 8). Beim Aufbau des Prototypen stellte ich fest, dass die Schläuche nur per Kabelbinder gesichert waren, was eigentlich ausreicht, allerdings bei 2,6 Bar Wasserdruck nicht wirklich Optimal ist; als Lösung habe ich mir Lebensmittel-tauglichen Universalkleber besorgt [1].

Die Ansteuerung läuft wie im Original über Triacs, am Leistungsteil sowie den Sensoren und Bedienelementen habe ich nichts verändert, dieses bleibt wie im Original. Die alten Funktionen wie das Reinigungsprogramm, eine/zwei Tassen Funktion bleiben soweit nur selbst einstellbar über eine externe Schnittstelle, geplant sind Extramodule für RS232, BT und RS485 als Anschluss für diese ist der UART vom PIC nach außen geführt.

Die Konfiguration wird später über ein Interface gemacht - ähnlich wie bei Routen - so dass man entweder per Terminal Programm oder Extrasoftware die Maschine einstellen kann; zudem soll damit die spätere Einbindung in ein Bus-System möglich gemacht werden.

Die Platine ist extra so aufgebaut, dass sie leicht nachzubauen ist; die meisten 230V Leitungen sind auf der nach innen ze-

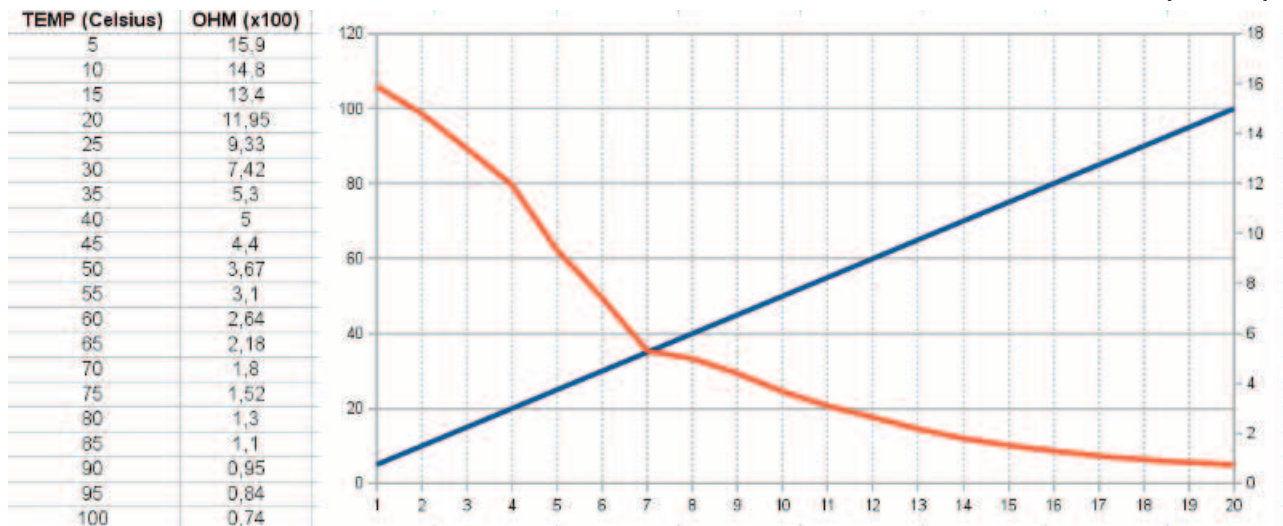


Abb. 4: Temperatursensor

genden Seite (Abbildung 8); der Niederspannungsteil ist auf der TOP Seite der Platine. Als Spannungsregler kommt ein XC62FP502PR [5] 5V LDO (kompatibel zum GV/LV-1117-5) zum Einsatz; bei den Triacs habe ich extra ein größeres Format mit mehr Pinabstand gewählt (TO-232/BTA-26 [6]); zur Ansteuerung kommen MOC3063M [7] mit Nulldurchgangserkennung zum Einsatz.

Um immer einen gleichmäßigen Espresso zu bekommen ist auf der Steuerung ein Uhrenquartz verbaut. Dieser wird als Referenz für das Brühprogramm benutzt und außerdem für den Sleepmode des Controllers.

Das Auswerten des Temperatursensors erfolgt über einen Spannungsteiler mit $2 \times 7,5\text{K}\Omega$.

Zusätzlich habe ich darauf geachtet, dass die Platine so dimensioniert ist wie die alte Steuerung, so dass die Halterungen und der Kühlkörper weiter verwendet

werden können. Im eingebauten Zustand soll man nicht sehen können, dass die Maschine umgebaut wurde.

Die LEDs auf der Platine dienen zum Debuggen und werden für den eigentlichen Betrieb nicht benötigt.

Als μC wird ein PIC16F876A [8] verwendet, allerdings ist dieser abgekündigt. Als Ersatz kann ein PIC16F886 verwendet werden, der auch Softwarekompatibel ist. Die Programmierung erfolgt über ICSP oder Bootlader (siehe Abbildung 5).

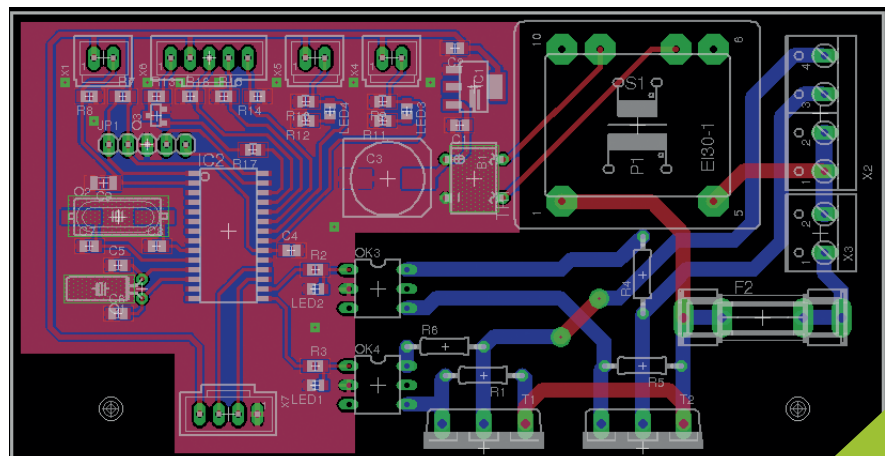


Abb. 5: Layout

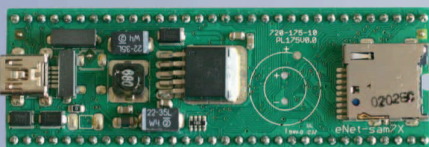
Anzeige



eNet-sam7X

Embedded Ethernet Mikrocontroller Modul für den industriellen Einsatz

ARM7, USB 2.0, 100 MBit Ethernet, Micro-SD, RTC, 4 MB Dataflash, Crypto Engine



thermotemp

- Embedded Linux
- Kernel Portierungen
- Board Support Packages
- RTOS und Bootloader
- Consulting
- Elektronik Entwicklung
- Mikrocontroller Module



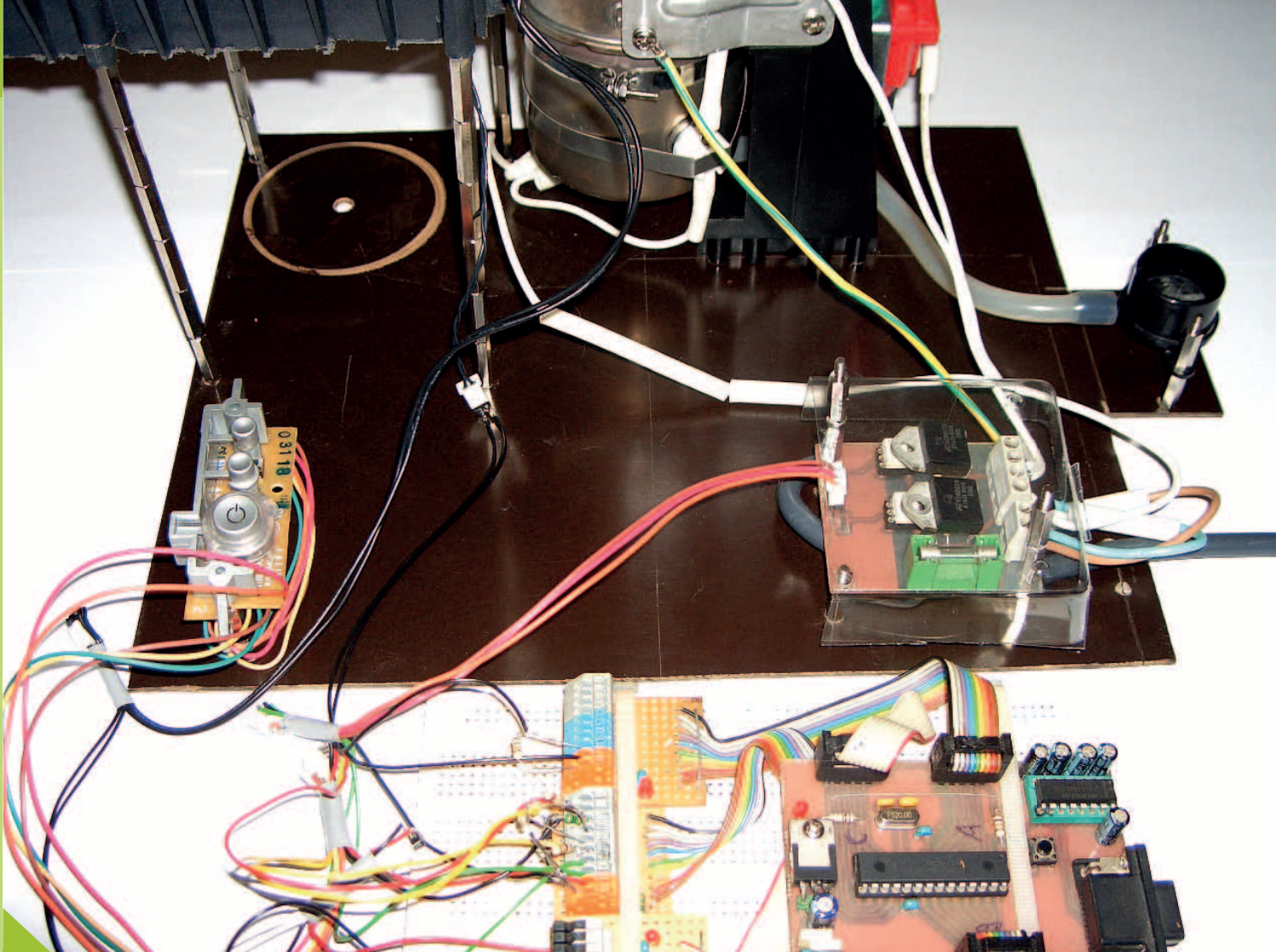


Abb. 6: Prototyp

Bootlader / ICSP

Damit die Steuerung über die Erweiterungen programmierbar ist kommt der Tiny Bootlader [9] zum Einsatz. Durch seine kleine Größe (100 Programmwörter) ist er dafür am besten geeignet. Er befindet sich nach dem Programmieren am Ende des Flash-Speichers. Am Anfang des Programms stehen drei Programmwörter für die Einsprung-Service-Routine des Bootladers, das Protokoll ist RS232, somit ist das Flashen über RS485, Bluetooth, Datenfunk und andere UART-fähigen Geräte problemlos möglich; der Bootlader ist nicht Timing-kritisch.

