www.elektor-magazine.de

.magazine

Januar / Februar 2013 | Nr. 505 / 506

elektor

DOPPEL-AUSGABE: 132 SEITEN – RANDVOLL MIT PROJEKTEN!

Programmierbar: Ringleuchte mit LEDs

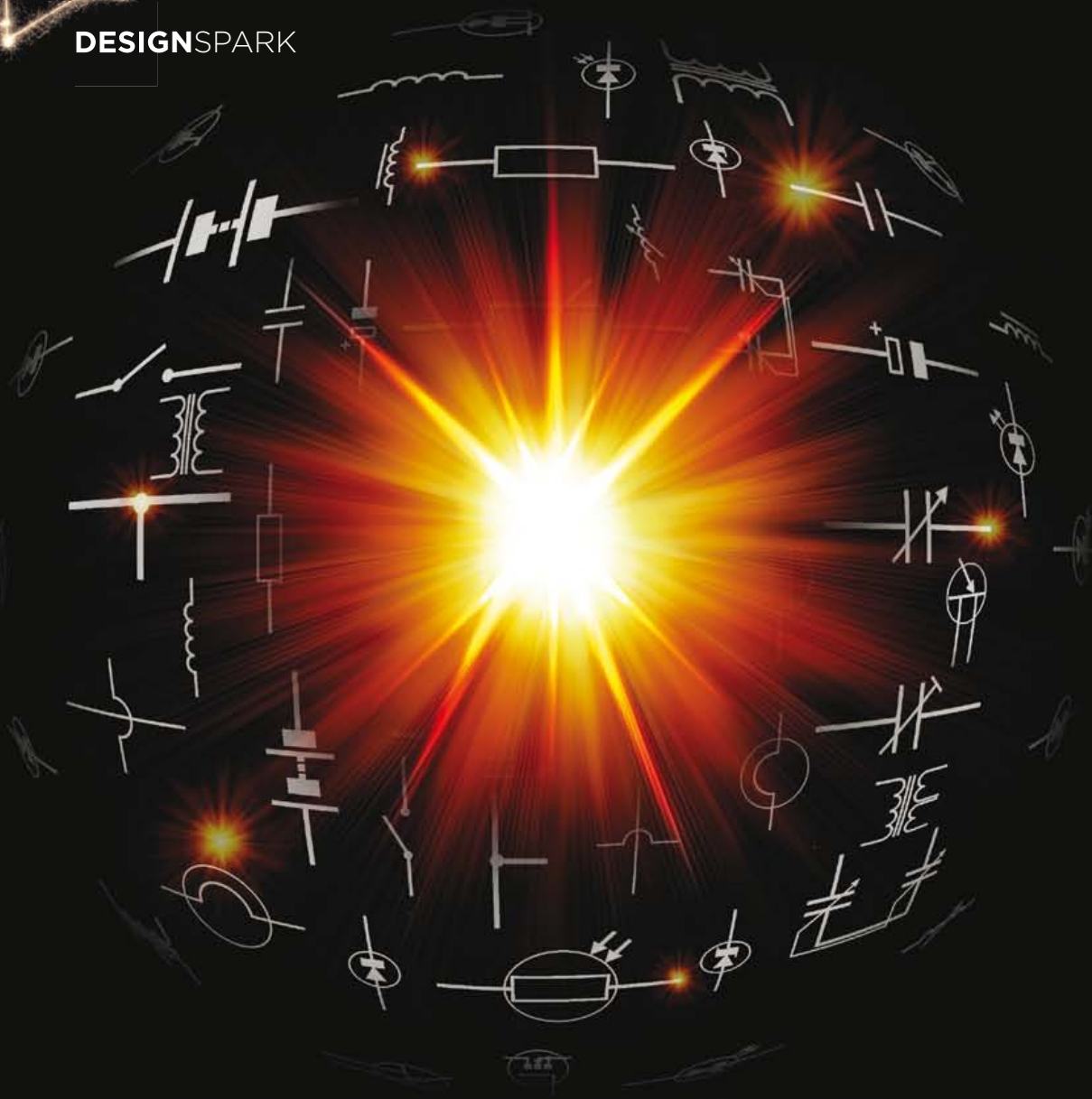
- Embedded Linux: I²C, UART, RS485 | Wecker mit Schaltfunktion
- **FPGA: Die erste Anwendung** | **Kompakter Multi-Tester**
- Flugfunk-Scanner | Kapazitiver Näherungsschalter
- Open Source Hardware
- **Retronik: Tonbandgerät Nagra IV**
- Neues aus dem Labor
- Elektor World: Rund um die Elektronik-Welt

G3078 E

(D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

4 198630 314904 02

DESIGNSPARK



NEW. FREE.
MODELSOURCE

OVER 80,000 FREE SCHEMATIC AND PCB SYMBOLS
IN MORE THAN 20 FORMATS, INCLUDING PADS,
ORCAD, ALTIUM AND CADSTAR.

Discover today at
www.designspark.com

UNIQUE
RESOURCES BY RS

Lesen Sie die neue Elektor ein Jahr lang in der ultimativen GOLD-Mitgliedschaft und profitieren Sie von allen Premium-Vorteilen!

Die Elektor-GOLD-Jahresmitgliedschaft bietet Ihnen folgende Leistungen/Vorteile:

- Sie erhalten **10 Elektor-Hefte** (8 Einzelhefte + 2 Doppelausgaben Januar/Februar und Juli/August) pünktlich und zuverlässig frei Haus.
- **Extra:** Jedes Heft steht Ihnen außerdem gleichzeitig als PDF zum sofortigen Download (für PC/Notebook oder Tablet) bereit.
- **Neu & Exklusiv:** Sie erhalten alle 2 Wochen per E-Mail ein neues Extra-Schaltungsprojekt (frisch aus dem Elektor-Labor).
- **Neu & Exklusiv:** Wir gewähren Ihnen bei jeder Online-Bestellung 10% Rabatt auf alle unsere Webshop-Produkte – dauerhaft!
- **Neu & Exklusiv:** Sie erhalten Online-Zugang zu den neuen Community-Bereichen *Elektor.LABS* und *Elektor.MAGAZINE*, wo wir Ihnen zusätzliche Bauprojekte und Schaltungsideen zur Verfügung stellen.
- **Extra:** Die neue Jahrgangs-DVD 2012 (Wert: 27,50 €) ist bereits im Mitgliedsbeitrag inbegriffen. Diese DVD schicken wir Ihnen sofort nach Erscheinen automatisch zu.
- **Extra:** Top-Wunschprämie (im Wert von 30 €) gibts als Dankeschön GRATIS obendrauf!

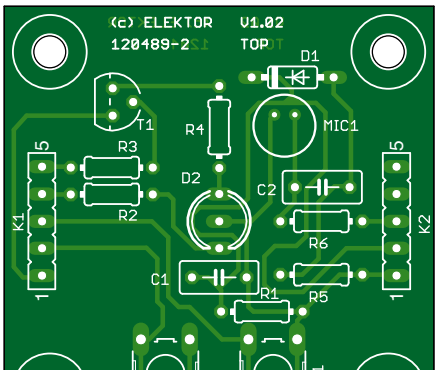
UMWELTSCHONEND – GÜNSTIG – GREEN

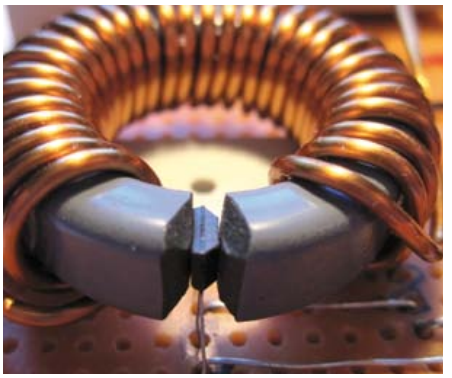
Möchten Sie Elektor lieber im elektronischen Format beziehen? Dann ist die neue GREEN-Mitgliedschaft ideal für Sie! Die GREEN-Mitgliedschaft bietet (abgesehen von den 10 Printausgaben) alle Leistungen und Vorteile der GOLD-Mitgliedschaft.

Jetzt Mitglied werden unter www.elektor.de/mitglied!

Inhalt

elektor magazine

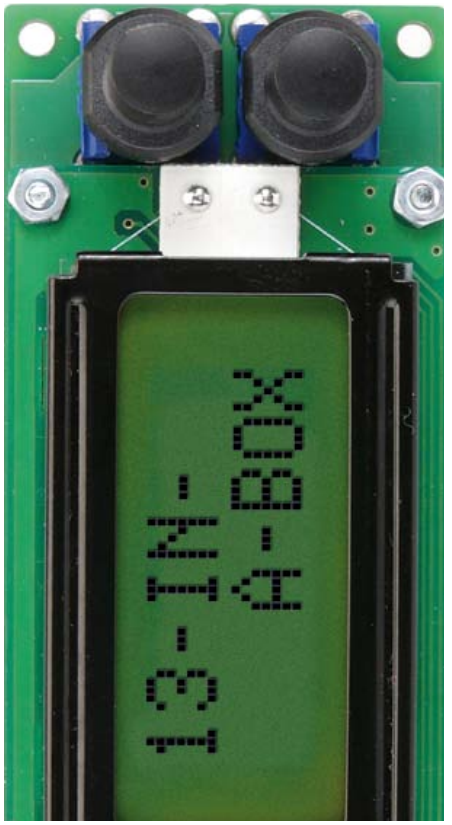


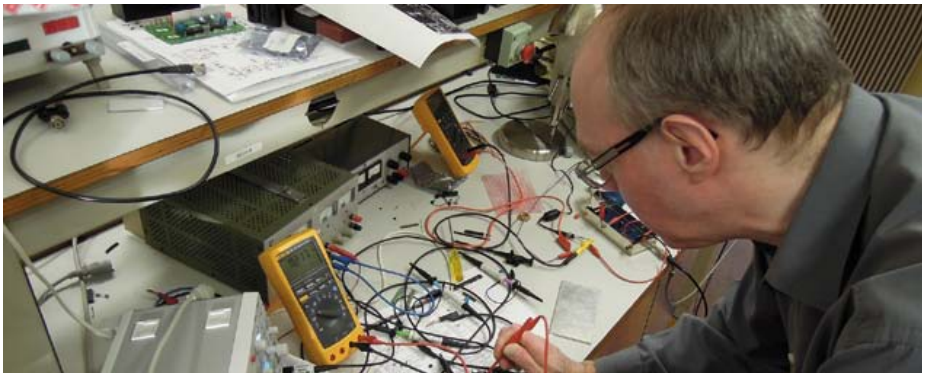


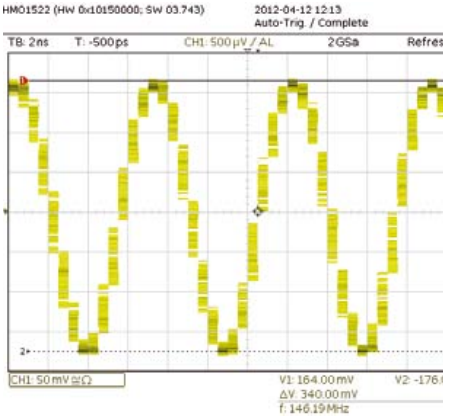
Community

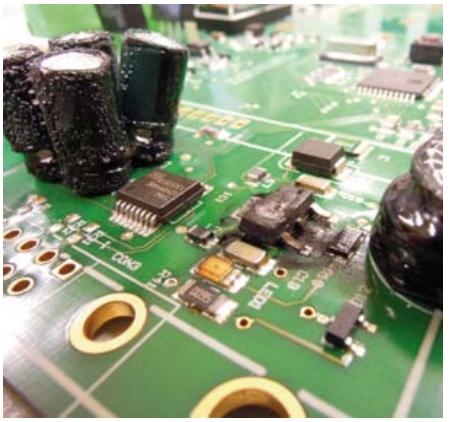
Labs

Projects









Tech the Future

Magazine

Impressum

44. Jahrgang, Nr. 505/506 Januar/Februar 2013
Erscheinungsweise: 10 x jährlich
(inkl. Doppelhefte Januar/Februar und Juli/August)

Verlag
Elektor-Verlag GmbH
Süsterfeldstraße 25
52072 Aachen
Tel. 02 41/88 909-0
Fax 02 41/88 909-77

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Anzeigen (verantwortlich):
Irmgard Ditgens
ID Medienservice
Tel. 05 11/61 65 95-0 | Fax 05 11/61 65 95-55
E-Mail: service@id-medienservice.de
Es gilt die Anzeigenpreisliste Nr. 42 ab 01.01.2012

Vertriebsgesellschaft:
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0 | Fax 0 22 25/88 01-199
E-Mail: elektor@ips-pressevertrieb.de
Internet: www.ips-pressevertrieb.de

Vertrieb Österreich
Pressegroßvertrieb Salzburg/Anif
Niederalm 300
Tel. +43/62 46/37 21-0

Druck: Senefelder Misset, Doetinchem (NL)
ISSN 0932-5468

Stark ins neue Jahr

Premiere: Sie halten hier die erste Doppelausgabe Januar/Februar von Elektor in Händen. Zu unserer extradicken Sommerausgabe (dem „Halbleiterheft“) gesellt sich nun also auch eine Winterausgabe, die über 130 Seiten stark ist. Wir haben uns diesmal aber keine Vorgaben gesetzt, dass wir so-und-so-viele Schaltungen veröffentlichen wollen, und haben uns auch nicht ausschließlich auf Schaltungs-Projekte beschränkt. Daher werden Sie auch in diesem Heft mit News aus der Elektronikwelt, Ideen und Hinweisen von anderen Lesern, Tipps aus unserem Labor, Reviews von neuen Produkten, einer neuen Folge unserer Rubrik „Retronik“ und vielem mehr versorgt, das ebenfalls zu Elektor gehört.
Bei den Projekten haben wir auf die bewährte Mischung aus „Groß“ und „Klein“ gesetzt. So sind zum Beispiel der Feuchte-Schalter, die Duo-LED-Kerze, die Digitaluhr mit Sound oder das 5-V-Netzteil typische „Halbleiterheft“-Projekte, die zum Nachbau einladen. Wer sich vor allem am Entwickeln eigener Applikationen erfreut, sollte sich besonders die neuen Folgen der Serien „Embedded Linux“ und „FPGA – Bauen Sie Ihren Chip“ anschauen. Zum Ende der Linux-Serie zeigt Benedikt Sauter, wie man auf dem PC komfortabel C-Programme erstellen und direkt auf dem Board ausprobieren kann. Und mein Kollege Clemens Valens erklärt uns Schritt für Schritt, wie man eine erste FPGA-Anwendung zum Laufen bringt. Beides will ich bei Gelegenheit auch einmal selbst ausprobieren. Die Artikel „Kapazitiver Näherungsschalter“ und „Rambo-S“ sind noch zwei weitere, persönliche Lese-Tipps von mir.
Ist unsere Artikel-Mischung gelungen? Schreiben Sie uns unter redaktion@elektor.de!

Jens Nickel

Unser Team

Germany
Ferdinand te Walvaart
+31 46 4389417
f.tewalvaart@elektor.de

United Kingdom
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

Netherlands
Harry Baggen
+31 46 4389429
h.baggen@elektor.nl

France
Denis Meyer
+31 46 4389435
d.meyer@elektor.fr

USA
Hugo Van haecke
+1 860-875-2199
h.vanhaecke@elektor.com

Spain
Eduardo Corral
+34 91 101 93 95
e.corral@elektor.es

Italy
Maurizio del Corso
+39 2.66504755
m.delcorso@inware.it

Sweden
Wisse Hettinga
+31 46 4389428
w.hettinga@elektor.com

Brazil
João Martins
+55 11 4195 0363
joao.martins@editorialbolina.com

Portugal
João Martins
+351 21413-1600
joao.martins@editorialbolina.com

India
Sunil D. Malekar
+91 9833168815
ts@elektor.in

Russia
Nataliya Melnikova
+7 (965) 395 33 36
Elektor.Russia@gmail.com

Turkey
Zeynep Köksal
+90 532 277 48 26
zkoksal@beti.com.tr

South Africa
Johan Dijk
+27 78 2330 694
j.dijk@elektor.com

China
Cees Baay
+86 21 6445 2811
CeesBaay@gmail.com

Unser Netzwerk

elektor · VOICE COIL · CIRCUIT CELLAR · hifi.nl MUZIEK - FILM - LIFESTYLE · audioXPRESS · techthe future

Die Elektor-Community

Unsere Partner und Sponsoren

Sie möchten Partner werden?