Auf der Platine ist auch der ICD (ICD2 kompatibel) vorhanden um den Bootlader oder ein Programm einzuspielen, oder um dieses debuggen zu können (Abbildung 7).

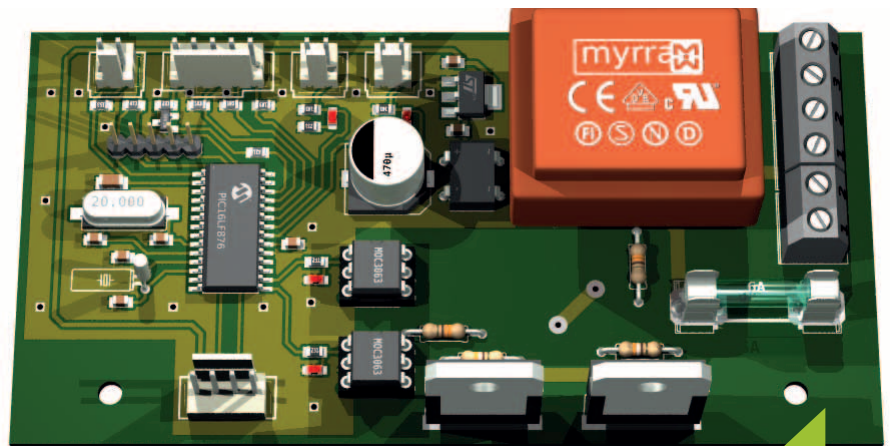


Abb. 7: Expressomaker V1.0

Software

Die Software ist in ASM mit dem Linux Programm Piklab [10] geschrieben. Um den Quellcode übersichtlich zu halten habe ich die Routinen in .inc Dateien je nach Funktionen gepackt.

Interrupt Service Routine (ISP)

Die ISP Routine wertet die Tasten aus (Interrupt on Change (PORTB) und übernimmt die Zeitmessung über den Uhrenquartz (Timer2/TMR1), um eine genaue Referenz für das Brühprogramm zu erhalten.

A/D-Wandler und Sensoren

Über den A/D-Wandler (RA0) wird der Temperatursensor ausgewertet.

Timer

Der Timer2 (TMR1) wird über den Uhrenquartz angesteuert, um eine präzise Referenz für den Wasserdurchfluss zu erhalten.

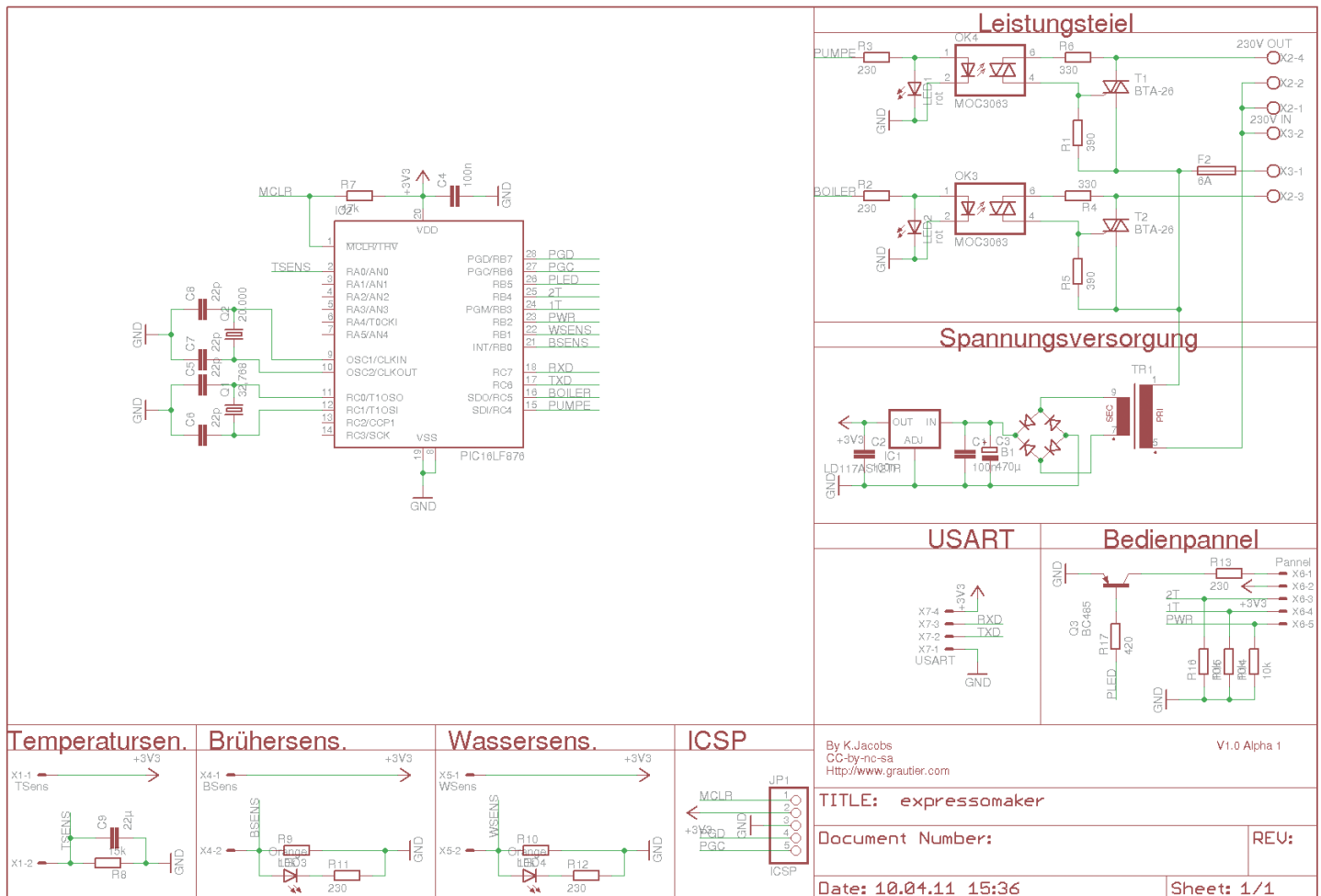


Abb. 8: Schaltung

Sleepmode

Beim Ausschalten werden alle Ausgänge auf Ruhezustand geschaltet und der PIC in den Sleepmode gesetzt. Über den

ToDo's

Der momentane Projektstatus ist Folgender: Die eigentliche Maschine ist fertig, die Software läuft momentan ohne Timer/Interrupts aber sie macht schon mal Kaffee :) Das externe Interface funktioniert noch nicht und ist auch noch

Literatur

- [1] Senseo Steuerung: <http://mosfetkiller.de/?s=kaffeecontroller>
- [2] Meder REED-Sensor: http://www.meder.com/sensoen_mk04000.html
- [3] Lebensmittel Kleber: <http://www.ottozeus.de/produkte/data/produktinformation-1541.pdf>
- [4] Ulka Vibrations Pumpe: http://www.ulka-ceme.co.uk/Ulka_HF_Models.html
- [5] XC62FP330-PR: <http://www.torex-europe.com/products/range/115>
- [6] BTA-26: <http://www.st.com/stonline/books/pdf/docs/7470.pdf>
- [7] MOC3063M: <http://www.fairchildsemi.com/pf/MO/MOC3063-M.html>
- [8] PIC16F876A / PIC16F886A: <http://www.microchip.com/wwwproducts/devices.aspx?ddocname=en010239>
- [9] Tiny Bootloader: <http://www.etc.ugal.ro/cchiculita/software/picbootloader.htm>
- [10] Piklab IDE: <http://piklab.sourceforge.net/download.php>
- [11] Projektseite: <http://www.grautier.com/grautier/>

Uhrenquartz wird der Timer2 (TMR1) getaktet, der bei Überlauf schaut, ob der Einschalter betätigt wurde.

nicht aufgebaut. Auf der Projektseite [11] findet Ihr nach der Veröffentlichung diese Artikels den Quellcode und die Eagle Dateien.

Evaluation Eddy

embedded GNU/Linux

Rother Christian

Eddy CPU

Die SystemBase Eddy v2.5 Module sind kleine programmierbare Embedded Device Server basierend auf dem Eddy CPU v2.5 Embedded CPU Modul mit einem leistungsfähigen ARM9 Prozessor, dem 32-Bit Atmel AT91SAM9G20 (400MHz). Als Speicher dient ein 32Mb SDRAM und ein 4MB-8MB Flash Memory. Auf dem Modul läuft ein GNU/Linux Kernel 2.6.21. Zudem verfügen diese Boards über zahlreiche serielle Hochgeschwindigkeitschnittstellen (RS232, RS422 und RS485 bis 921.6Kb/s), Ethernet

Eddy DK V2.1 (M4)

Mit dem Eddy Development Kit v2.1, können Entwickler einfach die verschiedenen Eddy v2.1 Module testen und evaluieren. Das Baseboard ist unter anderem mit Leuchtdioden für die verschiedenen Betriebszustände (Power, Ready, GPIO, serielle Tx/Rx Leitungen) des Boards ausgestattet. Auch ein auf Eclipse aufbauendes Software Development Kit, steht zur Verfügung, um die Eddy v2.1 Boards kundenspezifisch anpassen zu können.

Softwarequelle

Erste Anlaufstelle ist die Seite <http://embeddedmodule.com>, dort findet man Anleitungen, Dokumentationen, Software Development Kits, allerlei Sources und die aktuelle Firmware.

Um sich alle Dateien herunterladen zu können, muss man sich jedoch anmelden. Danach aber steht einem nichts mehr im Wege.

Als kleine Anmerkung sei erwähnt, dass ich vergebens auf eine Bestätigungsmail gewartet habe. Nach ein paar Wochen habe ich dann einfach versucht mich einzuloggen, was prompt funktionierte.

Beim anschließen des DK s fiel beim ersten Zugriff über picocom auf, dass die Baudrate nicht wie angegeben auf 9600 sondern auf 115200 eingestellt ist. Der richtige Befehl lautet also:

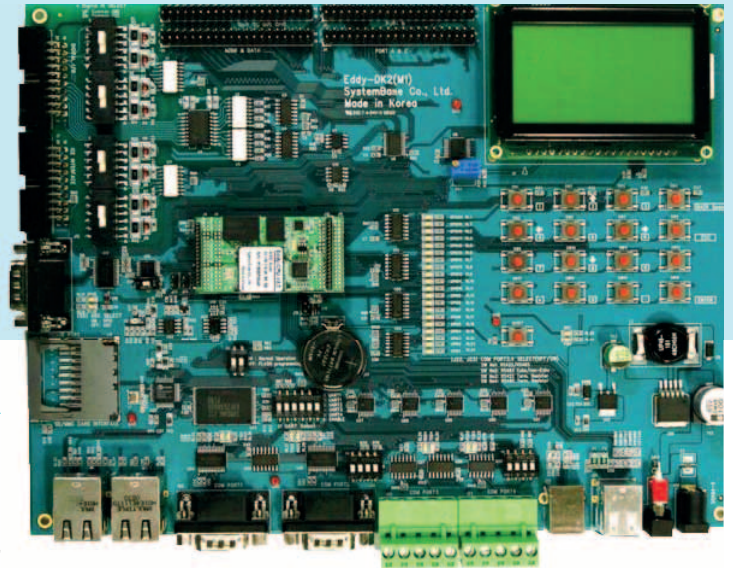
```
#picocom /dev/ttyUSB0 -b 115200
```

Danach können allerlei Eddy-Module-Informationen abgelesen werden, wie z.B die derzeitige IP. Diese benötigen wir auch für unser nächstes Ziel, dem Zugriff über die Ethernet-Schnittstelle. Natürlich ist die Default IP auch in dem Eddy DK Programmers Guide notiert, oder man liest sie einfach von dem LCD Display ab. Wie man es auch macht, als nächsten Schritt verbinden wir das DK-Board per Ethernet-Kabel mit dem PC.

Dank Auto MDI/MDIX kann dies sowohl per Direkt-Link als auch über ein Cross Link LAN-Kabel geschehen.

Bei mir wurde die Verbindung über einen Router geleitet.

Wie im Eddy User Guide Beschrieben wird die IP-Adresse des



und auch drahtlose Schnittstellen wie Bluetooth und WIFI können nachgerüstet werden.

Abb. 1:
eNet-sam7X

Etwas verwirrend war jedoch die Erkenntnis ein Eddy-CPU v2.5 Modul auf einem Eddy DK V2.1 Development Kit erhalten zu haben. Und nicht, wie erwartet, ein Eddy CPU-Modul v2.5 auf einem Eddy DK V2.5 oder eben ein Eddy-CPU v2.1 auf einem Eddy DK v2.1. Dies erklärt sich jedoch damit, dass das Eddy v2.5 Modul voll kompatibel mit dem v2.1 Development Kit ist.

PC auf 192.168.0.222 die Subnet mask auf 255.255.255.0 und der Default Gateway auf 192.168.0.1 gesetzt. Dies geschieht über das grafische Menü oder über die Konsole:

```
#sudo ifconfig eth0 192.168.0.222
#sudo ifconfig eth0 broadcast 255.255.255.0
#sudo route add default gw 192.168.0.1
```

Nachdem die Verbindung aufgebaut war, konnte ich mich per telnet auf dem CPU-Module anmelden:

```
#telnet 192.168.0.223
```

Die Standard Kennung ist wie im Eddy DK Programmers Guide notiert; Username: Eddy und Passwort:99999999.

Bald stellte ich jedoch fest, dass der Zugriff auf das Web Interface nicht möglich war. Die Eingabe der Adresse 192.168.0.223 in der Browserleiste führte leider nicht zum erwünschten Interface, sondern nach umleitung auf die Datei „/cgi-bin/getagent.cgi?type=s“ zur Ausgabe von Programmcode.

Beim Test auf einem Windows Betriebssystem bestätigte sich der Verdacht, dass dies am Zusammenspiel mit dem verwendeten Firefox Browser lag. Das Web Interface funktioniert tadellos auf dem Internet Explorer.

Eine Konfiguration ist aber auch über telnet ohne Weiteres möglich.

Schreiben einer Anwendung

Dank dem Eddy Programmers Guide ist eine Hallo Welt Anwendung zu schreiben kein Problem. Dazu muss erst einmal die

Entwicklungsumgebung aufgebaut werden:

Development Environment

Als erstes laden wir uns die aktuelle Entwicklungsumgebung von <http://embeddedmodule.com> herunter. Unter Downloads->Eddy V2.5 findet sich der Eintrag 12 mit Bezeichnung Source (Bootloader, Linux, Filesystem) – Feb. 15.2011 und unter dem Eintrag 4 findet sich ToolChain for Linux (ver 2.x) die dahinter

stehenden Dateien können nach Registrierung Heruntergeladen werden. Danach sollte das Archiv ‚filesystem_2.5.1.2_11215.tgz‘ und ‚lemonide_linux_10a.tar.gz‘ auf unserer Festplatte ruhen. Wir entpacken diese Archive mit folgender Befehlsfolge:

Die ToolChain

```
#sudo tar zxvf ~/Downloads/lemonide_linux_10a.tar.gz -C /
```

Das Filesystem

```
#mkdir ~/Development/Eddy/
#tar zxvf ~/Downloads/filesystem_2.5.1.2_11215.tgz -C ~/Development/Eddy/
```

Hallo Welt

Das ging schnell, oder? Auch die Hallo Welt Anwendung ist gleich geschrieben. Wir betreten das Eddy_APPS Verzeichnis mit folgender Befehlskette:

```
#cd ~/Development/Eddy/filesystem_2.5.1.2/src/Eddy_APPS/
```

Hier soll unsere Hallo Welt C Datei entstehen, wir Tippen also folgendes in die Kommandozeile:

```
#gedit hello_world.c
```

Die Wahl des Editors ist natürlich Ihnen überlassen ich habe mich wie so oft für den gedit entscheiden. Folgendes wird also in unseren favorisierten Editor getippt:

```
#include <stdio.h>
int main( void )
{
    while (1){
        printf(„hello world!!!\n“);
        sleep(1);
    }
    return 0;
}
```

Das Makefile im selben Ordner muss auch Angepasst werden. Wir ergänzen unser hello_world bei den Targets und fügen den Abschnitt hello_world dem Makefile hinzu.

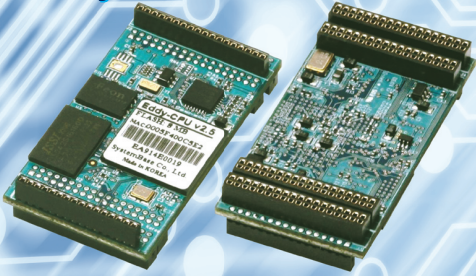
```
#gedit Makefile
.....
test_udp_client \
                testdk                hello_world #added
```

SystemBase Eddy v2.5 series

Linux-based MCU modules
with ARM9 core for serial
communications and
networking applications

- free development environment (Lin/Win)
- free communication library
- free reference projects
- free test applications

Eddy-CPU v2.5



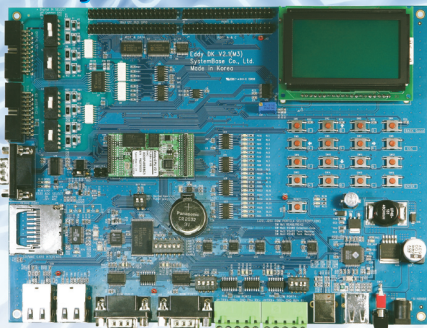
- 32-bit ARM9 @ 400 MHz
- 8 MB Flash ROM + 32 MB RAM
- Embedded Linux 2.6
- Ethernet 10/100 (PHY)
- RS-232 / RS-422 / RS-485

Eddy-MP v2.5



- Low-cost
- Mini PCI module
- Compatible with Eddy-CPU v2.5

Eddy-CPU DK v2.5



- Development kit for Eddy-CPU v2.5



Read more at
www.trenz-electronic.de/eddy25

Anzeige

```
LIBS          = -lrt SB_APIs/SB_APIs.a

all : $(TARGET)

#added

hello_world: hello_world.o

        rm -f $@

        $(CC) $(CFLAGS) $(LDFLAGS) $(IFLAGS) -o $@ $@.o $(LIBS)

        $(STRIP) $@

#/added
```

Anschließend können wir mit einem `make hello_world` den Kompilervorgang anwerfen.

```
#make hello_world
```

Um die crosscompilierte Datei ausführen zu können müssen wir sie zunächst auf das Modul schieben. Dies erledigt ftp nach Eingabe folgender Befehle:

```
#ftp 192.168.0.223
```

Die Standardkennung ist wieder; Username: Eddy, Passwort: 99999999.

```
ftp> bin
ftp>put hello_world
ftp>bye
```

Um unsere Hallo Welt Anwendung nun ausführen zu lassen, rufen wir wieder telnet mit der Standardkennung auf:

```
#telnet 192.168.0.223
```

Da wir uns nach dem Einloggen bereits im /tmp Verzeichnis befinden, können wir nach Rechtevergabe, die Datei ausführen.

```
#chmod +x hello_world
# ./hello_world
```

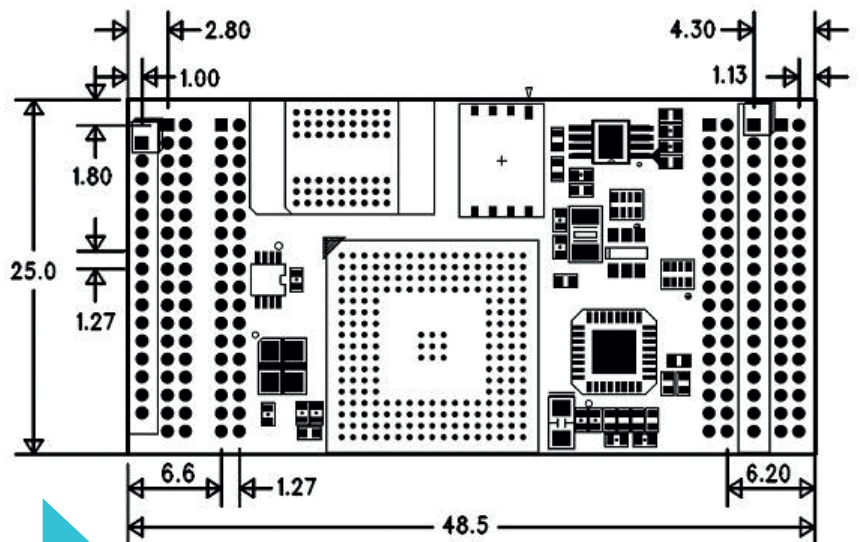


Abb. 2: Abmessung CPU-Modul

(Unit : mm)

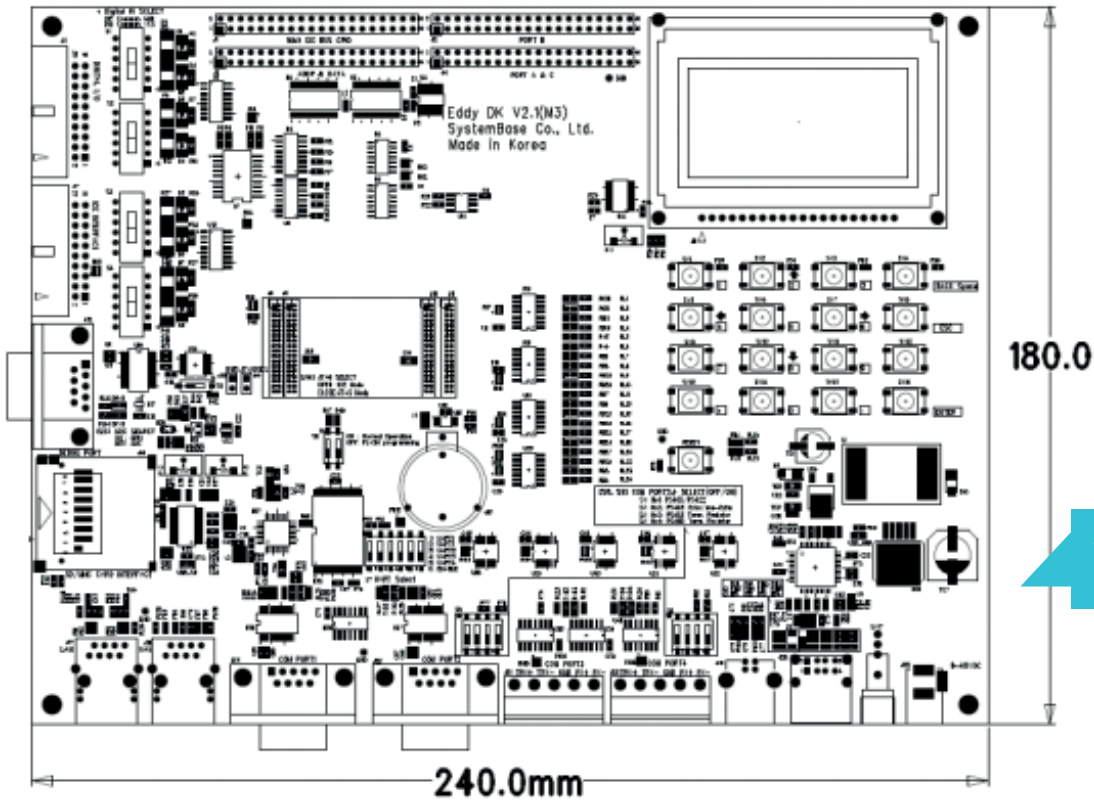


Abb. 3: Abmessung Basisboard

Stellenausschreibungen



Auf Jobsuche?