Kontaktieren Sie uns bitte unter service@id-medienservice.de (Tel. 0511/616595-0).

Elektor World

Zusammengestellt von
Wisse Hettinga

Jeden Tag, jede Stunde, jede Minute, ja, in jedem Augenblick schaffen, optimieren, rekonstruieren und entwickeln Ingenieure und Enthusiasten neue Elektronik. Oft nur zum Spaß, aber manchmal wird daraus (beruflicher) Ernst…

DAS INGENIEURINNEN-ARCHIV

VERNON — Kollege CJ von der Zeitschrift Circuit Cellar entdeckte ein interessantes Objekt – Frauen. Und vor allem – Frauen im Ingenieurwesen. CJ schrieb: „ In den späten achtziger und frühen neunziger Jahren trugen Ingenieure, die sich mit Mikrocontrollern beschäftigten, in Ingenieurfirmen arbeiteten und Elektrik/Elektronikvorlesungen an welterneuernden Universitäten hielten, Schnurrbärte. Kaukasische Männer, sportgestählt und mit fest geknöpftem Kragen, mit dicken bifokalen Brillen und Taschenschonern. OK, vielleicht verallgemeinere ich *ein klein wenig*. Es gab ein paar coole Jungs (vielleicht zehn?), deren Charaktereigenschaften (enthusiastisch, kreativ, neugierig), Interessen (gute Musik, Literatur, Motorräder) und Moden (professionell, aber lässig im Labor) das Aufkommen dieser geselligen, unprätentiösen, multitalentierten Techies des 21. Jahrhunderts vorhersagten, die die weltumkehrenden neuen Technologien entwickeln sollten. So, worin liegt das Problem? Wir haben doch nun faszinierende Techniker, mit denen wir arbeiten können, stimmt's? Das bemerkenswerteste Problem ist aber – es sind fast alles *Jungs*. Für zu lange Zeit waren weibliche Ingenieure in der Technikergemeinde unterrepräsentiert. Aber in den letzten paar Jahren ist doch eine Veränderung eingetreten und mehr und mehr weibliche Ingenieure drängen in den Vordergrund! Im Jahr 2012 hat *Circuit Cellar* einen besonderen Blick auf diese talentierten Erfinderinnen geworfen. Hier ein paar:

- Ayse Kivilcim Coskun („A Vision for 3-D Stacked Systems," *Circuit Cellar* 264)
- Hai (Helen) Li („Embedded Inquiries," *Circuit Cellar* 267)
- Jan Axelson („Debugging USB Firmware," *Circuit Cellar* 268)

Hai (Helen) Li

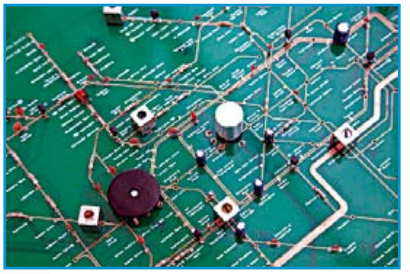
Und es gibt mehr im folgenden Jahr! Besuchen Sie unser „Female Engineers"-Archiv: http://circuitcellar.com/category/female-engineers/

TREFFEN SIE IHRE NEUE KOLLEGIN!

PUNE — Während in Europa Industrie und Bildungssektor händeringend neue Ingenieure suchen, kann man in Indien vor lauter Ingenieursstudenten kaum noch treten. Im Bharati Vidyapeeth's College of Engineering for Women in Pune studieren Mädchen IT und Embedded Electronics Engineering und sie sind eifrig und enthusiastisch bei der Sache. Diese Schule ist nur eine von rund 80 Ingenieur-Colleges allein in Pune! Dank des Ingenieurmangels in Europa haben Sie eine gute Chance, eines dieser Mädchen bald als neue Kollegin begrüßen zu dürfen.

„UNDERGROUND“-RADIO

LONDON — Erinnern Sie Sich, dass wir in der Sommerausgabe 2012 gefragt hatten, was Sie von bizarren Platinenlayouts halten – so wie den



Plan, das Londoner U-Bahn-Netz in eine funktionierende Schaltung zu verwandeln? Yuri Suzuki, ein geübter Designer, hat genau so verrückt gedacht und die *Tube Map* in ein funktionierendes Radio verwandelt. Das Ergebnis können Sie im *London Design Museum* und auf Yuris Website bewundern. http://yurisuzuki.com/works/tube-map-radio/

THE CASE AGAINST KILLER ROBOTS

AMSTERDAM — Es passiert nicht oft: Ein Aufruf zu einer öffentlichen Debatte über den Sinn einer Technologie, die wohl noch 20 bis 30 Jahre bis zu ihrer Verwirklichung benötigt. Aber genau dies ist es, was *Human Rights Watch* und *Harvard Law*

School's International Human Rights Clinic in ihrem Report *Losing Humanity: The Case Against Killer Robots* fordert. Beteiligen Sie Sich an der Debatte zu diesem Thema unter Techthefuture.com!

TEACHERS! LEAVE THEM KIDS ALONE

LIMBRICHT — Wegen einer Einladung ließ eine Gruppe von Lehrern ihre Schüler für einen Tag herrenlos zurück. Elektor wollte wissen, was im Technikunterricht in Holland so passiert und bot einigen Technik-Lehrern einen Rundgang durch unsere Labors an. Es entwickelte sich ein angenehmes informatives Miteinander. Wenn

auch Sie Technik/Elektronik unterrichten und an einem Besuch im Labor interessiert sind, lassen Sie es uns wissen. Sie sind immer willkommen, schicken Sie eine Mail oder rufen Sie an und wir arrangieren etwas..

HP35: EINE WOHLKALKULIERTE ANTWORT

Liebe Elektor-Redakteure - ich war sehr erfreut, in eurem „Retronik“-Artikel in Elektor November 2012 vom Taschenrechner HP35 zu lesen, wahre Nostalgie! Ich erinnere mich, einer der ersten Nutzer gewesen zu sein, und das Teil hat mich damals 2100 niederländische Gulden gekostet. Ich habe den Taschenrechner noch immer und er funktioniert. Nur der Ein/Ausschalter ist ein wenig unzuverlässig von Zeit zu Zeit. Ich habe auch die Modelle 31E, 71B, 18C, 28C und 200LX. Können Sie sich ein gutes neues Zuhause für diese Rechner vorstellen, etwa ein Museum? Zurzeit machen die Rechner wenig außer Staub zu sammeln, aber ich möchte sie nicht einfach wegschmeißen.
Gerard Keijser, Niederlande

Redakteur Jan Buiting dazu:
Ich freue mich natürlich riesig über Reaktionen der Leser zu den „alten Seiten“ am Ende des Hefts namens *Retronik*. Gerard war damit einverstanden, die Taschenrechner gegen mein Retronik-Buch zu tauschen. Die Kollektion ist hier zur Bewunderung und Erinnerung abgebildet.

PING-PONG MIT ARDUINO

CHIASSO — Wenn Sie das Schweizer Arduino-Büro in Chiasso für ein Meeting besuchen, enden Sie wahrscheinlich an einer Tischtennisplatte. Nicht etwa, um zu spielen, sie haben dort einfach keinen Tisch, der groß genug für sechs Leute ist. Elektor war da, um einen weltweiten Distributionskontrakt für Arduino-Produkte zu unterzeichnen und jawoll, es gab etwas zu essen und zu trinken, um das Ereignis zu feiern. Von links nach rechts sehen Sie Don, wie er Gianlucca überreden möchte, Massimo, wie er seine Mails checken möchte, David, wie er aus dem Bild verschwinden möchte und Eduardo, wie er sich von seiner besten Seite zeigen möchte. Seien Sie bereit für mehr Arduino-Projekte!

and now for something completely different...

Feedback zum Stromwandler

Ein interessanter technischer Thread entwickelte sich zum Artikel Stromwandler aus dem Oktoberheft 2012. Redakteur Jan Buiting hatte die Ehre, in der Korrespondenz das Mäuschen spielen zu dürfen. Sehen Sie, wie sich das Feedback entwickelte und spezialisierte.

Hallo Jan,

Da ich Stromwandler für die Luftfahrtindustrie entwickle, fand ich den Artikel „Stomwandler“ von Ed Dinning sehr interessant. Ich habe einen Kommentar zu Bild 5. Der Autor zeigt den Abschlusswiderstand RL auf der Sekundärseite des Stromwandlers an den Gleichrichterdioden. Dies bewirkt einen Ablesefehler am Instrument, der abhängig von der temperaturabhängigen Schwellspannung der Dioden ist. Da der Stromwandler eine Stromquelle darstellt, würde es die Anzeige von dem Temperatureinfluss der Diodenschwellspannung befreien, setzte man den Widerstand auf die Seite des Messwerks.
Ich habe diese Art, den Abschlusswiderstand zu platzieren, in meinem Artikel „Belastungsanzeige für Notstrom-Aggregate“ (Elektor 2/2012, Bild 2) verwendet. Und ich habe Schottky-Dioden eingesetzt, um die Belastung des Stromwandlers zu verringern.
Ich habe auch die Vorteile von Ringkernen im Artikel „All About Toroidal Transformers“ (Popular Electronics, Mai 1996, Seite 53) beschrieben.
Best regards,
BR, Chuck Hansen (USA)

Hi Chuck,

vielen Dank für die Mail, ja ich bin total einverstanden mit dem, was du über die Position des Abschlusswiderstands und des Temperaturkoeffizienten der Diodenschwellspannung sagst.
Die Idee hinter dem Artikel war ein einfaches isoliertes Anzeigeinstrument, das der gemittelte Hobbyist nutzen kann, zugegebenermaßen mit einer kleinen Nichtlinearität am unteren Ende der Skala.
Wenn Schottky-Dioden, typische 1 A-Typen, verwendet werden, hat ein Strom von 50 mA nur geringe Auswirkungen auf deren Vf.
Die Genauigkeit wird auch erhöht, wenn mit einer geringeren Flussdichte gearbeitet wird, aber der verwendete Kern ist leicht erhältlich und kann auch aus älterem Equipment ausgeschlachtet werden. Mit welcher Flussdichte würdest du bei deinen Ringkernen arbeiten? Als ich vor einigen Jahren bei einem Schaltanlagen-Hersteller gearbeitet habe, haben wir spezielle Legierungen eingesetzt und arbeiteten mit 0,01 T.
BR, Ed Dinning (UK)

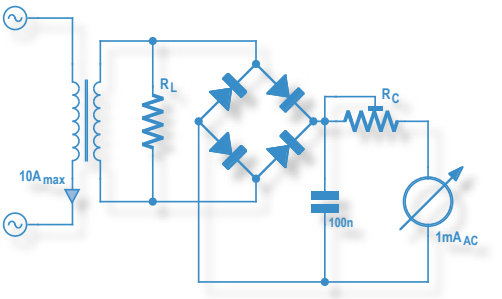
Hello Ed,

die Idee mit den „wiederaufbereiteten“ alten Trafos finde ich prima. Es ist aber schwer, das Spannungsverhältnis irgendeines kommerziell erhältlichen Trafos zu ändern, indem man die Anzahl der sekundären Windungen anpasst, weil die Sekundärwicklung sich immer auf der Innenseite befindet und die Primärwicklung für beste Fluxkopplung darüber. Und einige Extrawindungen auf der Außenseite der Primärwicklung anzubringen ist ebenso kompliziert, da das Kernfenster normalerweise ausgefüllt ist, um ein minimales Trafogewicht (und niedrige Kosten) zu erzielen.
In der Luftfahrt ist das Gewicht natürlich wichtig, deshalb habe ich die Flussdichte des Stromwandlers höher angesetzt als normal üblich. Die Anforderungen, die an den Stromwandler gestellt werden, unterscheiden sich natürlich, aber generell begrenze ich (bei Silizium/Eisen mit Bmax 16 kGauss oder 1,6 T) die Volllast-Flussdichte auf 3200 Gauss (0,32 T).
Wir Yankees haben scheinbar eine Aversion gegen das metrische System, aus mir unerfindlichen Gründen. Generator-Entwickler arbeiten immer noch mit Flussdichten in lines/sq inch!

Ein differentieller Schutz (zwei Stromwandler gegenphasig über einer Schutzschaltungslast) funktioniert gut, auch beim größtmöglichen Fehler-Durchlassstrom. Ich habe die Flussdichte für den Worst-case-Fehlerstrom begrenzt, unter Berücksichtigung des subtransienten Stroms, auf ungefähr 1,4 T bei maximaler Kerntemperatur. Mit der spezifizierten sekundären Last und dem primären Nennstrom darf der Phasenwinkelfehler 20 Minuten nicht überschreiten.
Wenn der Stromwandler zum Überstrom-Schutz oder zur Strombegrenzung in einem Regler eines Wechselspannungsgenerators eingesetzt wird, hängt der von mir verwendete Kernquerschnitt vom Lastwiderstand der Schaltung und von ein paar Gleichrichterdioden ab. Für Anzeigen und Messtechnik definieren die Militär/Aerospace-Spezifikationen die 400 Hz Lastimpedanz auf 1,48 + j2,09 Ω ±10%.
Wir haben hauptsächlich 3%-ige Silizium/Eisen-Ringkerne für unsere Stromwandler verwendet, aber für spezielle Anwendungen gibt es auch andere Legierungen:
Satmu-Metall, eine spezielle kornorientierte 65%-iges Nickel/Eisenlegierung, weichgeglüht in einem magnetischen Feld, entwickelt von Telcon Ltd in Großbritannien, dem originalen Entwickler von Mu-Metall, oder Deltamax Round Orthonol oder Supermalloy-Pulvermetall für Frequenzen über 2 kHz.
BR, Chuck Hansen

Hi Chuck,

danke für die Info. Magnetismus wurde in englischen Schulen sehr schlecht gelehrt, erst in Zoll und so weiter, dann im CGS-System. Und erst, wenn man zur Uni geht und die in SI (nicht MKS) lehren, wird alles klar. Es gibt hier einen Verein, der den Zentimeter abschaffen möchte (eine gute Idee, haben die Schulen doch die Metrifizierung gründlich vermasselt).
Bei den meisten der Transformatoren hier ist die Primärwicklung zuerst gewickelt, sodass sie einfacher modifiziert werden können. Auch gibt es viele Typen, bei denen Primär- und Sekundärseite aus Sicherheitsgründen Seite an Seite gewickelt sind (Kriechstrom und Aufbauhöhe); leicht neu zu wickeln, aber miserabel einzustellen.
Lediglich bei Audiotrafos und in Schaltnetzteilen, wo primär und sekundär sektionsweise und abwechselt gewickelt wird, um eine optimale Kopplung zu erzielen, wird davon abgesehen, die Primärwicklung innen anzubringen.
Ich habe ein wenig in der Luftfahrtindustrie gearbeitet; ein Design benötigte einen Supermendur, der auf 2 T / 400 Hz lief und zwangsgekühlt werden musste. Während der Magnetostriktion musste man einen Gehörschutz tragen. Er war für die Trafo/Gleichrichter-Einheit eines militärischen Flugzeugs bestimmt.
Ein weiteres interessantes Design war ein Frequenzkonverter in einer militärischen Transportmaschine. 3 Phasen nach 18 Phasen, für einen „wilden“ Frequenzbereich von 380...620 Hz, gekühlt mit Kerosin. Es hat vielleicht die Generatoren vereinfacht und den Leistungsfaktor verbessert, aber sicher die Wicklungen und die Elektronik verkompliziert!
Ich habe in dem Elektor-Artikel absichtlich auf alles verzichtet, was mit dem i-Zeichen und dem Leistungsfaktor zu tun hat.
BR, Ed Dinning