Dann besuchen Sie unseren Online-Stellenmarkt

journal.embedded-projects.net/stellenmarkt

CMake

Vorstellung des Buildsystems

Bernhard Walle

Einleitung

Zum Übersetzen von C- und C++-Programmen wird üblicherweise auf ein Buildsystem zurückgegriffen, sei es auf der Kommandozeile oder im Rahmen einer integrierten Entwicklungsumgebung (IDE). Traditionell kommt in der Unix-Welt `make` zum Einsatz. Das Entwickeln portabler Programme erfordert oftmals eine ganze Reihe von Tests im Vorfeld, um die Buildumgebung zu erkunden. Dies kann beispielsweise das Finden von Bibliothekspfaden sein oder das Feststellen der Endianess des Zielsystems. Hierfür hat sich in der Unix-Welt das `autoconf`-System durchgesetzt, bekannt durch die allgegenwärtigen `configure`-Skripten. Auch im Embedded-Bereich nimmt die Komplexität und Modularisierung der Software immer mehr zu. Die Unterschie-

de zwischen klassischer PC-Software und Embedded-Entwicklung verschwimmen: Zum einen haben viele eingebettete Systeme bereits die Leistungsfähigkeit von älteren PCs, zum anderen erobert mit Linux ein aus der PC-Welt stammendes Betriebssystem den Embedded-Markt und bringt seine Eigenschaften mit. Nicht zuletzt soll es möglich sein, ein Programm, das eigentlich für ein Gerät entwickelt wird, möglichst umfassend vorher auf dem PC testen zu können.

Wünschenswert ist somit ein Buildsystem, das für möglichst viele Systeme vorhanden ist, was oftmals auch Windows mit einschließt. Die oben erwähnte Kombination aus `autoconf`, `automake` und `libtool`—zusammenfassend als `Autotools`

bezeichnet —erfüllt zwar diese Kriterien, gilt aber als schwer zu erlernen und scheitert beim Windows-Support, sofern nicht auf Unix-Umgebungen wie Cygwin oder Mingw zurückgegriffen wird.

CMake kann hier punkten: Durch seine konsistente Syntax in Form einer prozeduralen Programmiersprache ist es schnell erlernbar. Nicht zuletzt kommt es den Nutzern von verschiedenen IDEs entgegen indem es deren Projektformate erstellen kann. Gerade in heterogenen Projektteams entfällt somit das aufwendige und fehlerträchtige Warten von mehreren Buildbeschreibungen in einem Projektrepository. Unterstützung für Crosscompilierung ist beim Embedded-Einsatz obligatorisch.

Beschrieben

Die Datei `CMakeLists.txt` enthält die Regeln, aus welchen Quellcodedateien welche Programme und Bibliotheken gebaut werden sollen und mit welchen Flags der Compiler dafür aufgerufen werden muss. Üblicherweise gibt es in einem Projekt mehrere davon, jeweils in einem Unterverzeichnis. Listing 1 zeigt ein Minimalbeispiel, das aus einem

Hello-World-Programm `hello.c` ein `hello`-Binary erstellt.

Listing 1: Hello, CMake

```
cmake_minimum_required (VERSION 2.6)
project(hello_cmake C)
```

```
add_executable(hello hello.c)
```

Die Syntax der Sprache ist eigentlich selbsterklärend. Anders als C und die meisten anderen Programmiersprachen, die im Unix-Umfeld entstanden sind, unterscheidet CMake nicht zwischen Groß- und Kleinschreibung.

Gebaut

Wie bereits angedeutet ruft CMake den Compiler nicht selbst auf sondern generiert aus einer Sammlung von Regeln die für ein natives Buildsystem benötigten Beschreibungsdateien. Dies können Makefiles für Unix sein oder Solution-Dateien für Microsoft Visual Studio. Das native Buildsystem übernimmt dann die Kompilation.

Abbildung 1 zeigt die grundlegende Funktionsweise schematisch. Eine Liste der unterstützten Systeme kann unter `[cmakedoc]` im Abschnitt `Generators` nachgeschlagen werden.

Um ein Projekt nun zu bauen, reichen folgende Befehle in einer Shell:

```
% mkdir build
% cd build
% cmake ..
% make
```

Neben dem Generator (hier standardmäßig `Unix-Makefiles`) kann das Resultat durch eine Vielzahl von Variablen beeinflusst werden. Welche Variablen zur Verfügung stehen, hängt vom jeweiligen Projekt und den einge-

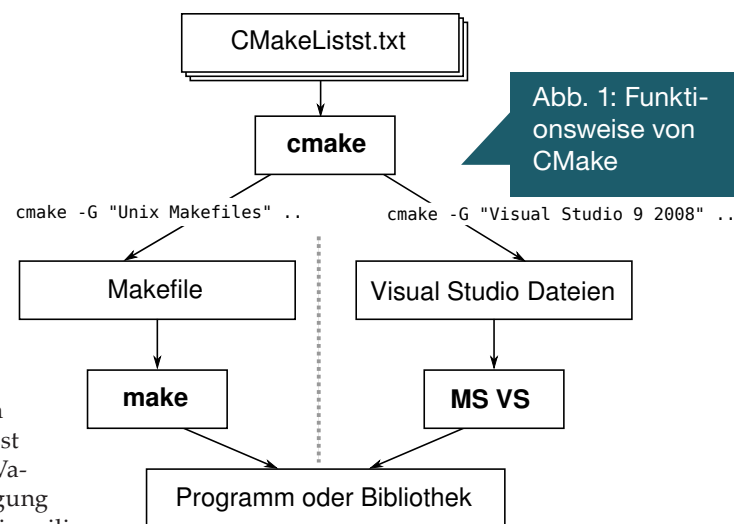


Abb. 1: Funktionsweise von CMake

bundenen Modulen ab. Um das Setzen und Ändern zu vereinfachen, gibt es neben dem Kommandozeilentool `cmake` auch eine interaktive Variante `ccmake` und die Qt-Oberfläche `cmake-gui`.

Während CMake sowohl sog. In-Tree-Builds als auch Out-of-tree-Builds unterstützt (die Begriffe beschreiben, wo die Objektdateien nach dem Bauen liegen, entweder im gleichen oder in einem anderen Verzeichnis), empfiehlt sich aufgrund der Anzahl der generierten Dateien und der einfacheren Handhabung der im Beispiel gezeigte Out-of-tree-Build. Das Buildverzeichnis kann dabei in einem Unterverzeichnis liegen oder an einer anderen Stelle. Das Argument im CMake-Aufruf gibt dabei den absoluten oder relativen Pfad zum Projektverzeichnis an. Abbildung 2 zeigt `ccmake` bei einem etwas komplexeren Projekt. Nach dem Start von `ccmake` startet man mit »c« die Konfiguration.

Die im Bildschirmfoto dargestellten Variablen werden automatisch ermittelt, können aber nachträglich angepasst werden.

Mit einem weiteren »c« bestätigt

man die Konfiguration und generiert mit »g« die Makefiles.

Während `BUILD_GUI` und `BUILD_ONLY_CORE` projektspezifisch sind, existiert der `CMAKE_BUILD_TYPE` immer wenn Unix-Makefiles generiert

werden. Dieser Wert kann auf »Debug«, »Release«, »RelWithDebInfo« oder »MinSizeRel« gesetzt werden. Abhängig davon werden die Kompilate mit oder ohne Debugginginformationen bzw. Optimierungen erzeugt.

Abb. 2: Das CMake-Frontend `ccmake`

Konfiguriert

Größere Projekte verwenden selten nur die Standardbibliothek. Da sich die genauen Compilerflags, die nötig sind, um externe Bibliotheken zu linken, oftmals im Detail unterscheiden, bietet CMake (wie Autoconf) Mechanismen an, diese dynamisch auf dem Zielsystem zu ermitteln. Für eine große Zahl von Bibliotheken werden bereits fertige Rezepte mitgeliefert. Dies gilt auch für Qt, wo neben den obligatorischen Include- und Linkerpfaden auch der Meta-Object-Compiler richtig aufgerufen werden will oder Übersetzungsdateien generiert werden müssen. Ein Blick ins Handbuch gibt hier Auskunft. Heutzutage verwenden viele Bibliotheken das Programm `pkg-config`, das es besonders einfach macht, die Flags zu ermitteln. Im System existiert für jede Bibliothek eine kurze Beschreibungsdatei, welche `pkg-config` dann parst und in

einem standardisierten Format ausgibt. Listing 2 illustriert dies am Beispiel von SQLite3.

Listing 2: SQLite zu einem Programm dazulinken

```
include(FindPkgConfig)

pkg_check_modules(SQLITE3 REQUIRED „sqlite3“)

if (NOT SQLITE3_FOUND)

    message(FATAL_ERROR „sqlite3 C/C++ library could not be found“)

endif (NOT SQLITE3_FOUND)

include_directories(${SQLITE3_INCLUDE_DIRS})
```

```
set(EXTRA_LIBS ${EXTRA_LIBS} ${SQLITE3_LIBRARIES})
```

```
link_directories(${SQLITE3_LIBRARY_DIRS})
```

Die Direktive `include` bindet ein CMake-Modul ein, die Funktion `pkg_check_modules` sucht nach dem `sqlite3`-Modul.

Schließlich fügt `include_directories` den gefundenen Includepfad von SQLite3 dem Compiler-Includepfad (-I) hinzu, analog `link_directories` für den Linkerpfad (-L). Um zu `hello_cmake` nun SQLite3 dazulinken, müsste man schließlich die folgende Zeile ergänzen:

```
target_link_libraries(hello_cmake ${EXTRA_LIBS})
```

Modularisiert

Bei größeren Projekten würde das Buildsystem mit einer einzigen `CMakeLists.txt` schnell unübersichtlich. Außerdem ist es wünschenswert, ein großes Programm in mehrere kleinere Module in Form von statischen Bibliotheken aufzuteilen. Dies ist besonders vorteilhaft beim Übersetzen von Unittests, da dann die Quelldateien nur einmal übersetzt werden.