Hi Ed,

ich bin mit der Behandlung der komplexen Zahlen einverstanden. Als wir die Testmessungen an den Stromwandlern begannen, haben wir einen 2R56-Drahtwiderstand (2,56 W) als Abschlusswiderstand einsetzen dürfen. Die Messergebnisse waren alle gleich.
BR, Chuck Hansen

Hello,

Ich habe eine Verbesserung in Bezug auf das Beispiel Nr.2 deiner „Leserschaltung: Stromwandler“ aus der Oktoberausgabe 2012. Es zeigt den Ausgang eines Stromwandlers, an den der Lastwiderstand angeschlossen ist (der immer angeschlossen sein muss, wie Du richtig sagst). Allerdings nimmt der Brückengleichrichter das SPANNUNGSSIGNAL vom Lastwiderstand und wegen der Schwellspannung der Brücke ist der Ausgang ein wenig nichtlinear, so dass kleine Stromsignale nicht angezeigt werden. Eine bessere Lösung ist es, den Ausgangsstrom auf der Trafoseite ERST gleichzurichten und DANN, nach der Gleichrichtung, den Strom mit einem Widerstand zu belasten.
Da die Ausgangsgröße eines Stromwandlers eben ein Strom ist und er als Stromquelle fungiert, muss der gleiche Strom vom Brückengleichrichter durch den Abschlusswiderstand fließen. Der Trafo muss lediglich für ein wenig mehr Spannung sorgen, um die Schwellspannung der Dioden zu überwinden und einen Sekundärstrom fließen zu lassen. Nun ist der Strom durch den Lastwiderstand unidirektional und ein einfaches RC-Filter ist ausreichend, um das Signal zu glätten und für die Anzeige zu skalieren. Das Instrument kann auf die kleinsten Sekundärströme reagieren, ohne dass man sich Sorgen wegen der Schwellspannung der Brücke machen müsste.
Der Effekt der Schwellspannung des Brückengleichrichters wird im letzten Abschnitt genannt, mit der Mathematik des Artikels gehe ich konform. Danke für diese gute Lehrstunde zum Thema Stromwandler, ich hoffe, sie wird gleichermaßen Ingenieure wie Hobbyisten zu einer korrekten Anwendung anregen.
KR, Jim Mettler, Triad Magnetics

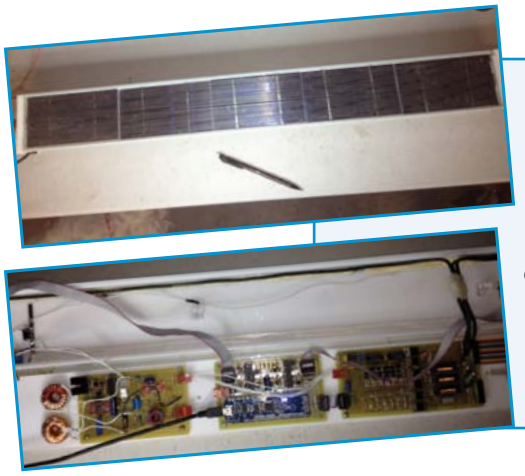
RL78 Green Energy Challenge
Die Gewinner

Beim RL78 Green Energy Challenge geht es um neue Entwicklungen für den modernen Alltag, in dem geringer Energieverbrauch, hoher Wirkungsgrad und erneuerbare Ressourcen zunehmend wichtiger werden. Teilnehmer aus über 67 Ländern maßen ihre Fähigkeiten bei der Entwicklung von „grünen" Anwendungen – wie beim Energy Harvesting, bei Messaufgaben oder bei energiesparenden Steuerungen – alles auf der Basis des RL78-Mikrocontrollers von Renesas mit einer von Renesas und Partnern zur Verfügung gestellten robusten Entwicklungsumgebung.

Vielen Dank für die Teilnahme und Glückwünsche an die Gewinner!

Die kompletten Wettbewerbsbeiträge samt Zusammenfassungen finden sich auf:

www.circuitcellar.com/contests/renesasRL78challenge/



1. Preis
Electrostatic Cleaning Robot

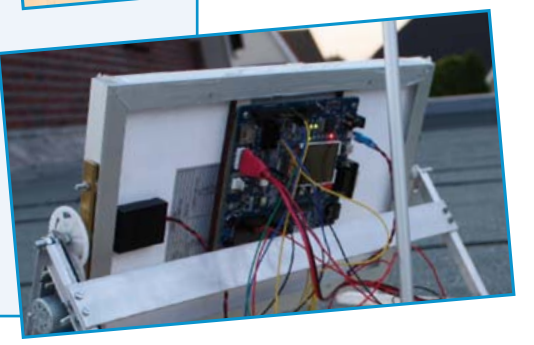
Scott Potter (USA)

Nachgeführte Spiegel, sogenannte Heliostaten, sind integrale Teile sogenannter CSP-Anlagen (**C**oncentrating **S**olar **P**ower). Für eine maximale Dampfausbeute, die ja in Energie transformiert wird, müssen die Spiegel sauber bleiben. Der innovative elektrostatische Reinigungs-Roboter auf RL78-Basis reinigt zuverlässig und wird zudem noch von Solarzellen gespeist. Er wandert über die Spiegel und beseitigt mit Hilfe eines Hochspannungs-Wechselfelds Staub und andere Ablagerungen.

2. Preis
Cloud Electrofusion Machine

Michael Hamilton (USA)

Gegenüber kommerziellen Elektroschweißgeräten benötigt diese Maschine 400 Mal weniger Energie. Die Cloud Electrofusion Machine eignet sich für das Verschweißen von 0,5...2"-Fittings aus Polyethylen. Das RL78-basierte Gerät liest den Bar-Code auf den Fittings zwecks Bestimmung der Verbindungsparameter und der Rückverfolgbarkeit. Vorhandene Barcode-Daten werden zusammen mit den GPS-Positionen auf einer SD-Karte abgelegt und wenn vorhanden zur Nachvollziehbarkeit und zur Qualitätskontrolle in einer Cloud-basierten Datenbank gespeichert.



3. Preis

The Sun Chaser: A GPS Reference Station

Sjoerd Brandsma (Niederlande)

Beim Sun Chaser handelt es sich um ein gutes, solarbasiertes Energy Harvesting System, das automatisch die optimale Ausrichtung eines Solarpanels berechnet. Bei Montage auf einer rotierenden Scheibe wird die Orientierung des Solar-Panels mit der registrierten GPS-Position bestimmt. Mit einem externen Kompass, einem internen Accelerometer, einem DC- und einem Schrittmotor kann die exakte Position des Panels bestimmt werden. Das System basiert auf dem Evaluations-Board RDKRL78G13 von Renesas, auf dem ein µC/OS-III-Real-Time-Kernel von Micrium läuft.

Besondere Erwähnung:

Water Heater by Solar Concentration

Pierre Berquin (Frankreich)

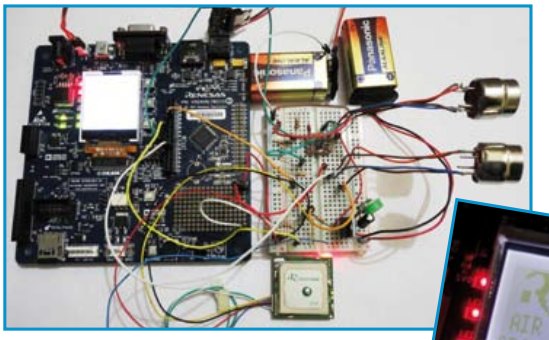
Diese Warmwasseranlage basiert auf dem RL78-Evaluations-Board und wurde dazu entwickelt, das Maximum des verfügbaren Sonnenlichts auf ein Wasserrohr zu bündeln, um dieses kontinuierlich zu erhitzen. Der Reflektor ist mit einem Gegengewicht für einfaches Kippen versehen und wird automatisch so verstellt, dass ein Maximum des Sonnenlichts mit dem geringsten Strombedarf reflektiert wird.

Besondere Erwähnung:

Air Quality Mapper

Raul Alvarez Torrico (Bolivien)

Wollen Sie sicher gehen, dass die Luft auf ihrer täglichen Laufstrecke gut ist? Der Air Qualität Mapper ist ein portables Gerät, das die Konzentration von CO_2 und CO aufzeichnet. Mit den so erzeugten „Smog-Karten" kann man die gesündesten Routen bestimmen. Das Gerät basiert auf einem RDKRL78G13 und erhält die Positionsdaten aus seinem GPS-Modul, die



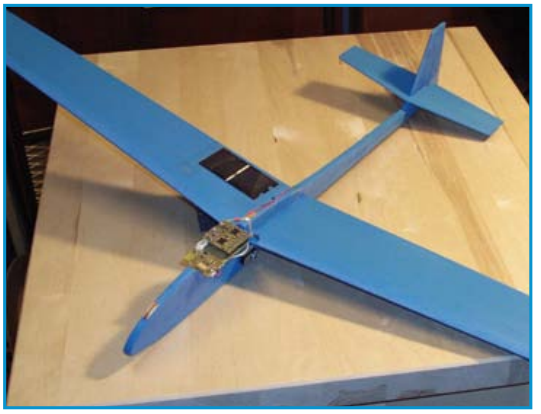
es zusammen mit CO_2- und CO-Konzentrationen der gewählten Route auf einer SD-Karte speichert. Mit einem PC kann man sich diese Daten anschauen und sie in einer Online-MySQL-Datenbank ablegen, wo sie dann in eine Google-Karte integriert werden.

Besondere Erwähnung:

High-Altitude Low-Cost Experimental Glider (HALO)

Jens Altenburg (Deutschland)

Das experimentelle Gleiter-Projekt „HALO" besteht aus drei wichtigen Teilen: Einem Wetterballon als Träger, einem Gleiter (die



Nutzlast des Ballons) und einer Basis-Station am Boden zur Kommunikation und Anzeige von Telemetrie-Daten (nicht Teil des Wettbewerbs). Für Tests wurde der REFLEX-Flugsimulator genutzt. Der Gleiter hat einen eigenen Mikro-GPS-Empfänger, Sensoren und einen energiesparenden Mikrocontroller. Das System kann aufsteigen, eine bestimmte Höhe erreichen und zu festgelegten Koordinaten zurückkehren.

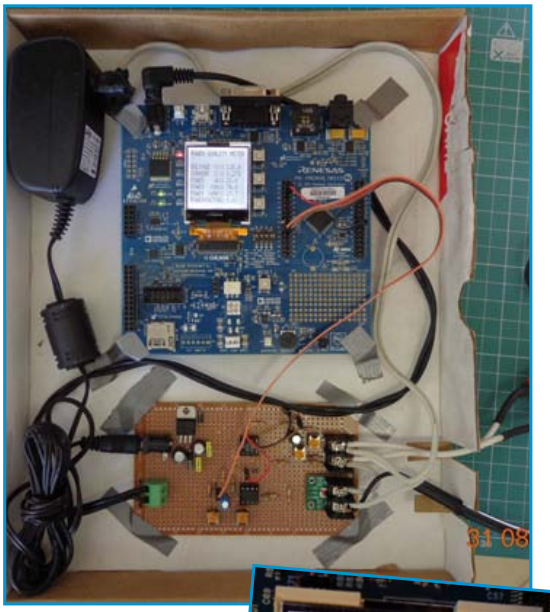
Besondere Erwähnung:
Wireless Remote Solar-Powered „Meteo Sensor“

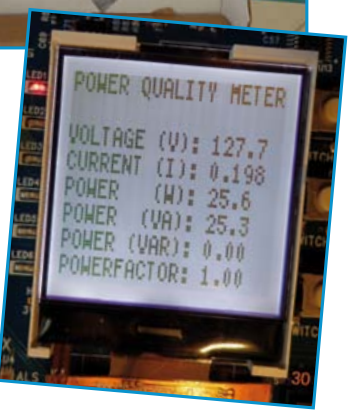
Grzegorz Kaczmarek (Polen)

Mit dem „Meteo Sensor“ kann man sehr einfach meteorologische Daten erfassen. Die RL78-basierte Elektronik führt zyklische Messungen von Temperatur, Luftfeuchtigkeit, Luftdruck sowie Betriebsspannung durch und gibt die Daten per Funk weiter. Der Empfänger lauscht auf die gesendeten Daten. Dieser Ansatz vereinfacht die Erfassung von Wetterdaten und den Aufbau eines lokalen Mess-Netzwerks. Er ist für einen niedrigen Energieverbrauch und eine lange Batterie-Lebensdauer optimiert.

Besondere Erwähnung:
Portable Power Quality Meter

Andrè Barbosa (Brasilien)





Die Überwachung des häuslichen Stromverbrauchs wird immer angesagter. Das Portable Power Quality Meter nutzt einen RL78-Controller für die Berechnung des Leistungsfaktors, der totalen harmonischen Verzerrungen, der Netzfrequenz, der Netzspannung und des Energieverbrauchs. Die Daten werden für spätere Analysen gespeichert.

Kostenlose Dozentenseminare

Egal ob im Informatik-, Mechatronik oder Elektrotechnikunterricht: Mikrocontroller- und Programmierkenntnisse werden immer wichtiger. Solche Kenntnisse vermittelt man am besten nach dem Prinzip „Learning by Doing“ und dazu gehört ein flexibel zu nutzendes, modulares System aus Hardware und Software.
Die E-blocks-Module kann man einfach zusammenstecken, ein Löten ist hier nicht nötig. Mit den mehr als 300 Einzelelementen lassen sich eine Vielzahl kleinerer und größerer Projekte aufbauen, die einen großen Anwendungsbereich abdecken. Dazu gehört die Entwicklungsumgebung „Flowcode“, mit der auf einfache, grafische Weise Software erstellt werden kann. So entstehen in kurzer Zeit funktionierende Systeme – von LED-Blinkern bis hin zur PID-Motorsteuerung.
Für alle Lehrer, Ausbilder und Dozenten haben wir einen guten Tipp: Elektor organisiert kostenlose und unverbindliche Einführungsseminare zu den E-blocks, los geht es am 18. Februar 2013 in Bocholt (NRW).

www.elektor.de/dozentenseminar

Protokollanalyse bei Oszilloskopen

Hameg bietet eine Bus-Protokoll-Analyse auch für die Mixed-Signal-Oszilloskope der Einstiegsklasse an. Die Option HOO12 erlaubt das Triggern und Decodieren von CAN- und LIN-Busprotokollen. Zusammen mit den Optionen HOO10/11 für I2C, SPI und UART/RS232 erhalten Embedded-Entwickler eine Komplettlösung für die Entwicklung von Systemen, die über einen Bus kommunizieren. Für alle Protokolle wurde eine Tabellendarstellung der decodierten Werte implementiert, die in einer Zeile die kompletten Informationen eines Nachrichtentelegrammes darstellen. Spezielle Triggereigenschaften erlauben dabei die gezielte Isolierung einzelner Nachrichten. Sehr hilfreich ist auch die für individuelle Messaufgaben mögliche Beschriftung der 2 bzw. 4 analogen und 8 bzw. 16 digitalen Kanäle. Passend für die Protokollanalyse des CAN-Busses bietet der Hersteller einen 200-MHz-Differenztastkopf und einen 800-MHz-Differenztastkopf an. Beide zeichnen sich durch hervorragende elektrische Eigenschaften (3,5 pF bzw. 1 pF Eingangskapazität, 1 MΩ bzw. 200 kΩ Eingangswiderstand), eine Vielzahl praktischen Zubehörs und flexible Stromversorgungsmöglichkeiten aus. Kombiniert mit dem entsprechenden Oszilloskop der HMO-Serie (oder jedem anderen Oszilloskop) lassen sich damit viele Messungen differentieller Signale realisieren, ohne das Signal merklich zu belasten.

www.hameg.de

RFID-Chips in Platinen

Im letzten Heft haben wir darüber berichtet: Beta Layout ist es gelungen, Platinen mit RFID-Chips auszustatten. Leider wurde bei der Nachricht ein falsches Bild abgedruckt. Das richtige Foto zeigen wir hier: Die Chips werden schon bei den ersten Produktionsschritten in die Platinen integriert, die damit eindeutig identifiziert werden können. Dies stellt gleichzeitig eine Art unsichtbarer Kopierschutz dar. Die RFID-Chips arbeiten im UHF-Band zwischen 860 und 960 MHz, das weltweit verwendet wird.

www.beta-layout.com

Intelligente TFT-Module

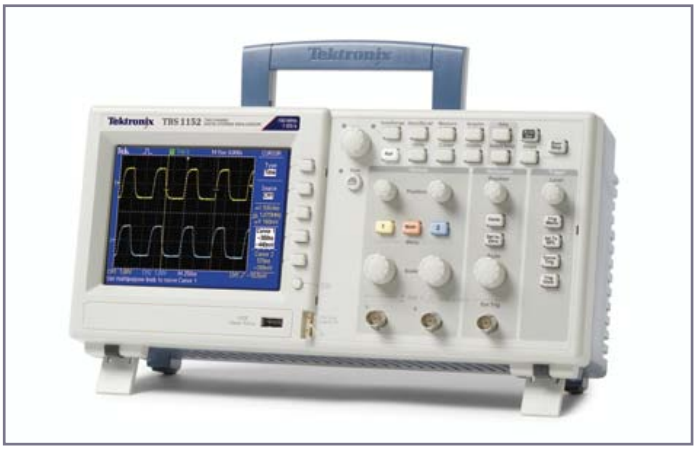
War die Ansteuerung von TFT-Displays bisher mit erheblichem eigenen Hard- und Softwareaufwand verbunden, so sind die intelligenten Displays von Electronic Assembly durch das integrierte Mikrocontrollersystem samt optionalem Touchpanel sofort betriebsbereit. Lediglich eine Stromversorgung von 5 V ist erforderlich. Mit den 3,2"- und 4,3"-TFT-Displays hat Reichelt nun solche intelligenten TFT-Module in sein Programm aufgenommen.