Ein Beispiel soll dies verdeutlichen:

```
|-- src
| | main.c
|-- md5
| | md5.c
|-- tests
|-- CMakeLists.txt
```


Um nun die imaginäre Bibliothek libmd5.a zu bauen, reicht das in Listing 3 gezeigte Rezept, welche aus md5.c die Bibliothek libmd5.a baut. Da der genaue Name plattformspezifisch ist, kümmert sich CMake automatisch um das Präfix »lib« und um die Endung »a«.

Generiert

Oftmals möchte man die aus dem Versionsverwaltungssystem Version extrahierte Version direkt ins Programm ein-kompilieren um sie in der Hilfe anzeigen zu können. Am schönsten geht dies über eine config.h-Datei, die aus config.h.in erzeugt wird. Letztere ist in Listing 4 dargestellt, Listing 5 zeigt das dafür notwendige CMake-Rezept, welches im Wesentlichen das Programm svnversion aufruft und seine Ausgabe in der Variablen SVNVERSION speichert. Die Variable SVNVERSION_RESULT hingegen enthält den (numerischen) Rückgabewert des Prozesses.

Kreuz-Kompiliert

Um CMake nun in einer Cross-Buildumgebung für Embedded-Linux-Projekte zu nutzen, sind (bei einfachen CMake-Projekten) nur wenige Handgriffe erforderlich. Kurz gesagt konfiguriert man CMake, einen anderen Compiler zu nutzen und nach Bibliotheken in einem anderen Sysroot zu suchen. Auch Projekte, die über pkg-config gefunden werden wollen, kommen problemlos damit klar. Diese Einstellungen legt man in einer Toolchain-Datei ab, wie in Listing 6 gezeigt.

Listing 3: Hello, CMake.

```
set(libmd5_SRC md5.c)

add_library(md5 STATIC ${libmd5_SRC})
```

Im obersten CMakeLists.txt bindet man das Unterverzeichnis nun mit add_

Listing 4: Die config.h.in-Datei, die Vorlage für config.h

```
#ifndef CONFIG_H
#define CONFIG_H

#define PACKAGE_VERSION „@SVN-VERSION@“

#endif /* CONFIG_H */

CMake

5 / 6
```

Listing 5: CMake-Befehle um aus config.h.in die config.h zu erzeugen exe-

subdirectory(md5) ein. Und schließlich teilt man CMake noch mit target_link_libraries (analog zum SQLite3-Beispiel) mit, dass es die md5-Bibliothek dazulinken muss. Möchte man nun eigene Shared-Libraries bauen, so reicht es im Prinzip, das STATIC durch SHARED zu ersetzen.

```
cute_process(COMMAND svnversion
WORKING_DIRECTORY ${PROJECT_SOURCE_DIR}

RESULT_VARIABLE SVNVERSION_RESULT

OUTPUT_VARIABLE SVNVERSION

OUTPUT_STRIP_TRAILING_WHITESPACE)

configure_file(
„${PROJECT_SOURCE_DIR}/config.h.in“
„${PROJECT_BINARY_DIR}/config.h“)
```

Listing 6: Toolchain-File

```
SET(CMAKE_SYSTEM_NAME Linux)
SET(CMAKE_SYSTEM_VERSION 1)

# specify the cross compiler

SET(CMAKE_C_COMPILER arm-v5te-linux-gnueabi-gcc)

SET(CMAKE_CXX_COMPILER arm-v5te-linux-gnueabi-g++)

SET(CMAKE_FIND_ROOT_PATH /home/bwalle/projects/dockstar-staging)

SET(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)

SET(CMAKE_FIND_ROOT_PATH
```

MODE_LIBRARY ONLY)

SET(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)

Nun weist man CMake beim Generieren der Makefiles an, diese Toolchain-Datei zu nutzen:

```
% cmake -DCMAKE_TOOLCHAIN_FILE=path/to/toolchainfile.cmake ..
```

Einige Embedded-Linux-Buildsysteme wie ptxdist (<http://www.ptxdist.org>) unterstützen den Prozess, indem die notwendige Toolchain-Datei automatisch aus den Projekteinstellungen erstellt wird. [cmakecrosscompiling] bietet weitergehende Informationen zum Thema Crosscompiler und CMake.

Und weiter?

Leider konnte der Artikel die Verwendung und die Vorzüge von CMake nur kurz anreißen. Selbstverständlich kann man mit wenigen Zeilen Code ein install-Target generieren. Die Erweiterung CPack ermöglicht es, Installationspakete für Linux (Tarballs, RPMs, DEBs), Windows (NSIS-Installer) oder Mac OS (Disk-Images) zu erstellen. Besonders nützlich ist CTest: Hiermit lassen sich (Unit-)Tests einbinden und mit make test ausführen. Auf Wunsch kann man die Ergebnisse an ein Dashboard

(CDash) senden sich grafisch die damit abgetesteten Codezeilen anzeigen lassen.

Leider hakt es an einigen Stellen an der Dokumentation. Allerdings verbreitet sich CMake in der OpenSource-Welt zunehmend und so gibt es genügend Beispielcode. Wer es ganz genau wissen möchte, sollte das von den Autoren geschriebene Buch [MasteringCMake] lesen.

Literatur

[1] [cmakedoc] CMake 2.8 Documentation

[2] [cmakecrosscompiling] CMake Cross Compiling

[3] [MasteringCMake] Ken Martin, Bill Hoffman: Mastering CMake

USBprog 4.0

Der Open Source Programmer

Benedikt Sauter

Der neue USBprog 4.0 mit integriertem USB RS232 und Pegelwandler für 3,3V

Die Vorgeschichte

Das Projekt USBprog startete zum ersten Mal 2007. Ein kleiner universeller Programmer, welcher es mit Hilfe eines PC-Programms und einem „Online Archiv“ einfach ermöglicht AVR-, ARM-, CPLD-, etc. -Programmer zu erstellen. Dank des Open-Source Konzepts von der Hardware bis hin zur Software wur-

den in der Zwischenzeit viele ermutigt mitzumachen und es sind einige Anwendungen für den USBprog entstanden. Nach knapp vier Jahren wurde jetzt eine neue Version USBprog 4.0 (Abbildung 1) mit den Erfahrungen der letzten Jahre entwickelt.

Die neuen Features

- Integrierte 3V3 Pegelwandler (per Schalter wählbar 3,3V oder 5,0V „Target“)
- Extra integrierter USB-RS232 Schnittstelle (vollständige PC-Belegung)
- Passendes Gehäuse
- Anschlussbelegung
- AVR-Programmierung (6 polig, 10 polig),
- ARM-Debugging (20 polige)
- RS232 (inkl. RTS, DTC, ...)
- 8 optionale IO- oder AD-Kanäle (PORT A des integrierten ATmega32)
- Features siehe auch Abbildung 2

Abb. 1: USBprog Bootloader- und Pegelwandler-schalter

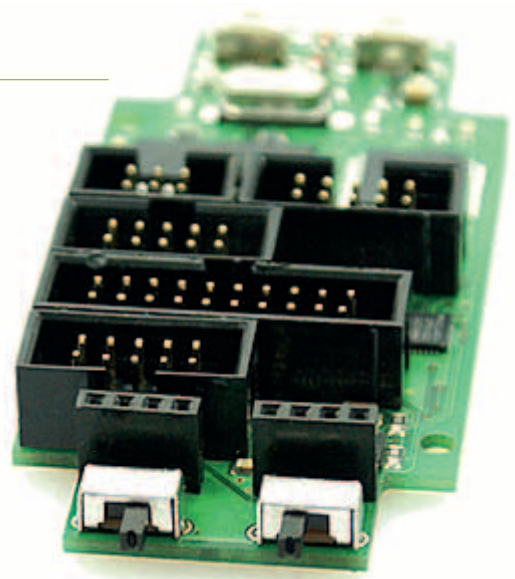
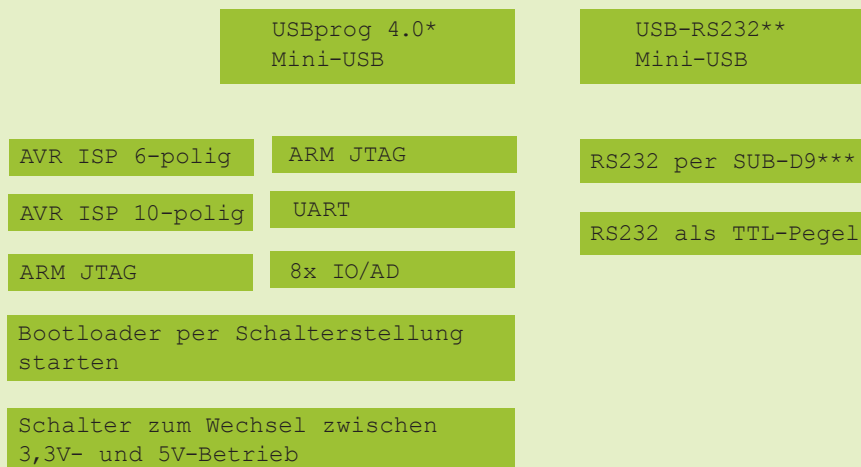


Abb. 2: Übersicht USBprog 4.0



* USB-Schnittstelle des USBprog 4.0

** USB-Schnittstelle des USB-RS232-Wandlers (erscheint als extra COM bzw. /dev/ttyUSBx Schnittstelle im Betriebssystem)

*** Vollständige UART inkl. HW-Handshake etc.

Kurzbeschreibung

Der USBprog bietet mit einer einfachen flexiblen Hardware eine Plattform für ein Universalprogrammiergerät an. Über ein Computerprogramm (Windows oder Linux) kann man bequem aus einem „Online-Pool“ auswählen für welchen Mikroprozessor man das Programmiergerät benötigt – und entsprechend mit einem Klick die interne Firmware austauschen. Aufgrund der modernen USB-Hardware und der freien Schaltung sind in der Zwischenzeit sehr stabile Lösungen für die z.B. AVR, ARM, AVR32, CPLD, etc. Entwicklung entstanden. Weitere Lösungen sind immer noch am entstehen.

Vorhanden:

- AVR ISP Programmer (kompatibel mit AVR-Studio, BAS-COM, avrdude, etc.)
- ARM JTAG Debugger + Programmer (OpenOCD)

- AVR32 Rettungsbootloder für NGW100
- CPLD Programmer (XSUF Player für Xilinx)
- USB zu UART Schnittstelle (aktuell fest auf 38400 Baud)

Aktuell entsteht:

- Logikanalysator (Überarbeitung der ersten Version)
- JTAG Schnittstelle für Xilinx ISE Studio (Helfer gesucht)
- JTAG ICE 2 für AVR Studio 4 und 5 anpassen (Helfer gesucht)

USBprog 4.0 im Einsatz

Mit dem neuen USBprog ergibt sich eine einfachere Handhabung für die aktuellen Programmierfirmwares. Für die Programmierung oder das Debugging von ARM-Prozessoren kann mit der neuen schnellen Firmware: <http://code.google.com/p/usbprog-jtag/> bequem ohne Kabelwirrwarr (siehe Abbildung 3) entwickelt werden. Der Integrierte Pegelwandler funktioniert hier natürlich auch.

Die klassische Anwendung ist der AVR ISP mk2 Klon. Hierfür finden sich ein 10-poliger und 6-poliger ISP Stecker auf dem USBprog (siehe Abbildung 4 und 5). In Abbildung 1 sieht man die Schalter. Der rechte Schalter ist für die Auswahl des Pegels von 5,0 V (nach außen) oder 3,3 V (nach innen). Der linke Schalter startet auf der inneren Seite den Bootloader (Achtung erst nach dem neu Ab- und Anstecken des USBprogs am Computer). In der äußeren Stellung startet die eigentliche Firmware nach dem Anstecken automatisch. Der Schalter beschreibt so das Verhalten beim Anstecken des USBprogs (innen = Bootloader, außen = Firmware).

In Abbildung 7 sieht man ebenfalls eine Neuerung des USBprogs. Mit dem RS232-Kabeladapter verhält sich der USB-RS232 Wandler, welcher über die zweite USB-Buchse mit dem Computer verbunden wird, wie eine alte klassische COM-Schnittstelle am PC. Es werden alle Signale der RS232 Schnittstelle unterstützt (Handshake, etc.). Benötigt man keinen Pegelwandler sondern möchte direkt eine UART-Schnittstelle von einem Mikrocontroller abgreifen, kann man sich direkt auf den 4-poligen UART-Stecker verbinden (siehe Abbildung 10). Der linke 4-polige Stecker ist der UART-Anschluss vom extra integrierten USB-RS232 Wandler. Der rechte kann als UART über den USBprog verwendet werden, wenn die entsprechende Firmware geladen ist. Abbildung 11 zeigt noch zu guter Letzt den neuen USBprog 4.0 inkl. vollständigen Kabelset.

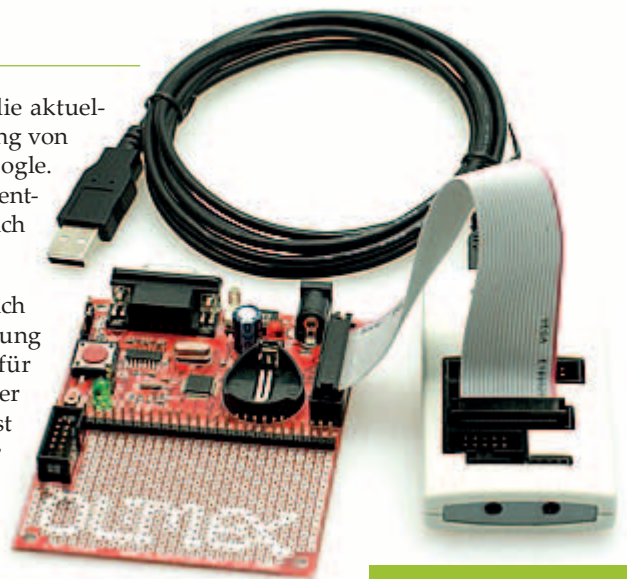


Abb. 3: USBprog ARM JTAG-Anschluss-sam7X



Abb. 5: USBprog AVR 6 poliger ISP-Anschluss

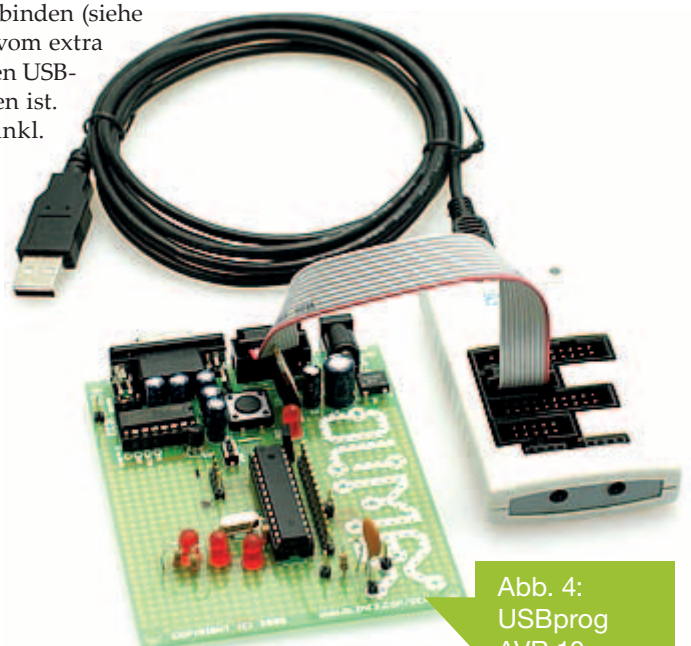


Abb. 4: USBprog AVR 10 poliger ISP-Anschluss

Die Schaltung

Der Kern der Schaltung ist natürlich gleich geblieben, um vollständig kompatibel zu den bestehenden Firmware-Versionen zu sein. Neu hinzu kam der integrierte USB-RS232 Wandler. Hier wurde ein CP2102 von Silabs verwendet. Wie man im Schaltplan sieht, benötigt er kaum extra Bauteile. Zudem ist in dem Baustein ein integrierter 3,3 V Regler verbaut, welcher für die „Pegelwandler“-Funktion verwendet wird. Statt einer Pegelwandlung - wie bisher - wird jetzt die Schaltung entweder mit 5,0V oder 3,3 V betrieben. Ausgewählt wird dies mit dem o.g. Schalter (Abbildung 1).

Der USBN9604 und ATmega32-A können problemlos mit 3,3 V betrieben werden. Für die echte COM-Schnittstelle, welche per USBprog angeboten wird, wird noch ein Treiberbaustein für die Erzeugung der richtigen Pegel benötigt. Ein MAX3243 wurde auf Grund seiner geringen Baugröße ausgewählt.

Auf der Platinen befinden sich jetzt auch:

- 20 poliger JTAG Anschluss für ARM
 - 10 poliger AVR-ISP Stecker
 - 6 poliger AVR ISP Stecker
 - 4 poliger UART Stecker
 - 10 poliger IBM ATX Standard für COM-Schnittstelle
- und 3 Leuchtdioden, die den Zustand des USBprog anzeigen (Abbildung 6)

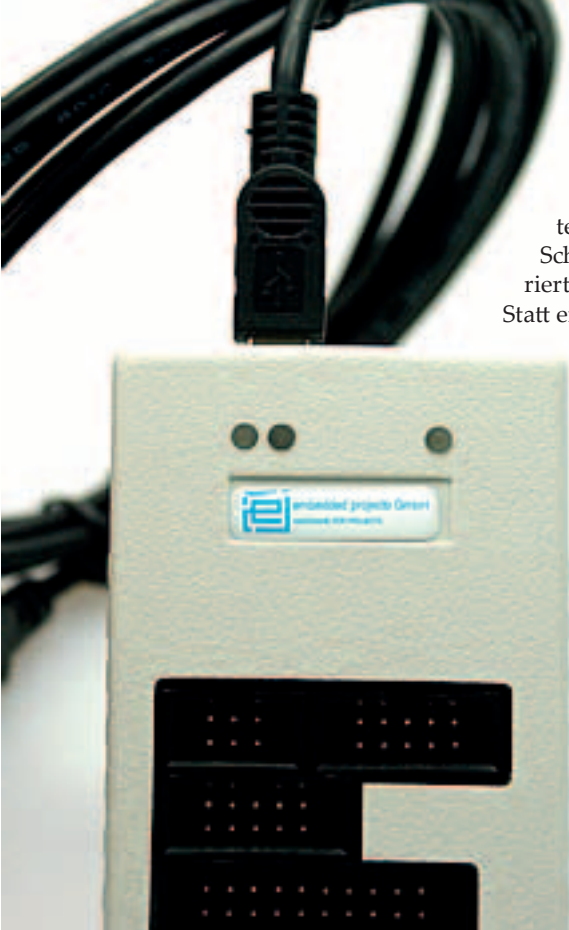


Abb. 6: USBprog Statusleuchtdioden

Live-CD für USBprog Tools

Bernhard Walle hatte eine sehr gute Idee. Eine Live-CD für Windowsbenutzer. Auf der CD ist alles installiert. Um die Firmware auf dem USBprog zu tauschen also einfach CD ins Laufwerk, davon booten. In dem Live-Betriebssystem findet Ihr dann die USBprog-GUI:

Link: <http://prdownload.berlios.de/usbprog/USBprog.i686-0.3.0.iso> (Image-Datei für Brennprogramm)

Bernhard hat außerdem ein sehr gutes Handbuch zum USBprog zusammengestellt:

Link: <http://bwalle.de/programme/usbprog/USBprog.pdf>

Installation unter GNU/Linux

Mittlerweile sind die USBprog Tools seit ein paar Jahren fester Bestandteil von Debian und daher kann man in den meisten Distributionen - die auf Debian basieren - ganz einfach die Tools installieren. Natürlich sind diese ebenfalls für USBprog 4.0 einsetzbar.

```
sudo apt-get install usbprog-gui avdrude
```

Installation unter Windows 7 64 Bit

Die USBprog-GUI kann man unter Windows direkt mit dem USBprog Installer installieren:

<http://prdownload.berlios.de/usbprog/USBprog-0.2.0.exe>

Es gibt mit dem AVR Studio unter Windows 7 bei 64 Bit leider ab und an Probleme. Ein netter USBprog Nutzer schickte mir diese Info zu:

„Das Tool dafür nennt sich Zadig: http://sourceforge.net/apps/mediawiki/libwdi/index.php?title=Main_Page <http://sourceforge.net/projects/libwdi/files/zadig/>

Nachdem man den USBProg angeschlossen hat startet man das Tool (benötigt Administrator Rechte) und kann auswählen, welcher Treiber für den USBprog (der jetzt in der Liste auftauchen sollte, in meinem Fall als AVRISP MK2 Clone) installiert werden soll (ich persönlich habe libusb0 (v.1.2.4.0) verwendet, ich weiß nicht, ob die anderen beiden auch funktionieren). Das geht recht fix und schon läuft der USBProg (alles ohne reboot). „