Neben diversen Schriftarten sind auch umfangreiche Grafikfunktionen schon ab Werk installiert. Die TFT-Displays mit 16 bit Farbtiefe und Auflösungen von 480x272 (4,3") bzw. 320x240 Pixeln (3,2") lassen sich in beliebiger Lage einbauen. Schriftarten, Bilder, Animationen und Makros können wahlweise über RS232, I2C oder SPI in das 4 MB große Flash-ROM übertragen werden.

Mit dem ebenfalls bei Reichelt erhältlichen Evaluation Board EA 9777-2USB lassen sich die TFT-Displays bequem programmieren, parametrisieren und prüfen. Für Einsteiger in die Parametrierung der TFT-Displays hat der bekannte Distributor ein Starterkit zusammengestellt, das alles für eine schnelle und einfache Einarbeitung bietet: Das Paket besteht aus einem 4,3"-TFT-Display mit Touchscreen, dem Evaluation Board EA 9777-2USB sowie Treiber-CD, Editor, Compiler und Demos.

www.reichelt.de

Einsteigeroszilloskope von Tektronix



Tektronix ist schon seit langem für professionelle Messgeräte bekannt, die folglich auch in höheren Preisregionen angesiedelt sind. Jetzt aber wurden DSOs der neuen Reihe TBS1000 vorgestellt, die speziell für den Bereich Unterricht und Hobby gedacht sind. Die DSO-Familie besteht aus fünf Modellen preiswerter Zweikanal-Oszilloskope mit Bandbreiten von 25 bis 150 MHz bei einer Abtastrate von 500 bis 1000 MSamples/s und einer immerhin fünfjährigen Garantie. Dabei wird bei der neuen TBS1000-Familie weitgehend Technik der bewährten Reihe TDS2000 eingesetzt, was gleiche Möglichkeiten bezüglich Triggerung, automatische Messungen, USB-Anschlüsse und FFT-Funktionen bedeutet.

Geräte der TBS1000-Reihe verfügen über einen USB-2.0-Anschluss auf der Frontplatte, über den z. B. Messdaten auf einem USB-Stick gespeichert werden können. Ein weiterer USB-2.0-Anschluss auf der Rückseite ist für den Anschluss an einen PC oder einen PictBridge-kompatiblen Drucker gedacht. Speziell für den Unterricht wird zum DSO eine CD mit Experimenten, Anleitungen für Dozenten und Einführungen in das Messen mit Oszilloskopen beigelegt. Die neuen DSOs sind mit einem 5,7"-TFT-Farb-Display ausgestattet und wiegen nur 2 kg.

www.tektronix.com

Updates und Korrekturen

Zapper nach H. Clark

Elektor Halbleiterheft 7-8/2010, S. 83 (090030)

Für die korrekten Zeiten sollten bei dem „Zapper" einige Bauteilwerte geändert werden: Damit die Abschaltzeit etwa zwei Minuten beträgt, sollte R4 auf 150 kΩ und C3 auf 470 nF gesetzt werden. Damit der Frequenzhub von 28...75 kHz genau passt, sollte der Wert von R12 besser 6,8 kΩ und der von C5 dann 2,2 nF betragen.

GPIB-nach-USB-Konverter

Elektor 11/2012, S. 48 (100592)

Unser Leser Dieter Augustin hat uns mitgeteilt, dass der GPIB nicht erst 1984 eingeführt, sondern bereits Mitte der 70er Jahre standardisiert wurde.
Dazu der Wikipedia-Link:
http://de.wikipedia.org/wiki/GPIB
Vielen Dank für diesen Hinweis!

Nixie-VU-Meter

Elektor 11/2012, S. 28 (110744)

Es gibt leider fehlerhafte Bauteil-Bezeichnungen im Schaltplan und in der Stückliste.
IC5 und IC7 sind vom Typ 78L05, nicht vom Typ 7805, und in der Stückliste sind einige ICs falsch nummeriert.
Die Stückliste lautet korrekt:
IC3 = LM324
IC4 = Nicht vorhanden
IC5,IC7 = 78L05
IC6 = NE555.

SDN – Software Defined NIC

Elektor 11/2012, S. 54 (110733)

Der Weblink zur Elektor-Webseite zum Artikel weist leider einen Tippfehler auf.
Richtig lautet er: [2] www.elektor.de/110733

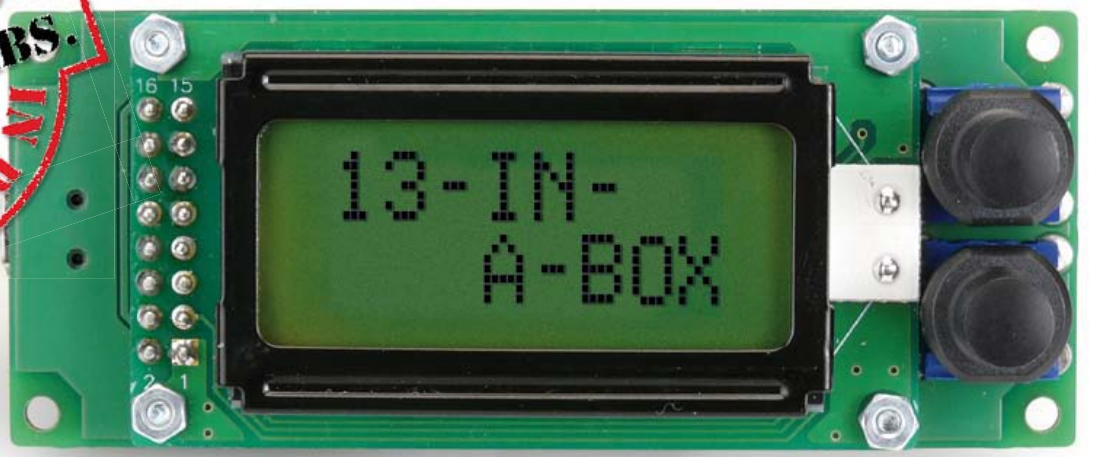
Retronik: HP-35: Revolution in der Tasche (1972)

Elektor 11/2012, S. 78 (120454)

Bei diesem Artikel haben sich Übersetzungsfehler eingeschlichen, auf die uns der Autor freundlicherweise hinwies.
Die Textpassage: „Weiter musste man sich als Programmierer nicht mit einer begrenzten Stack-Tiefe beschäftigen" ist folgendermaßen korrekt: „Für den Anwender entfällt damit auch die Beschränkung bei der Anzahl der Klammerebenen. Er sollte dabei aber nicht die maximale Stacktiefe von vier Registern aus den Augen verlieren."
An anderer Stelle findet sich noch ein Fehler, welcher besagt, dass HP „mehrere hundert" Taschenrechnermodelle entwickelte – tatsächlich sollte es „mehr als hundert" Modelle heißen.

Helferlein im Laboralltag

Ideen für einen handlichen Multitester



Von **Thomas Ücok** und **Jens Nickel** (D)

Bei der Entwicklung von Mikrocontroller-Anwendungen sind immer wieder kleine Tests nötig. Gibt der UART tatsächlich etwas aus? Welcher Pegel liegt gerade an Pin PC2? Habe ich meinen Timer richtig konfiguriert, um eine Frequenz zu bestimmen? Für all diese und noch viele weitere Aufgaben hat unser Leser Thomas Ücok einen handlichen Multitester auf Basis des bekannten Minimod-Moduls von Elektor entwickelt, der Grundlage für ein interessantes Elektor-Projekt ist.

Auf dem Tisch eines Elektronikentwicklers herrscht im Normalfall ja drangvolle Enge. Wenn mal schnell etwas zu testen ist, muss oft erst Platz geschaffen werden für Oszilloskop, Frequenzgenerator und Co. Dann muss man die Geräte noch hochfahren und einstellen. Gerade bei kleinen Test-Aufgaben ist dieser Aufwand gar nicht nötig, wie wir hier beweisen. Mit dem Minimod-Modul [1] von Elektor, das über einen AVR-Mikrocontroller und ein Display verfügt, hat der Autor einen handlichen Multitester realisiert, der einfach bedienbar ist (**Bild 1**). Folgende Funktionen lassen sich über ein Menü auswählen:

1. TTL-Pegeltester 5 V/3 V (High/Low/Open)
2. Frequenzmessung bis 7 MHz
3. CMOS-Pegeltester 5 V (High/Low/Open)
4. Analog-Spannungsmessung 0..5 V
5. Impulszähler (Pegelwechsel nach High)
6. Impulszähler (Pegelwechsel nach Low)
7. UART-Test-Empfänger
8. UART-Test-Sender
9. Auslesen der ROM-ID eines 1-Wire-Sensors
10. Pulslängenmessung Low-Pegel (max. 45 ms)
11. Pulslängenmessung High-Pegel (max. 45 ms)
12. Rechteckgenerator mit den Frequenzen 1 kHz, 10 kHz, 100 kHz und 500 kHz
13. Messung der Pulsdauer eines Servo-Steuersignals

In Arbeit: Ein neuer Multitester

Leider ist das Minimod-Modul nicht mehr bei Elektor erhältlich. Doch wir fanden die Idee von Thomas Ücok so gut, dass wir uns zu einer Neuentwicklung des „Multitesters“ entschlossen haben, die nicht mehr auf das Minimod-Board angewiesen ist.
Im großen Bild sieht man einen ersten Prototyp. Die Testergebnisse werden auf einem alphanumerischen Mini-LCD mit 8 * 2 Zeichen angezeigt. Die Betriebsspannung von 5 V erhält der Tester über eine USB-Buchse, womit sowohl ein PC oder Notebook als auch ein kleines USB-Netzteil zur Stromversorgung in Frage kommen. Zwei Taster genügen, um alle Funktionen anwählen und bedienen zu können.
Zu einem Messgerät, das im harten Laboralltag praxistauglich sein soll, gehört natürlich auch ein robustes Gehäuse. Zum ersten Mal haben wir für ein Projekt ein spezielles Gehäuse designen lassen, das mit einem 3D-Drucker gedruckt werden kann. **Bild 2** zeigt einen Prototyp dieses Gehäuses.

Bild 1.
Der Multitester von Thomas Ücok basiert auf dem Minimod-Modul.

Schaltung

In **Bild 3** sieht man den gegenwärtigen Stand der Schaltung, bei der einige gute Ideen von Thomas Ücok umgesetzt wurden. Herzstück ist ein ATmega328P, der klassisch beschaltet ist. Hierzu gehören ein 16-MHz-Quarz, eine 5-V-Stromversorgung und ein ISP-Header zur Programmierung. Das LCD-Display wird im 4-bit-Modus angesteuert. Mit P1 lässt sich der Kontrast einstellen, mit JP1 wird optional das Backlight des LCD-Displays aktiviert. Die USB-Schnittstelle dient in erster Linie zur Spannungsversorgung der Schaltung. Es kann hierüber aber auch neue Firmware in den Controller eingespielt werden, wenn man den Bootloader „USBaspLoader“ verwendet [2]. Dieses Tool realisiert die USB-Kommunikation „in Software“. Daher braucht man keinen USB/Seriell-Wandlerchip in der Schaltung.
Über I2C ist ein EEPROM-Baustein 24C512 angeschlossen, der zum Ablegen von Konfigurationsdaten genutzt werden kann.
Das einzig Ungewöhnliche an der Schaltung dürfte der Audio-/Video-Switch MAX4584 [3] sein, der ebenfalls an der I2C-Schnittstelle hängt. Dieser Switch legt das Prüf-Signal (das man zum Beispiel über eine Prüfspitze zuführen kann) auf verschiedene Eingänge des Controllers. Denn bei der AVR-Familie gibt es leider keinen Pin, der alle Mess- und Signal-Funktionen beherrscht. Verwendet wird der Pin PC7/ADC7 zur Messung der TTL/CMOS-Pegel, zur Spannungsmessung und zur Impulszählung, zum anderen der Pin PD4/T0 zur Frequenzmessung. Dieser Pin wird auch als UART-Senderausgang verwendet.
Dem Analogeingang vorgeschaltet ist ein Spannungsteiler (R1, R2, R3). Dieser spannt den Eingang mit 1 V vor, das entspricht in den meisten TTL-Familien dem verbotenen Bereich. Dadurch lassen sich bei der TTL-Pegelmessung auch offene Eingänge detektieren. Im Modus „Analog“ misst man dann auch immer 1 V, solange keine andere Spannung anliegt.
Zur Frequenzmessung wird ein Timereingang benötigt (T0). Der Pin PD4/T0 wird aber auch für das LCD-Display verwendet. In der vorlie-

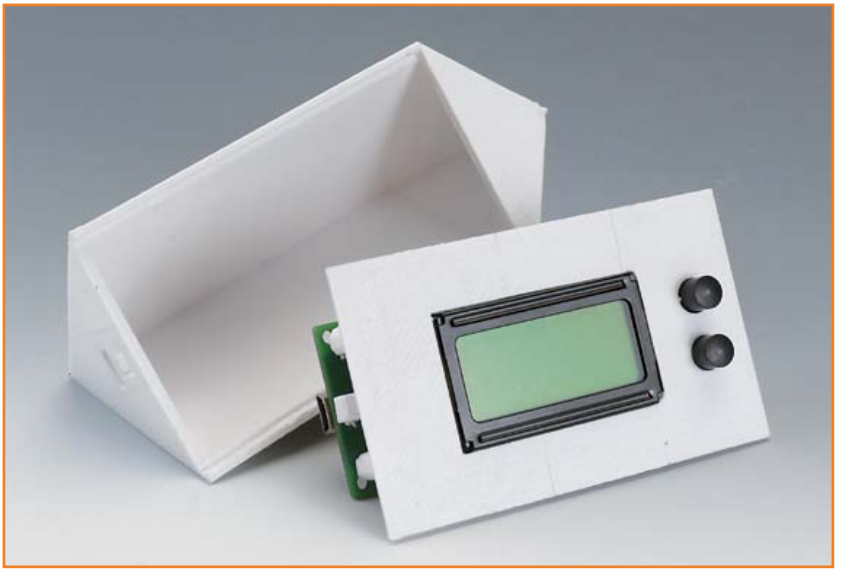


Bild 2.
Ein Gehäuse für den Prototyp wurde mit einem 3D-Drucker angefertigt.