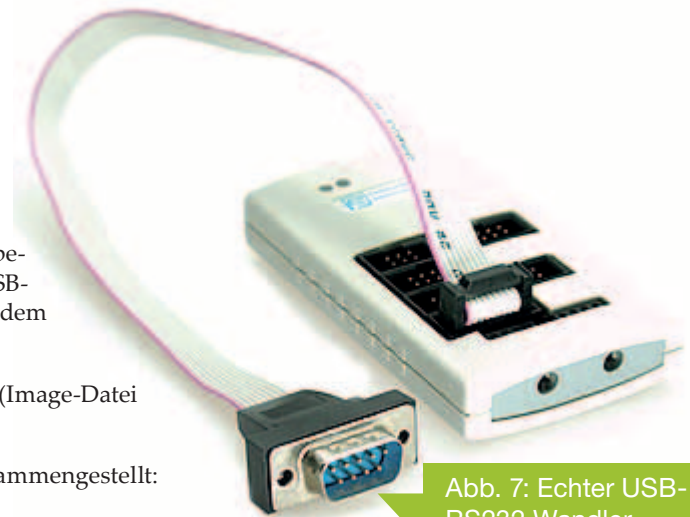
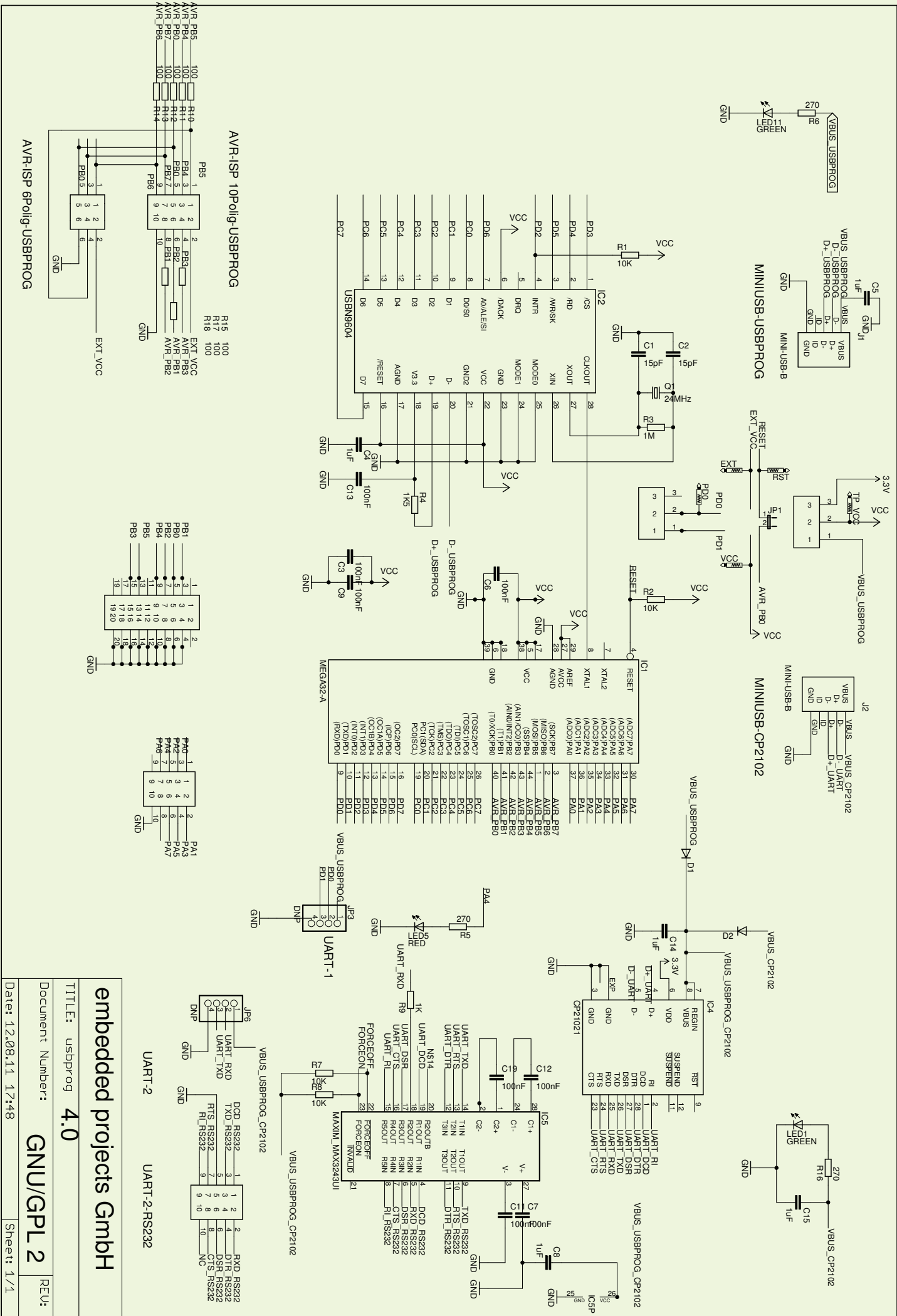


Abb. 7: Echter USB-RS232 Wandler



embedded projects GmbH

TITLE: usbprog 4.0

Document Number: GNU/GPL 2

Date: 12.08.11 17:48

REV: SHEET: 1/1

Abb. 8: Schaltplan

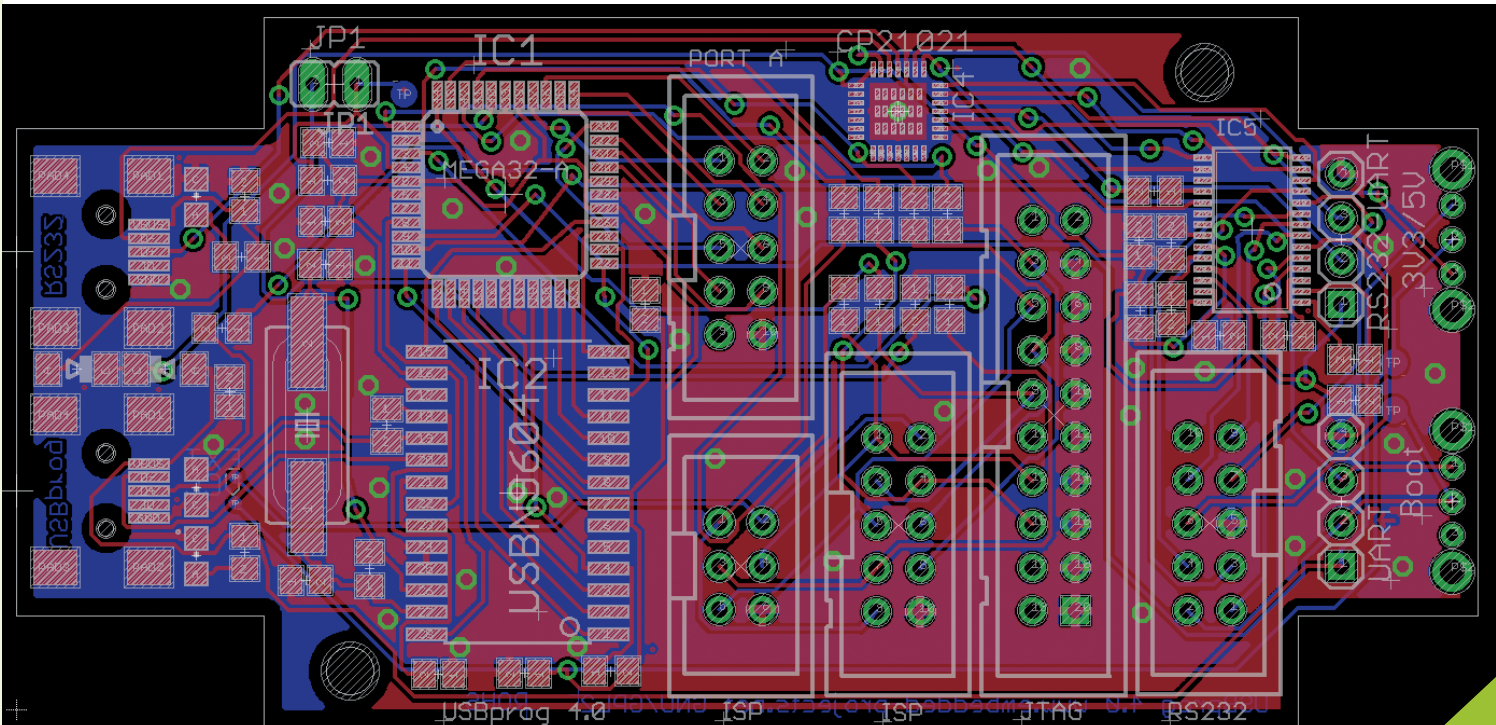
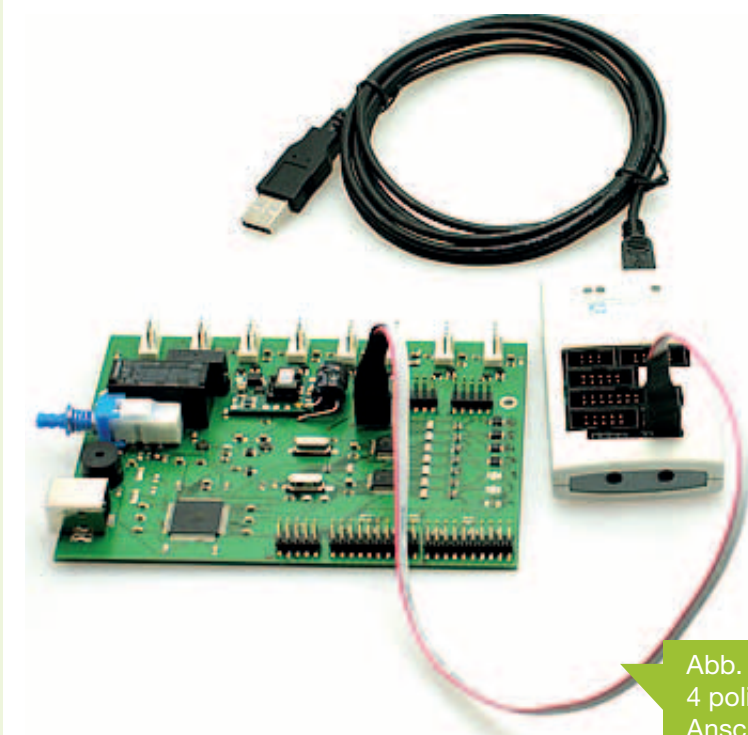


Abb. 9: Platinenlayout

Abb. 10: USBprog
4 poliger UART-
AnschlussAbb. 11: Erweiter-
tes Kabelset für
USBprog 4.0

Update für AVR Studio 5

Die alte Firmware des USBprog für die AVR Programmierung verweigert die Zusammenarbeit mit dem AVR Studio 5. Mittlerweile liegt ein Patch vor. Es muss einfach mit dem USBprog-GUI Tool einmal die Firmware „AVR ISP mk2 Klon“ neu eingespielt werden. Danach klappt der Zugriff von Windows mit dem AVR Studio 5 ohne Probleme.

Der Link zu aktuellen Version lautet:
<http://svn.berlios.de/svnroot/repos/usbprog/trunk/avrismk2klon/main.bin>

Die neue Homepage

Alle neusten Infos, Download, Anleitungen, FAQs, etc. findet man ab sofort unter:

<http://wiki.embedded-projects.net>

Alle alten Seiten werden nach und nach in das Wiki integriert.

Die USBprog Familie

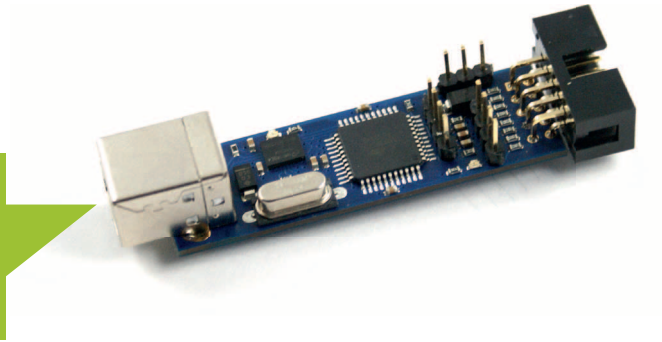


Abb. 13: Der „klassische“ USBprog 3.0. Alle SMD-Bauteile sind vormontiert. Nur die Durchsteckbauteile müssen selbst gelötet werden.



Abb. 14: USBprog 4.0
Endlich im stabilen Gehäuse

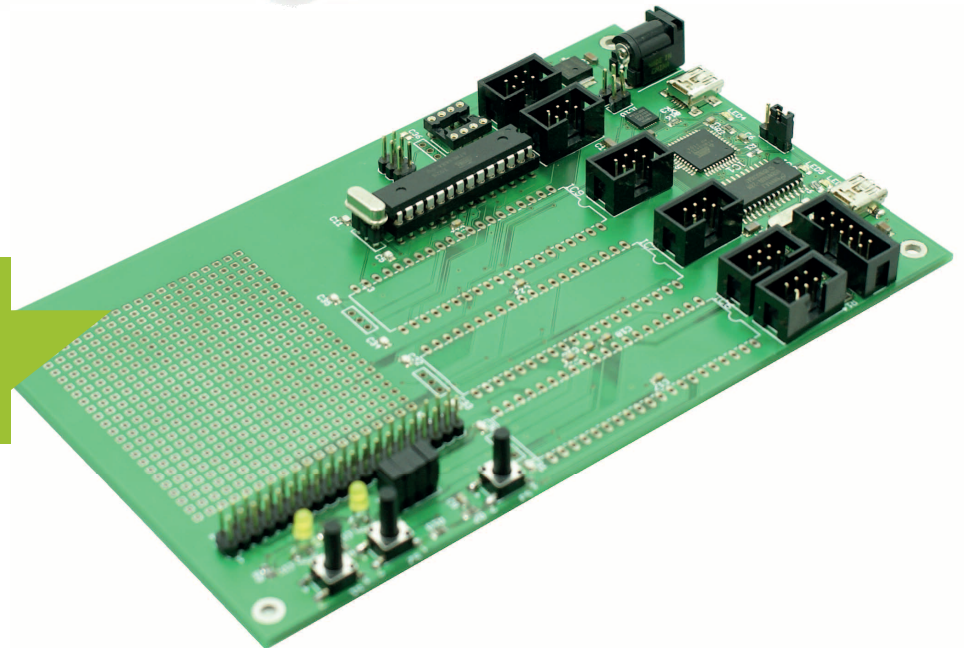


Abb. 15: USBprog AVR STK
AVR Evaluationsboard für
bekannte AVR im DIP Gehäuse.
Erscheinung: November 2011

Links

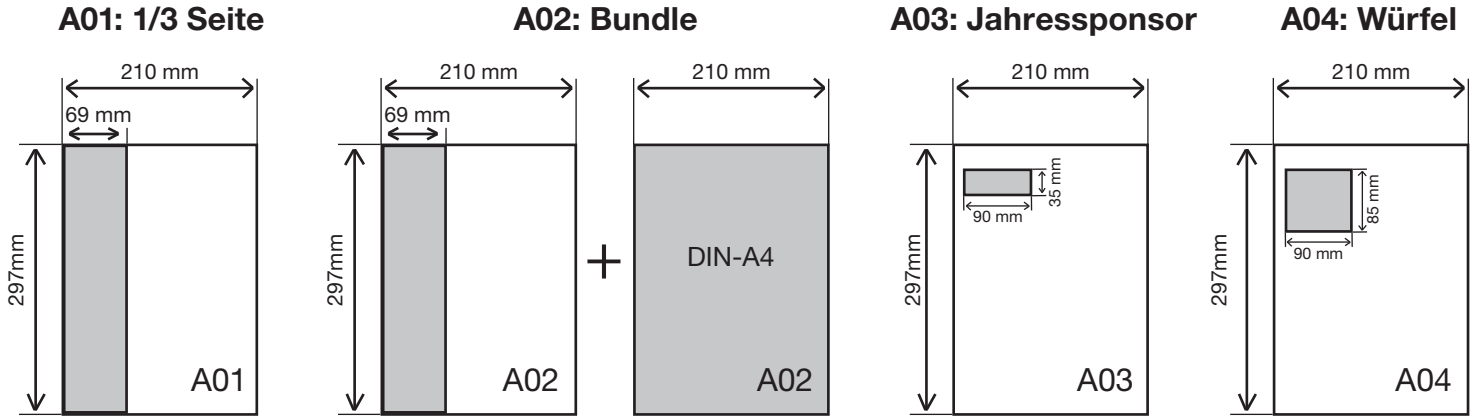
{1} Alle Dokumente zum USBprog: <http://wiki.embedded-projects.net>

{2} Online-Shop für USBprog: <http://shop.embedded-projects.net>

Ihre Anzeige im embedded projects Journal

Ansprechpartner:

embedded projects GmbH
Holzbachstraße 4
D-86152 Augsburg
Tel. +49(0)821 / 279599-0
Fax: +49(0)821 / 279599-20
Mail: journal@embedded-projects.net



embedded projects Journal:

Das embedded projects Journal ist eine Zeitschrift, die unter einer Open-Source Lizenz steht. Alle Artikel und die Zeitschrift können so unter Einhaltung dieser Lizenz weiter verarbeitet werden. Das besondere an dem Open-Source Projekt Journal ist, dass die Zeitschrift nicht nur frei als PDF-Version verfügbar ist, sondern von den Lesern kostenlos abonniert werden kann. Druck und Porto werden durch Anzeigen von Sponsoren finanziert.

Aktuell wird jede Ausgabe ca. 8000 mal online und ca. 2000 mal als Papierversion gelesen.

Anzeigenformate und Preisliste:

Bezeichnung	Bezeichnung	Format in mm Breite x Höhe	Preis*	Erscheinung in
A01	1/3 Seite	69 x 297 mm	150 €	1 Ausgabe
A02	Bundle	69 x 299 mm + 210 x 297 mm	400 €	1 Ausgabe
A03	Jahressponsor **	90 x 35 mm	100 €	4 Ausgaben
A04	Würfel	90 x 85 mm	100 €	1 Ausgabe

* Alle Preise zzgl. Ust.

** Im Angebot begriffen: 1. Frei Haus Lieferung jeder Ausgabe.

2. Ihre Anzeigen wir zusätzlich auf unserer Homepage unter:
<http://www.ep-journal.de> im Firmenverzeichnis veröffentlicht.

Ausgabe	Anzeigenschluss	Erscheinungsmonat
04 / 2011	15.11.2011	Dezember
01 / 2012	15.02.2012	März
02 / 2012	15.05.2012	Juni
03 / 2012	15.08.2012	September
04 / 2012	15.11.2012	Dezember



Bestellfax an +49(0)821 / 279599-20

Anzeige in nächster Ausgabe veröffentlichen

A01 A02 A03 A04

Anzeige veröffentlichen in / ab Ausgabe ___ / ___

Firma: _____

Vor- / Nachname: _____

Anschrift: _____

Tel. / E-Mail: _____





Widerstands-Sortiment
SMD0805, 1%, TK 100, RoHS
62 Werte E12-Reihe
6200 Widerstände

€ 45,-
inkl. 19% MwSt zzgl. Versand


<http://www.FundF.net>

→ firma.embedded-projects.net
DAS HARDWARE FOR YOUR PROJECTS-PORTAL


 In unserem Online-Shop finden Sie eine große Auswahl verschiedenster Mikrocontrollerboards, Programmer, Debugger u.v.m.
→ shop.embedded-projects.net

 Speziell für Studenten und Hochschulen, bieten wir diese Ausbildungsinitiative an. Mikrocontrollerboards für den kleinen Geldbeutel.
→ student.embedded-projects.net

Unser Büro in Augsburg besteht aus leidenschaftlichen Entwicklern. Sprechen Sie uns an, wir finden eine Lösung für Ihr Problem.
→ projekte.embedded-projects.net



Holzbachstraße 4, D-86152 Augsburg
Tel +49 (0) 821 279599-0
Fax +49 (0) 821 279599-20
info@embedded-projects.net



embedded projects GmbH
HARDWARE FOR PROJECTS

Interesse an einer Anzeige?

info@embedded-projects.net

FIND  www.f-y-e.de

your engineer

Der Experten-Wegweiser zu Ihrem Elektronikentwickler

Elektronik- / Softwareentwicklung	Find-Your-Engineer ist ein persönliches Empfehlungsnetzwerk. Firmen die Elektronik-Experten suchen, wenden sich bitte direkt an:
Layout	
Mechatronik	
Bestücker / EMS-Dienstleister	
EMV-Dienstleister	

Mit der besten Empfehlung!

Markus Kessler
kontakt@find-your-engineer.de

MIXED MODE

Software-Entwickler/in gesucht! (München)

C++ C# .NET MDA UML

Embedded Software

Realtime Java



www.mixed-mode.de/jobs



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

Regelmäßig

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 [2] 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://journal.embedded-projects.net>

[2] Online - Shop für Journal:
<http://www.embedded-projects.net>

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 / 2795990 oder sendet eine E-Mail an die oben genannte Adresse.

Impressum

embedded projects GmbH
Holzbachstraße 4
D-86152 Augsburg
Telefon: +49(0)821 / 279599-0
Telefax: +49(0)821 / 279599-20

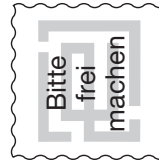
Veröffentlichung: 4x / Jahr
Ausgabenformat: PDF / Print
Auflagen Print: 2500 Stk.
Einzelverkaufspreis: 1 €

Layout / Satz: EP
Titelfoto: Claudia Sauter

Alle Artikel in diesem Journal stehen unter der freien Creative Commons Lizenz. Die Texte dürfen, wie bekannt von Open Source, modifiziert und in die eigene Arbeit mit aufgenommen werden. Die einzige Bedingung ist, dass der neue Text ebenfalls wieder unter der gleichen Lizenz, unter der dieses Heft steht veröffentlicht werden muss und zusätzlich auf den originalen Autor verwiesen werden muss. Ausgenommen Firmen- und Eigenwerbung.

Dies ist ein Open Source Projekt.

Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by/3.0>



embedded projects GmbH
Holzbachstraße 4
D - 86152 Augsburg

Name / Firma

Straße / Hausnummer

PLZ / Ort

Email / Telefon / Fax

- Ich möchte jede zukünftige Ausgabe erhalten
- Wir möchten als Hochschule / Ausbildungsbetrieb jede weitere Ausgabe bekommen. Bitte gewünschte Anzahl der Hefte pro Ausgabe ankreuzen. 5 10
- Ich möchte im embedded projects Journal werben oder eine Stellenanzeige aufgeben. Bitte schicken Sie mir Infomaterial, Preisliste etc. zu.

BESSER GLEICH ONLINE KALKULIEREN.

STARRE UND FLEXIBLE LEITERPLATTEN.



LEITON 
RECHNEN SIE MIT BESTEM SERVICE

Endlich wird's einfach und Sie bleiben flexibel. Den umständlichen und aufwändigen Kalkulationsprozessen machen wir einen dicken Strich durch die Rechnung. Sie kalkulieren Ihre Leiterplatten online – ganz bequem, ganz schnell. **Einzigartig: Die Online-Kalkulation gilt auch für Serien und flexible Leiterplatten.** Das macht uns weltweit so schnell keiner nach. Einmalig ist zudem der **Leiterplatten-Expressdienst** von LeitOn mit unserer Garantie: Wenn wir bei Expressdiensten den vereinbarten Übergabetermin an den Versender nicht einhalten können, schenken wir Ihnen die bestellten Platinen! Sie möchten mehr darüber wissen? Wir bieten persönliche Beratung am Telefon und einen kompetenten Außendienst. Sie können bei LeitOn immer mit dem besten Service rechnen.

www.leiton.de

kontakt@leiton.de

Info-Hotline +49 (0)30 701 73 49 0