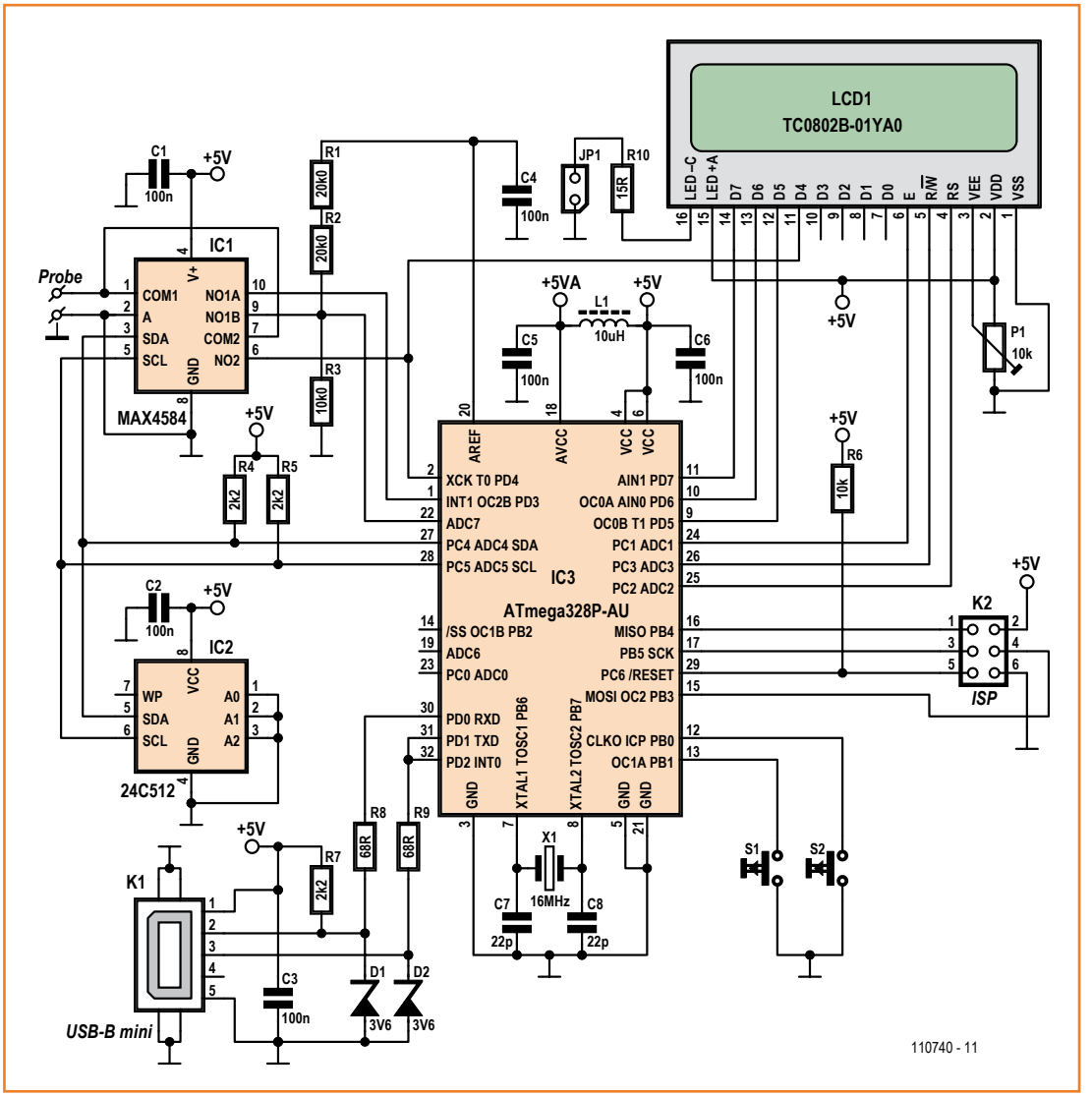


Bild 3.
Basis der Schaltung ist ein ATmega328 (pinkompatibel zum bekannten ATmega88).

genden Schaltung kommt hier ein kleiner Trick zum Einsatz: Während der Messung erfolgt keine LCD-Ausgabe, so dass man den Pin in dieser Zeit zum Timereingang umkonfigurieren kann.

Platine

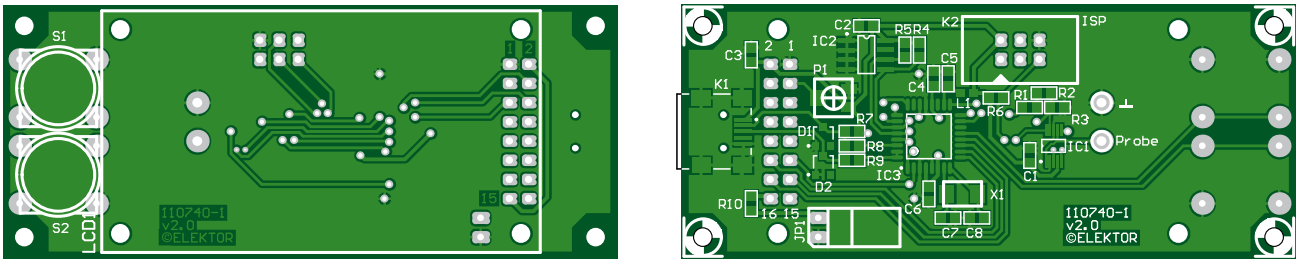
Der Platinen-Prototyp von Elektor ist zweiseitig ausgeführt (siehe **Bild 4**). Die Taster und das Display werden auf der ansonsten unbestückten Seite angeschlossen. Das Display kann man natürlich verdrahten, etwas schöner ist eine Steck-Lösung. Hierzu werden Board-Buchsen von Harwin an die Anschlüsse des Displays gelötet. Die Multitester-Platine wird dagegen mit zwei Stiftleisten bestückt (jeweils 8 Pins, rund). Die Pins dieser Stiftleisten sind an einer Seite stärker (0,7 mm), es ist aber die andere Seite, die in die Platinenlöcher gesteckt wird. JP1 wird liegend platziert, was Platz spart. Zur Befestigung des Displays dienen vier M2-Schrauben und 12 Muttern.

Für das Prüfsignal (Probe) und Masse werden 1,3-mm-Stifte auf der Bauteilseite in die Platine gelötet, diese kann man anschließend in die Horizontale biegen, so dass das Modul auf dieser Seite schön flach bleibt. Den ISP-Steckverbinder muss man ja nur im Bedarfsfall bestücken.

Software

Im gegenwärtigen Prototyp kommt die Software des Autors zum Einsatz, die in Bascom-Basic programmiert ist. Die Software bietet (wie bei jedem Projekt) noch vielfache Möglichkeiten für Erweiterungen und Verbesserungen.

Über den gegenwärtigen Stand des Projekts kön-



nen Sie sich auf unserer eLabs-Website informieren [4]. Dort kann man ein Platinenlayout, eine Stückliste und natürlich die Software herunterladen. Die Bedienmodi des Multitesters, welche die Software bisher beherrscht, sind ebenfalls ausführlich beschrieben.
Natürlich freuen wir uns sehr über Kommentare, Verbesserungsvorschläge, eigene Entwicklungen und weitere Hinweise!

(110740)

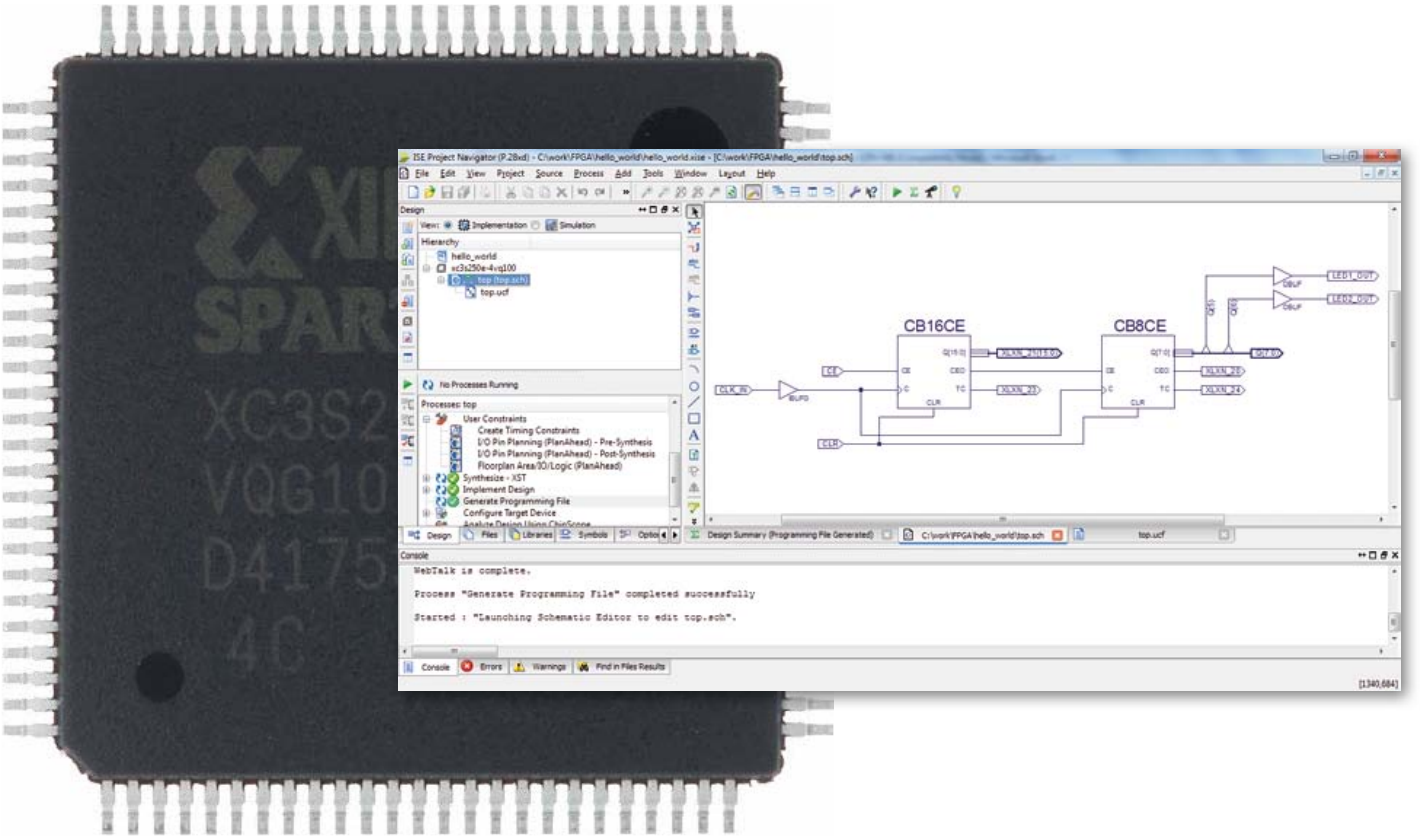
Weblinks

[1] www.elektor.de/090773
[2] www.obdev.at/products/vusb/usbasploader.html
[3] www.maximintegrated.com/datasheet/index.mvp/id/2080
[4] www.elektor-labs.com/9120802389

Bild 4.
Die meisten Bauteile sitzen auf einer Seite des Platinen-Prototyps, auf der anderen werden das Display und die Taster angeschlossen.

Bauen Sie Ihren Chip! (2)

250.000 Ports lassen eine LED blinken



Text:
Clemens Valens
(Elektor-Labor)

Projekt:
Raymond Vermeulen
(Elektor-Labor)

In der ersten Folge haben wir gezeigt, was ein FPGA ist und was er kann. Gleichzeitig haben wir unser FPGA-Board vorgestellt, mit dem Newcomer und Fortgeschrittene gleichermaßen Erfahrungen sammeln können. Weil FPGAs hoch komplexe Komponenten sind, die mit nicht minder komplexen Tools programmiert werden müssen, wollen wir die ersten Schritte langsam gehen. Wir werden sehen, dass diese Schritte überschaubar und nicht allzu schwierig sind. Erst ein vertieftes Eindringen in die Welt der FPGAs erfordert ein überdurchschnittliches Maß an Verständnis und Zeit.

1 Erst Software installieren,...

Damit Sie in diesem Monat aus dem Stand heraus starten können, hatten wir Ihnen empfohlen, in der Zwischenzeit die kostenlose DVD von Xilinx anzufordern, auf der die Tools zusammengestellt sind. Die DVD ist zweigeteilt, die Tools können sowohl unter Windows als auch Linux installiert werden.

Den DVD-Inhalt können Sie auch downloaden, doch ähnlich wie FPGA-Hersteller ihre Produkte mit einer Vielzahl von Anschlüssen versehen, sind auf der DVD Unmengen Bits und Bytes gespeichert. Die Software hat einen Umfang von rund 6 GB. Für den Download ist wichtig, dass der *Xilinx Download Manager* (**Bild 1**) funktionsfähig in Ihren Browser eingebunden ist. Mit Firefox gelang uns dies nicht auf Anhieb, wahrscheinlich weil wir übersehen hatten, bestimmte Optionen zu aktivieren. Mit dem Internet-Explorer kamen wir sofort ans Ziel. Der Download Mana-

ger kann abgebrochene Downloads fortsetzen, ohne noch einmal von vorn beginnen zu müssen. Nachteilig empfanden wir die relativ niedrige Download-Geschwindigkeit.

An unserem FPGA-Laborplatz ist kein Linux-PC verfügbar, so dass wir uns nachfolgend auf die Windows-Version beziehen. Die Installation unter Linux dürfte jedoch sehr ähnlich verlaufen. Legen Sie die DVD in das Laufwerk ein oder entpacken Sie die heruntergeladene TAR-Datei, starten Sie anschließend das Programm *xsetup.exe* (falls es nicht selbsttätig startet). Klicken Sie auf *Next*, akzeptieren Sie die beiden Lizenzvereinbarungen und wählen Sie *ISE WebPack* (**Bild 2**). Klicken Sie wieder auf *Next* und lassen Sie alle Installationskomponenten angehakt. Eigentlich genügt es, jetzt nur *Acquire or Manage a License Key* zu aktivieren, doch die übrigen Komponenten können später wichtig sein. Klicken Sie auf *Next*, wählen Sie den Ort der Installation, klicken Sie noch einmal auf *Next* und betrachten Sie die Zusammenfassung, die nach einem Klick auf *Install* erscheint. Abhängig von Ihrem PC haben Sie nun mehr oder weniger Zeit zum Kaffeetrinken, auf unserem PC dauerte die Installation ungefähr eine dreiviertel Stunde.

Nachdem die Installation beendet ist, fordern Sie eine Lizenz bei Xilinx an. Klicken Sie im *License Configuration Manager* (**Bild 3**) die Auswahl *Get Free ISE WebPack License* an und klicken Sie auf *Next*. Jetzt erscheint ein Fenster, in das die Daten einzutragen sind, die für die Lizenz benötigt werden. Nach einem Klick auf *Connect Now* werden Sie mit der Website von Xilinx verbunden. Dort müssen Sie sich registrieren lassen, falls dies nicht schon früher geschehen ist. Eine neue Registrierung führt Sie in einen Bereich des Web-Auftritts, aus der die Lizenzseite nur umständlich erreichbar ist. Der kurze Weg besteht darin, im *License Configuration Manager* auf den Tab *Acquire a License* zu klicken. Dann erscheint wieder die erste Seite, so dass die Prozedur sofort starten kann.

Bevor Sie auf die Seite für die Lizenzanforderung gelangen, müssen Sie sich durch mehrere Formulare hindurcharbeiten, dort müssen Sie Ihre Daten eintragen. Wählen Sie anschließend *ISE Design Suite: WebPACK License* und klicken Sie auf die Schaltfläche *Generate Node-Locked License*. Es erscheint eine Zusammenfassung, und nach Anklicken von *Next* wird Ihnen mitgeteilt, dass die Lizenzdatei an die bei der Registrierung hinterlegte Emailadresse geschickt wurde. Speichern Sie die Datei so auf Ihrer Festplatte, dass sie schnell auffindbar ist, und gehen Sie zurück

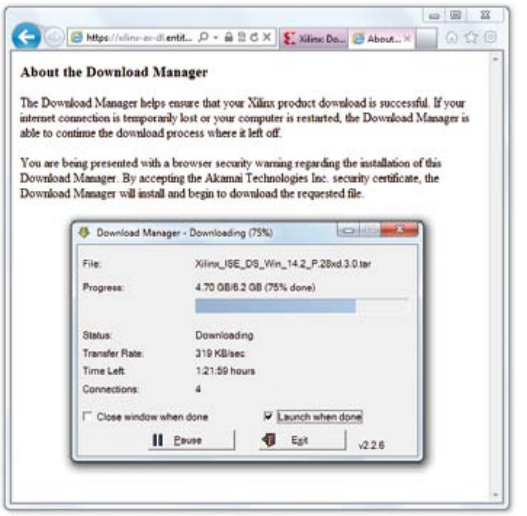


Bild 1.
Der Xilinx Download Manager. Er ist nur betriebsbereit, wenn dieses Fenster erscheint.

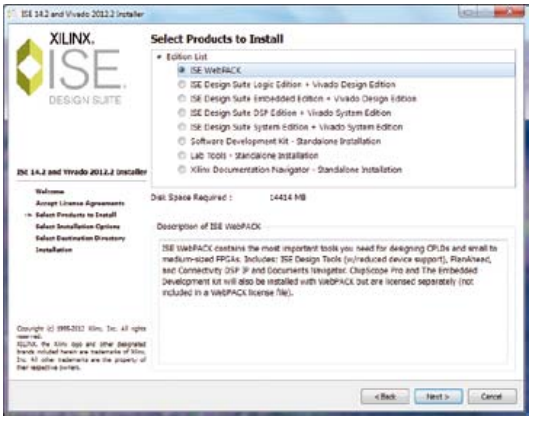


Bild 2.
Mit dem ISE Installer wird das ISE WebPack installiert.

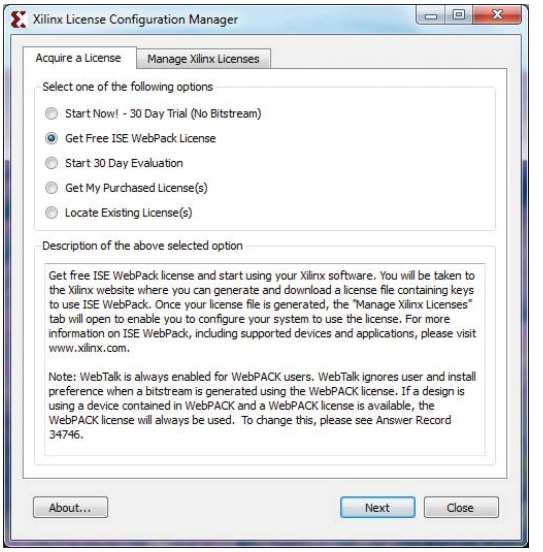


Bild 3.
Das ISE WebPack kann mit einer Lizenz von Xilinx genutzt werden.

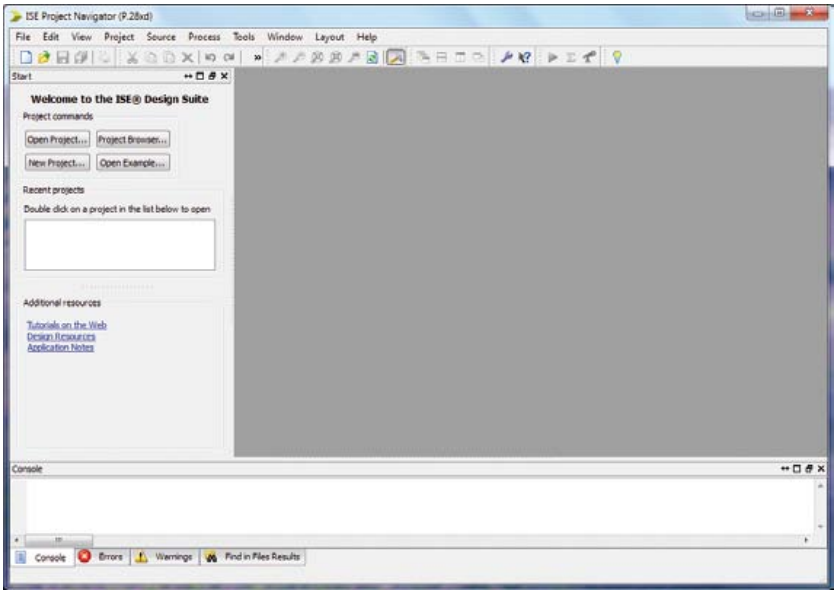


Bild 4.
Der ISE Projektmanager ist bereit für ein neues Projekt.

zum *License Configuration Manager*. Klicken Sie diesmal auf den Tab *Manage Xilinx Licenses* und anschließend auf die Schaltfläche *Copy License*. Navigieren Sie zur Lizenzdatei und klicken Sie auf *Open*. Danach muss die Meldung „*License installation was successful*" erscheinen. Wenn dem so ist, können Sie den Manager mit einem Klick auf *Close* schließen. Das Installationsfenster schließt sich, wenn Sie auf *Finish* klicken.

2 ...bevor es richtig losgeht...

Wenn die ISE Tools installiert sind und die Lizenz akzeptiert wurde, kann es losgehen. In der Programmliste ist die Option *Xilinx Design Tools* hinzugekommen. Öffnen Sie die Programmliste, öffnen Sie anschließend die *ISE Design Suite 14.2* (die Versionsnummer kann abweichen), öffnen Sie danach die *ISE Design Tools* und wählen Sie den *Project Navigator* aus. Im Menü sind eine 32-bit- und eine 64-bit-Version aufgeführt, wählen Sie die Version, die ihrem Betriebssystem entspricht. Jetzt erscheint das Fenster, das in **Bild 4** dargestellt ist. Klicken Sie auf *New Project*, um den *New Project Wizard* zu öffnen. Geben Sie die Daten anhand des Beispiels in **Bild 5** ein. Im Feld *Description* können Sie ihr neues Projekt kurz beschreiben. Unsere erste FPGA-Anwendung konzipieren wir in Form eines Schaltschemas, denn diese Form ist uns vertraut. Das Programmieren in einer Sprache wie VHDL oder Verilog verschieben wir auf spätere Zeit. Wählen Sie als *Top-level source type* die Option *Schematic*. Weil FPGA-Anwendungen meistens ziemlich umfangreich sind, werden sie fast immer hierarchisch strukturiert. In der Hierarchie gibt es verschiedene Ebenen, eine tiefer liegenden Ebene enthält mehr Details als die Ebene darüber. Die oberste Ebene, *top-level* genannt, ist nichts weiter als ein stark vereinfachtes Blockschema. Jeder Block wird in der darunter liegenden Ebene in detailliertere Blöcke aufgeschlüsselt, so lange, bis die Funktionen in jedem Detail definiert sind. Für unsere erste Anwendung, das Blinken einer LED auf dem FPGA-Board, ist das Arbeiten mit mehreren Ebenen noch nicht nötig, alles geschieht auf der Ebene *top-level*.

Klicken Sie auf *Next*, so dass das nächste Fenster des Wizards erscheint (**Bild 6**). In den meisten Feldern können die vorgegebenen Werte stehen bleiben, dies trifft jedoch nicht für die Felder *Family*, *Device*, *Package* und *Speed* zu. Die Daten in diesen Feldern hängen vom verwendeten Chip auf dem FPGA-Board ab, es ist ein Chip aus der Familie Spartan 3E mit der Typenbezeichnung XC3S250E im Gehäuse VQ100. Als *Speed* wählen Sie die Standard-Version *-4*, die schnelle Version hat als *Speed* die *-5*. Die Angabe ist auch auf dem Chip zu finden, dort ist unten *4C* aufgedruckt. Klicken Sie auf *Next*, um die Zusammenfassung der Projektdaten zu betrachten. Die Daten wollen wir hier nicht erläutern, an dieser Stelle sind keine Änderungen mehr möglich. Klicken Sie deshalb auf *Finish*.

Sie haben nun ein noch leeres Projekt vor sich. Nehmen Sie sich die Zeit für einen kurzen Rundumblick, bevor Sie über *Project → New Source* den *New Source Wizard* öffnen (**Bild 7**). Wählen Sie *Schematic* als *Source Type*, als Dateinamen nehmen Sie *top*, dann vergewissern Sie sich, dass *Add to project* angehakt ist. Anschließend klicken Sie auf *Next*, so dass eine Zusammenfassung erscheint. Das Fenster schließt sich, wenn Sie auf *Finish* klicken.

Nach einer kurzen Umschaltpause hat sich die Ansicht verändert. Jetzt sehen Sie ein leeres Fenster, in das Sie Ihr Schaltschema eingeben können. Links vom Hauptfeld ist eine Liste eingeblendet, in der diverse Optionen stehen. Unter der Liste sind bereits einige Tabs sichtbar, es gibt jedoch noch viele weitere Tabs. Um alle Tabs ins Bild zu bringen, klicken Sie mehrfach auf den Pfeil neben dem Tab *Options*. Der Tab *Design* war bereits vor dem Start des *New Source Wizard* vorhanden. Sie können sich jetzt davon überzeugen, dass das Schaltschema *top* zum Projekt hinzugefügt wurde.

3 ...mit dem Konstruieren eines Schemas

Das erste Übungsziel, das wir uns setzen, ist das

Blinken einer LED auf dem FPGA-Board. Damit das Blinken für das menschliche Auge erkennbar ist, darf die Blinkfrequenz höchstens etwa 10 Hz betragen. Der FPGA wird vom Mikrocontroller mit 8 MHz getaktet. Aus dieser Frequenz können wir eine Blinkfrequenz ableiten, indem wir die Taktfrequenz wiederholt durch 2 teilen. Das wiederholte Teilen durch diesen Faktor lässt sich mit einer Flipflop-Kette leicht bewerkstelligen. Wenn die Kette aus 22 Flipflops besteht, beträgt die Frequenz am Kettenende 8 MHz / 2^{22} = 1,9 Hz. Die LED würde knapp zweimal in der Sekunde blinken. Im Schema können Sie entweder 22 hintereinander geschaltete Flipflops oder einen bis 2^{22} zählenden Zähler platzieren. Klicken Sie auf den Tab *Symbols*, neben dem Tab *Options* in der linken unteren Ecke von ISE. Wählen Sie *Counter* in der Liste *Categories*. Unter *Symbols* sind nun alle Zähler aufgelistet, die in das Schema übernommen werden können. Ein 22-bit-Zähler ist nicht darunter, er muss aus zwei Zählern niedrigerer Ordnung zusammen gestellt werden. Der erste Zähler in der Liste heißt *cb16ce*, was auf einen 16-bit-Zähler hindeutet. Mit ihm wollen wir beginnen.

Symbole platzieren

Wählen Sie durch Anklicken den Zähler *cb16ce* aus und bewegen Sie den Mauszeiger zum Schema. An den Mauszeiger ist jetzt der Umriss eines Zählersymbols geheftet. Klicken Sie auf den Ort im Schema, an dem Sie den Zähler platzieren wollen. Wie vermutet ist der Zähler ein 16-bit-Typ mit diversen Ein- und Ausgängen. Um die Funktionen herauszufinden, klicken Sie mit der rechten Maustaste auf das Symbol. Im nun erscheinenden Menü wählen Sie *Symbol*, und anschließend *Symbol Info*. Die ISE-Hilfefunktion wird auf der zum Symbol gehörenden Seite geöffnet. Dort steht unter anderem, dass CLR auf Low und CE auf High liegen müssen, damit der Zähler zählt. Ferner ist dort angegeben, dass mit dem Signal an Ausgang CEO ein nachfolgender Zähler getaktet werden kann. Da der Zähler hier bis 2^{16} zählen soll, können die Ausgänge Q0...Q15 unbenutzt bleiben. In gleicher Weise fügen Sie den *cb8ce* als zweiten Zähler in das Schema ein. Bereits in der letzten Folge unserer Artikelreihe war zu lesen, dass zwischen den logischen Blöcken eines FPGA und der Außenwelt so genannte I/O-Blöcke liegen. Jedes Signal, das dem FPGA über seine Anschlüsse zugeführt wird oder ihn verlässt, muss einen I/O-Block passieren. Hier gilt dies für das LED- und Taktsignal sowie für die Signale CE und CLR, die von außen gesteuert werden sollen. Für das Koppeln mit I/O-Blöcken hält ISE spezielle Symbole bereit, sie sind in der Kategorie *I/O* zu finden.
Taktsignale haben spezielle Eigenschaften, deshalb

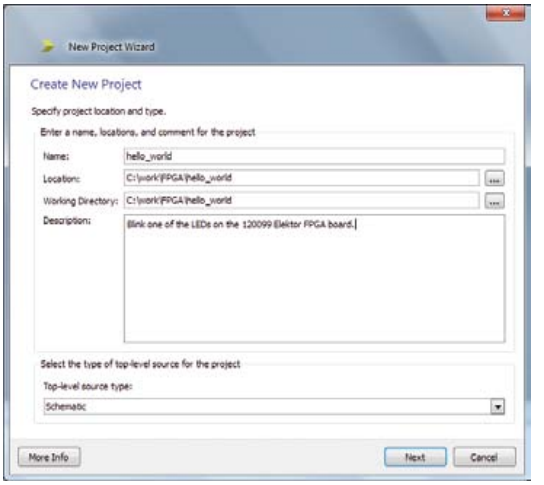


Bild 5.
Neue Projekte werden mit dem *New Project Wizard* gestartet.

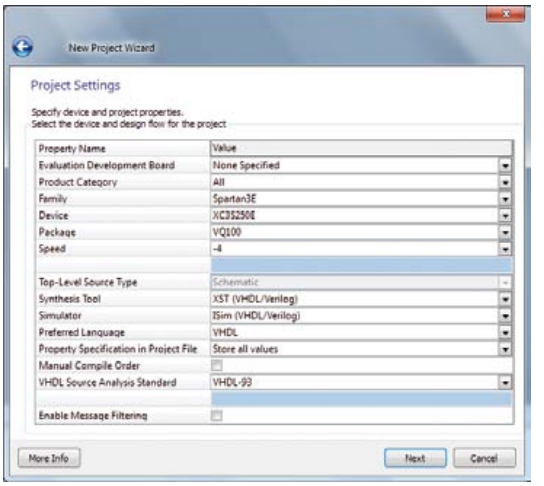


Bild 6.
In der Liste müssen die korrekten FPGA-Daten eingestellt sein.

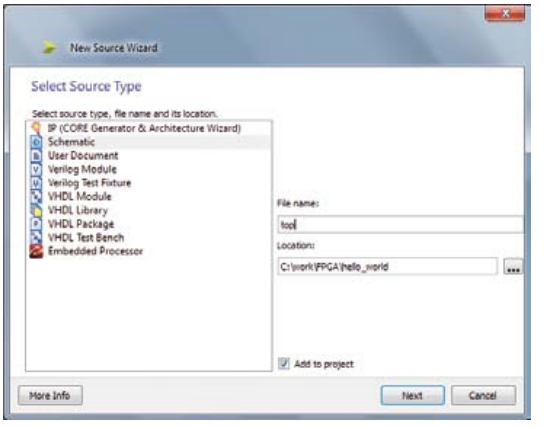


Bild 7.
Ein „Wizard" hilft beim Hinzufügen von Dateien zu einem Projekt.

Mikrocontroller-Firmware: Das Wesentliche

Im Hauptteil dieser Artikelfolge haben wir den Weg von der Anwendungsidee zur Bitstream-Datei beschrieben, die dem FPGA Leben einhaucht. Doch was geschieht hinter den Kulissen, damit die Datei in den FPGA übertragen wird? Das Laden der Datei ist Aufgabe des Mikrocontrollers, der sich auf dem FPGA-Board befindet. Nachfolgend wollen wir einen Blick auf die Mikrocontroller-Firmware werfen, die im Elektor-Labor entwickelt wurde. Interessierte Leser können die Source- und Hex-Dateien von der Elektor-Website [1] herunterladen. Die Firmware gehört in allen Teilen der Klasse der Open-Source-Software an. Wir möchten erfahrene Leser anspornen, die Firmware weiter zu entwickeln.

Hinter Open-Source-Projekten steht unter anderem die Philosophie, dass es wenig sinnvoll ist, das Rad ständig neu zu erfinden. Für viele Zwecke und Anwendungen existieren bereits Open-Source-Programme, oft stehen umfangreiche Bibliotheken zur freien Verfügung. Davon haben wir auch hier Gebrauch gemacht. Unser Dank gilt unter anderem Dean Camera für die Bibliothek *LUFA* [3] sowie ChaN für *Petit FatFs* [4]. Ohne diese Open-Source-Komponenten wäre unser Projekt spartanischer ausgefallen, der Aufwand an Zeit und Mühe wäre ungleich höher gewesen. Unsere *LUFA*-Bibliothek hatte die Versionsnummer 120219, die Version von *Petit FatFs* war R0.02a.

Das Updaten der Mikrocontroller-Firmware ist ohne ISP möglich. Auf dem Chip befindet sich ein DFU-Bootlader, er ist Teil des Projekts *LUFA*. Unter Windows kann die Firmware mit der Software „Atmel Flip“ [5] in den Mikrocontroller übertragen werden.

Im normalen Betrieb wird der Bootlader beim Start des FPGA-Board übersprungen. Der Bootlader wird nur aktiv, wenn die Taster S1 und S2 wie folgt bedient werden: Zuerst S1 und dann S2 drücken, anschließend zuerst S1 und dann S2 loslassen. Der verwendete Treiber ist Bestandteil von „Atmel Flip“.

Doch zurück zur Anwendung: Nach dem Start wird zuerst der FPGA programmiert, dann werden die Prozeduren ausgeführt, die auf USB aufsetzen. Diese Reihenfolge ist notwendig, weil *LUFA* die USB-bezogenen Prozeduren über Interrupts abarbeitet. Ferner können zeitkritische Prozeduren erledigt werden, bevor die Interrupts aktiv sind. Die Abgrenzung half uns, die genannten Vorgänge voneinander getrennt zu halten. Unterstützend war die Tatsache, dass die micro-SD-Karte nicht gleichzeitig auf dem PC und lokal auf dem Board eingebunden sein kann.

In der ersten Phase wird die Hardware initialisiert. Die I/Os und die *m{2.0}*-Pins werden gesetzt, um den FPGA im Modus *SerialSlave* zu konfigurieren. Auch *Hswap* wird auf High gesetzt, damit die Pullups der I/Os während der Konfiguration deaktiviert sind. Die micro-SD-Karte wird identifiziert und für den Betrieb vorbereitet, die Kommunikation mit unterschiedlichen Typen läuft nicht immer gleich ab. *Petit-FatFs*-Funktionen binden die Partition auf der micro-SD-Karte lokal ein. Ist eine Datei *config.bin* vorhanden, wird diese Datei geöffnet. Leitung *Prog_B* wird für 1 ms auf Low gesetzt, um den Reset des FPGA herbei zu führen. Wenn anschließend Leitung *Init_B* auf High geht, kann der Datentransport starten. Datenblöcke der micro-SD-Karte werden gelesen und über den USART-Bus in den FPGA geschrieben, der Bus wird über Funktionen der *LUFA*-Bibliothek gesteuert. Während des Datentransports findet nach jedem übertragenen Byte eine Prüfung statt, sie stellt fest, ob Leitung *Init_B* auf Low gegangen ist. Der Sprung nach Low wäre ein Indiz für einen aufgetretenen Fehler. Nach dem Übertragen der Datei wird geprüft, ob Signal *Done* auf High ist.

High ist das Zeichen dafür, dass die Konfiguration erfolgreich verlief. Zum Schluss wird die micro-SD-Karte vom System abgemeldet. Die peripheren SPI- und USART-Komponenten werden deaktiviert, und die benutzten Pins werden zurück in den hochohmigen Zustand (*Hi-Z*) versetzt.
Petit FatFs macht den lokalen Einsatz von FAT16 oder FAT32 ohne überlagertes Betriebssystem möglich. Diese Bibliothek ist unter *..\FPGA_board_Config_v1_0\local_fat* im ZIP-Archiv zu finden, das von [1] aus dem Web herunter geladen werden kann. In *pff.h* werden *Defines* verwendet, die das Dateisystem FAT32 unterstützen. Nur *read* ist aktiviert, da auf die micro-SD-Karte lokal nicht geschrieben wird. *MMC.c* ist Bestandteil des Anwendungsbeispiels von AVR, das von [3] herunter geladen werden kann. Die vorhandene Low-Level-Funktionalität wurde zum größten Teil durch *LUFA*-Bibliotheksfunktionen ersetzt.

In der zweiten Phase wird eine Endlosschleife gestartet, die den Datenaustausch über USB mit einem virtuellen COM-Port und der micro-SD-Karte als *Mass Storage Device* abwickelt. Dazu wird die Hardware neu initialisiert, auch die micro-SD-Karte wird wieder identifiziert und diesmal für den SPI-Datenaustausch vorbereitet. Aus der ersten Phase wird das Ergebnis der Fehlerbehandlung verwendet, um den zugehörigen String über den virtuellen COM-Port zum PC zu senden. Das Betriebssystem behandelt die micro-SD-Karte nun wie einen USB-Massenspeicher. Das bedeutet, dass SCSI-Kommandos über USB übertragen werden und das Betriebssystem nicht mehr lokal läuft, sondern entfernt auf dem PC. Funktionen, die eine micro-SD-Karte in dieser Weise behandeln, sind in *LUFA* nicht Standard. Wir haben sie nach einiger Recherche im Internet gefunden und so angepasst, dass sie mit *LUFA* Version120219 kompatibel sind. Bestimmte, nun überflüssige Teile stehen noch in der Datei *FPGA_board_Config.h*, sie können eventuell später gelöscht werden.

Die *LUFA*-Bibliothek dient hauptsächlich dazu, die USB-bezogenen Funktionen abzuhandeln. Doch sie enthält auch diverse Low-Level-Funktionen zum Steuern diverser peripherer Komponenten. Zum *LUFA*-Paket gehören Hardware-Profile für die Kommunikation mit verschiedenen Board-Typen. Dort haben wir ein Profil hinzugefügt, das unser FPGA-Experimentierboard unterstützt. Weil die Zeit drängte, mussten wir die Kompatibilität zugunsten der schnelleren und einfacheren Programmierung aufgeben. Im *LUFA*-Paket sind auch Demo-Projekte enthalten, unsere Firmware basiert auf dem Demo-Projekt *VirtualMassStorage*. Weiterführende Informationen und die Details von *LUFA* können der Dokumentation entnommen werden, die von [3] heruntergeladen werden kann.

Zum Debuggen dient ein virtueller COM-Port, der Fehlermeldungen zum PC sendet. Während unserer Firmware-Entwicklung hat sich gezeigt, dass dieses Verfahren ausgesprochen komfortabel ist. Mit einigen Anpassungen kann auch eine Kommunikation in umgekehrter Richtung stattfinden.

Ein Tipp zum Schluss:
Für einfache, funktionsartige Notationen sind *Defines* einsetzbar, ohne Funktions-Overhead. Wir haben davon Gebrauch gemacht, um Signale unkompliziert auf FPGA-Anschlüsse zu setzen oder von dort zu lesen. Die Funktionen bleiben stets übersichtlich. Zum Beispiel steht in der Datei *FPGA_board_Config.h*:

```
#define Release_FPGA_reset()       DDRB&=~(1<<DDB4)
#define Init_B_status()            ((PINC >> PC6) & 0x01)
```

In einer Funktion von *FPGA_board_Config.c* wird dies so angewendet:

```
while (!Init_B_status()) {;;}
```

oder:

```
Release_FPGA_reset();      // release the Prog_b line
[/code]
```

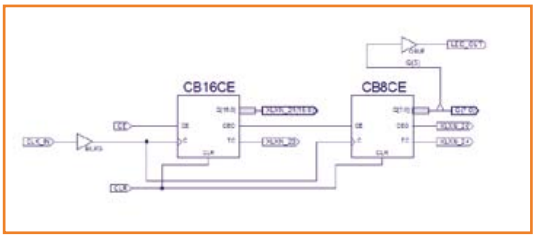


Bild 8.
Das Schema des FPGA-Blinkgebers.

muss vor den Takteingang das besondere Symbol *ibufg* gesetzt werden. Dieser Puffer bewirkt, dass das Taktsignal unmittelbar an einer speziellen Taktleitung des FPGA liegt. Die übrigen Eingänge werden ebenfalls über Puffer verbunden, diese Puffer heißen *ibuf*. Die Hilfeseite zu *ibuf* empfiehlt, diese Symbole nicht von Hand zu platzieren. Das überlassen wir ISE, immer dort, wo es möglich ist. Wir müssen jedoch so genannte I/O-Marker einfügen, die ähnliche Funktionen wie Label haben. Auch in unserem sehr simplen Schema setzen wir keine *ibuf*-Symbole, sondern nur I/O-Marker. Ausgänge müssen grundsätzlich über Puffer nach außen geführt werden. Es gibt mehrere Wege, den Typ eines Ausgangs festzulegen. Für die LED-Steuerung genügt der einfache Ausgangspuffer mit dem Namen *obuf*. Im Schema bedeutet dies, dass ein Puffer *obuf* seinen Platz am Ausgang Q des Zählers *cb8ce* erhält. Einen *ibufg* setzen wir vor den Eingang des Zählers *cb16ce*.

5 Leitungen legen

Verbindungen zwischen Komponenten können Sie verlegen, wenn Sie auf die Schaltfläche *Add Wire* klicken, die sich oben in der Symbolleiste zwischen dem Schema und dem Tab *Options* befindet. Wenn der Mauszeiger auf die Schaltfläche fährt, erscheint eine Kurzbeschreibung der Funktion. Nach dem Anklicken ändert der Mauszeiger seine Gestalt in ein Fadenkreuz um. Mit dem Fadenkreuz können Sie die zu verbindenden Anschlüsse der Komponenten anklicken. Der Modus ist jetzt *Autorouter* (siehe Tab *Options*), was besagt, dass Sie nur den Anfangs- und Endpunkt einer Leitung anklicken müssen. Die Leitung zwischen den Punkten verlegt ISE selbst, auf mehr oder weniger elegante Weise. Mit der Maus können Sie Leitungssegmente verlagern, falls Ihnen ein Leitungswirrwarr zu unübersichtlich erscheint. Verbinden Sie die Anschlüsse C der Zähler miteinander, das Gleiche gilt für die Anschlüsse CLR. Anschluss CEO des *cb16ce* verbinden Sie mit CE des *cb8ce*. Den Eingang von *obuf* lassen Sie vorläufig offen.

6 Marker und Label

Der nächste Schritt ist das Hinzufügen von I/O-Labeln mithilfe der Schaltfläche *Add I/O Marker*. Wenn die Option *Add an automatic marker* aktiviert ist, können Sie unter bestimmten Voraussetzungen unmittelbar auf einen Anschluss klicken, um ein Label anzubringen. Sollte das nicht möglich sein, fügen Sie zuerst mit *Add Wire* ein Leitungs- oder Bussegment hinzu. Mit einem Doppelklick schließen Sie ein Segment ab. Fügen Sie an den Ausgang Q[7:0] von *cb8ce* ein Bussegment an, es wird später gebraucht.

Den I/O-Labeln hat ISE selbsttätig erzeugte Namen zugewiesen, sie können über die Schaltfläche *Add Net Name* geändert werden. Tragen Sie zuerst den Namen des *Net* ein (Tab *Options*), bevor Sie neue Namen vergeben. Benennen Sie alle Leitungen um, die am FPGA außen angeschlossen sind (CLR, CE, CLK_IN und LED_OUT). Geben Sie dem Bus an Ausgang Q[7:0] des *cb8ce* den Namen Q(7:0). Eckige Klammern werden von ISE in runde Klammern gewandelt. Q(7:0) bedeutet, dass es sich um einen Bus mit den Signalen Q0...Q7 handelt.

7 Bussignal anschließen

Damit die Frequenz des an Zähler *cb16ce* liegenden Eingangssignals durch 2^{22} geteilt wird, muss der Eingang von *obuf* mit Ausgang Q5 des Zählers *cb8ce* verbunden werden. Eine Methode, die ISE akzeptiert, ist folgende: Klicken Sie auf die Schaltfläche *Add Bus Tab*, und unter *Options* wählen Sie *Bottom* oder auch eine andere Orientierung. Jetzt können Sie im Schema ein trichterförmiges Symbol mit der Breitseite in Richtung Bussegment Q(7:0) platzieren. Verbinden Sie den Ausgang über ein Leitungssegment mit dem Eingang von *obuf*. Weisen Sie dieser Verbindung den Namen Q(5) zu. Ihr Schema muss jetzt ungefähr so aussehen, wie es in **Bild 8** dargestellt ist.
Bevor wir den FPGA-Anschlüssen Label zuweisen, empfiehlt es sich, die Konstruktion auf Funktionsfähigkeit zu testen. Normalerweise würden wir unser Schema simulieren, doch unser FPGA-Blinkgeber ist so simpel, dass wir den Schritt überspringen können. Wir wollen jedoch versuchen, das Schema zu synthetisieren. Damit ist gemeint, dass ISE das Schema in die Ports und Flipflops umsetzt, die in seinen Bibliotheken enthalten sind. Der Prozess ist mit dem Compilieren eines Programms vergleichbar, das in einer höheren Sprache geschrieben wurde.

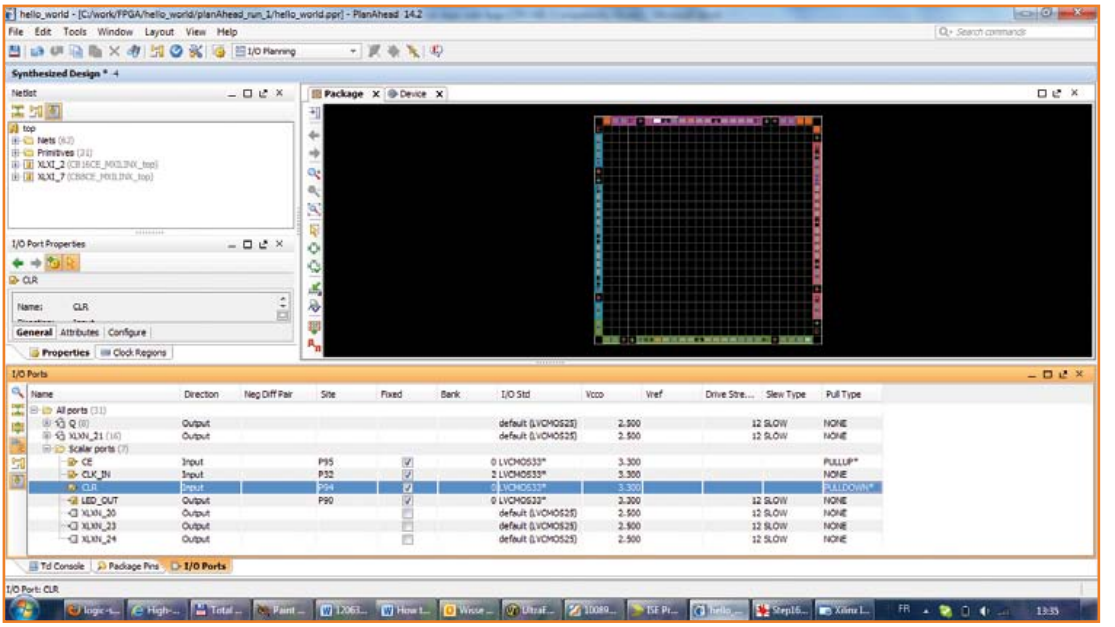


Bild 9.
Mit diesem Tool werden interne FPGA-Signale an FPGA-Anschlüsse gelegt. Ferner ist das Bearbeiten der UDF-Datei in einem Texteditor möglich.

Synthetisieren

Klicken Sie im Tab *Design* doppelt auf *Synthesize – XST*. Im Fenster *Console* erscheinen diverse Meldungen, und ein rotierendes Symbol signalisiert, dass im Hintergrund etwas geschieht. Nachdem der Prozess erfolgreich abgeschlossen ist, erscheint links von *Synthesize – XST* ein grüner Haken. Wenn Sie die Meldungen im Fenster *Console* betrachten, erfahren Sie, dass das Schema mit 24 Flipflops realisiert wurde.

Anschlüsse zuweisen

Im nächsten Schritt wollen wir den FPGA-Anschlüssen Label zuweisen. Starten Sie über den Tab *Design* das Tool *PlanAhead*. Dazu müssen Sie das Menü *User Constraints* ausklappen und mit einem Doppelklick *I/O Pin Planning (PlanAhead) Post-Synthesis* wählen, denn unser Projekt wurde bereits synthetisiert. ISE fragt an, ob Sie dem Projekt ein *User Constraint File* (UCF) hinzufügen wollen. Das beantworten Sie mit „OK“, was „Ja“ bedeutet. Abhängig von den Eigenschaften Ihres PC kann es einen Moment dauern, bis *PlanAhead* startklar ist.

Der Tab *I/O Ports* von *PlanAhead* (unten) listet die Label aus unserem Schema auf. Wenn Sie den Baum *Scalar ports* ausklappen, erscheinen die Label der zu verbindenden Anschlüsse. Nehmen Sie Teil 1 unserer Artikelfolge zur Hand und schlagen Sie die Bilder 7 und 8 auf. Auf dem FPGA-Board liegen die LEDs an den Pins 90 und 91. Klicken Sie in die Zeile LED_OUT und anschließend in die Spalte *Site*. In der ausklappenden Liste wählen Sie P90 oder, wenn es Ihnen lieber ist, P91. Klicken Sie in die Spalte *I/O Std* und wählen Sie LVCMOS33, dies ist der Standard auf dem FPGA-Board. Die übrigen Eintragungen gibt das Tool vor. Der nächste Anschluss ist CLK_IN. Auf dem FPGA-Board erhält der FPGA das Taktsignal grundsätzlich über Pin 32, so dass Sie P32 in der Spalte *Site* wählen müssen. Auch hier setzen Sie den I/O-Standard auf LVMOS33. Für CE und CLR können Sie unter den Anschlüssen, die zu den Steckverbindern K4 oder K5 führen, einen beliebigen Anschluss wählen. Wir haben uns für P94 (CLR) und P95 (CE) entschieden. Ais I/O-Standard müssen Sie wieder LVCMOS33 wählen. Für die verwendeten Anschlüsse aktivieren Sie die internen Pullup- und Pulldown-Widerstände, indem Sie in der Spalte *Pull Type* auf den zugehörigen Eintrag klicken (siehe **Bild 9**). Klicken Sie auf die Schaltfläche *Save Constraints*, um die Konfiguration zu speichern, und kehren Sie in ISE zurück zum Tab *Design*.

Implementieren

Nun wollen wir unser Projekt implementieren, was bedeutet, dass wir es in logische Blöcke und zugehörige Komponenten übersetzen lassen. Doppelklicken Sie auf *Implement Design* und warten Sie, bis ein grüner Haken erscheint. Eventuelle Fehlermeldungen oder Warnungen können Sie unten in den Tabs *Errors* oder *Warnings* betrachten. Wenn Sie alle Schritte konsequent so durch-

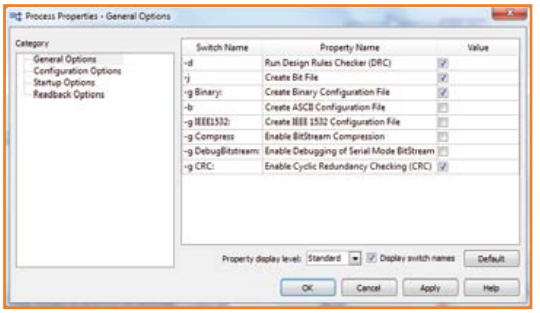


Bild 10.
Bitstream-Optionen, erste Seite.

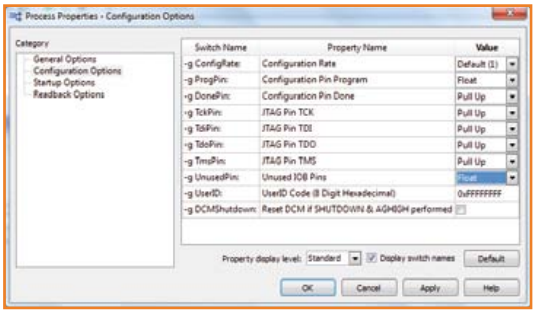


Bild 11.
Bitstream-Optionen, zweite Seite.

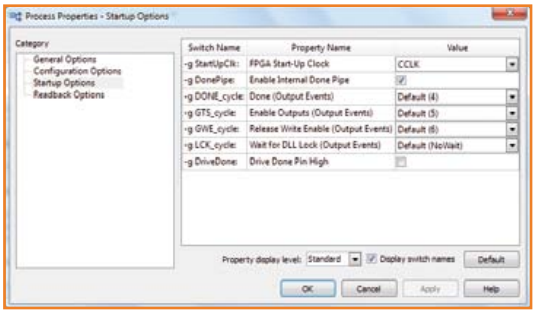


Bild 12.
Bitstream-Optionen, dritte Seite. Die vierte Seite muss nicht angepasst werden.

geführt haben, wie sie vorstehend beschrieben sind, werden keine Fehler- oder Warnmeldungen ausgegeben.

11 Bitstream generieren

Noch sind wir nicht am Ziel angelangt. Was noch fehlt, ist das Erzeugen einer Datei, mit der wir den FPGA programmieren können. Für das Erstellen dieser Datei, hier *Bitstream* genannt, gibt es mehrere Wege. Eine für unser Projekt geeignete Methode ist folgende: Klicken Sie im Tab *Design* mit der rechten Maustaste auf *Generate Programming File*, und in dem aufklappenden Menü wählen Sie *Process Properties*. Wählen Sie in der Liste *Category* die *General Options* und haken Sie *Create Binary Configuration File* an (**Bild 10**). Wählen Sie die nächste Kategorie, *Configuration Options* genannt, und setzen Sie die Werte für *Configuration Pin Program* und *Unused IOB Pins* auf *float* (**Bild 11**). In der Kategorie *Startup Options* haken Sie *Drive Done Pin High* an (**Bild 12**). Die *Readback Options* können Sie unverändert lassen. Klicken Sie auf OK, um die *Process Properties* abzuschließen, und doppelklicken Sie auf *Generate Programming File*, um den Bitstream zu erzeugen. Wenn keine Komplikationen aufgetreten sind, erscheint auch hier nach kurzer Zeit ein grüner Haken. Der Projektordner (siehe *ISE title bar*) enthält nun eine Datei mit dem Namen *top.bin*.

12 FPGA programmieren

Führen Sie die micro-SD-Karte in den Kontaktschlitz auf dem FPGA-Bord ein und schließen Sie das Board über USB an den PC an. Das Betriebssystem fügt nun ein neues *Mass Storage Device* zur Liste der Laufwerke hinzu. Kopieren Sie die Datei *top.bin* auf das neue Laufwerk und benennen Sie dort die Datei in *config.bin* um. Melden Sie das Laufwerk mit *Hardware sicher entfernen* vom Betriebssystem ab und entfernen Sie die USB-Verbindung. Jetzt ist der spannende Moment gekommen, in dem sich zeigt, ob Sie Ihr Ziel erreicht haben: Drücken Sie auf dem FPGA-Board Taster S1. Wenn eine LED blinkt, dürfen Sie sich gratulieren!

Kleine Hausaufgabe

Bleiben Sie in Übung, bis die nächste Folge dieser Artikelreihe erscheint. Konstruieren Sie nach dem oben stehenden Vorbild einen zweiten FPGA-Blinkgeber, der die zweite LED in anderem Rhythmus blinken lässt.

(120630)gd

Weblinks

[1] www.elektor-magazine.de/120630
[2] www.xilinx.com/support/download/index.htm
[3] www.fourwalledcubicle.com/LUFA.php
[4] http://elm-chan.org/fsw/ff/00index_p.html
[5] www.atmel.com/tools/FLIP.aspx

Hypermoderner 7-Tage-Wecker

Von **Michael J. Bauer**
(Australien)

Digitaluhren mit Alarmfunktion sind wahrlich nicht neu, doch diese hier ist etwas ganz Besonderes: Sie ermöglicht eine umfassende Automatisierung im Arbeitsraum, dem Schlafzimmer oder der Küche des Besitzers. Der Wecker kann Lampen, Computer und Peripherie, AV-Geräte und vieles andere mehr ein- und ausschalten. Ganz Ausgebuffte könnten sogar auf die Idee kommen, sich von dem Gerät morgens wecken zu lassen!

Die Idee zu diesem Projekt wurde aus der Frustration geboren: mein Wecker war trotz eines bekannten Herstellernamens und eines stattlichen Preises nur sehr umständlich zu bedienen und ermöglichte auch nicht die Funktionen, die ich mir wünschte. Als ich mit dieser Idee Freunde und Arbeitskollegen konfrontierte, erkannte ich, dass sie alle mit ihren Weckern und Weckerradios nicht wirklich zufrieden waren. Hier eine Liste der meist genannten Beschwerden:

- Man vergisst, vor dem Zubettgehen den Alarm einzuschalten (weil er natürlich morgens ausgeschaltet werden musste).
- Das Uhrenradio ist nicht richtig auf den Sender eingestellt oder die Lautstärke so niedrig gedreht, dass der Alarm nicht gehört wird.
- Umständliches Ritual, um die Zeit oder die Weckzeit zu ändern. Normalerweise muss man dazu dauerhaft einen Knopf für „Stunden“ und einen anderen für „Minuten“ drücken, während die Ziffern im Display stets zu langsam oder zu schnell voranschreiten.
- Die Alarmzeit ist jeden Tag die gleiche, ob Werktag oder Wochenende. Die meisten Menschen müssen aber an Werktagen früher aufstehen. Bei manchen Leuten ändert sich die Weckzeit sogar Tag für Tag.
- Die Zeit, in welcher der Alarm ertönt, ist fest eingestellt (meist zwischen sieben und neun Minuten) und kann nicht geändert werden.
- Es ist furchtbar, den „Schlummer“-Taster (snooze) so schnell wie möglich betätigen zu müssen, um den enervierenden Dauerweckton abzuschalten. Besser: Der Alarm ertönt nur einige Sekunden und setzt dann für ein paar Minuten aus. Das wiederholt sich, bis man den Vorgang irgendwie beendet.
- Das Display ist zu hell in der Nacht oder zu dunkel am Tag.
- Es gibt einen Schalter, um die Displayhellig-

DIY kann nicht mit den Preisen der Massenprodukte mithalten, aber viele Dinge besser machen!

keit einzustellen, aber dieser ist umständlich zu benutzen. Warum passt sich die Helligkeit nicht automatisch dem Umgebungslicht an?
- Ich möchte nicht gerne von einem Radiosender geweckt werden, aber warum produziert der Wecker keinen wohlklingenden Weckton? Warum hat man keine Wahlmöglichkeiten?
- Kleine Kinder spielen gerne mit den Tasten des Weckers und ändern Zeit, Weckzeit oder sonst etwas.
- Die Schlummer-Taste ist zu klein, schlecht platziert oder irgendwie schwierig im Dunklen zu bedienen.
- Der Wecker brüht keinen Kaffee auf.

Fühlen Sie sich an Ihren Wecker erinnert? Eine Recherchetour durch die lokalen Elektroläden zeigte mir, dass es überhaupt keinen Wecker gibt, der die Anforderungen auf meinem Wunschzettel zufriedenstellend erfüllt. Warum habe ich aber Elektronik-Ingenieur gelernt? Ich kann meinen „idealen" Wecker doch selbst entwerfen und bauen!

Design eines komfortablen Weckers

Ich strebte eine Kombination von einfach zu nutzenden, gebräuchlichen und notwendigen Funktionen und einer erweiterten Funktionalität an, ohne dabei die Unzulänglichkeiten kommerzieller Produkte nachzuahmen. Natürlich kann ein Selbstbau-Projekt keinen „Preiskampf" mit kommerziellen Massenprodukten führen, aber die Befriedigung, etwas besser gemacht zu haben, wiegt vieles auf.

Eine lokale, intuitiv zu bedienende Oberfläche, bestehend aus Display, einer Zahl von Tastern und einem Drehencoder, sollte mannigfaltige Operationen ermöglichen, auch wenn der Wecker nicht am Computer hängt. Diese lokale Steuerung wird natürlich vor allem zur Einstellung der Weckzeit und zur Anzeige der wichtigsten Parameter gebraucht und nicht zur Eingabe eines komplexen 7-Tage-AN/AUS-Musters. Dies lässt sich viel bequemer mit einem PC (und einer Windows-Software) über den USB realisieren.

Die Vorderseite des Weckers muss eine große 4-Digit-Anzeige aufweisen, in der normalerweise die Tageszeit erscheint. Dazu kommen bestimmte zusätzliche (hintergrundbeleuchtete) Angaben wie PM, 24 hr, A1–A4, S1–S4, MON, TUE, WED. Die „Alarm"-Angaben A1, A2, A3, A4 zeigen den AN/AUS-Status von bis zu vier täglichen Alarmereignissen. Ebenfalls vier Anzeigen S1, S2, S3, S4 geben den AN/AUS-Status der zeitgeschalteten Ausgänge an.

Die Displayhelligkeit muss variabel sein, und zwar gesteuert durch die Umgebungshelligkeit (automatic mode), das Alarmereignis (Power-safe-mode) oder manuell durch die Benutzeroberfläche. Der Modus lässt sich natürlich ebenfalls einstellen.

Die Alarm-Funktion weist eine Vielzahl von konfigurierbaren Optionen auf wie die Wahl des Alarmtons, eine automatische Lautstärkesteigerung (Ramp-up-Funktion, anfängliche und maximale Lautstärke, Zahl der und Intervall zwischen den Alarmwiederholungen und vieles, vieles mehr). Es gibt eine gute Auswahl von (schon „eingebauten") Alarmtönen, die vom Anwender umgestaltet werden können. Für jedes Alarmereignis im 7-Tage-Rhythmus lässt sich – wenn gewünscht - ein individueller Ton wählen. Dabei muss es sich nicht um ein simples „Piep" handeln, es lassen sich beliebige digitale Soundclips (PCM/MP3) mit guter Soundqualität über die USB-Schnittstelle auf die Alarmuhr laden.

Die Schlummer-Funktion kennt zwei Modi: Im klassischen Modus wird der Alarmton durch einen einzelnen Druck auf den Taster unterbrochen und nach einer einstellbaren Zeit (zum Beispiel nach zehn Minuten) fortgesetzt. Dies wiederholt sich, bis der Alarm manuell (etwa durch drei schnelle Betätigungen des Tasters) deaktiviert wird oder bis ein (einstellbares) Zeitintervall abgelaufen ist. Im alternativen „Auto-repeat"-Schlummermodus wiederholt sich der Alarm in einem einstellbaren Intervall und mit einer ebenfalls einstellbaren Anzahl von Wiederholungen, bevor er abbricht. Ein einziger Druck auf die Schlummer-Taste stoppt den Alarm. Der Unterschied: Im zweiten Fall muss der Alarm morgens nicht aus- und ergo am Abend nicht wieder eingeschaltet werden!

Im normalen Betrieb dient der Drehencoder der manuellen Einstellung der Displayhelligkeit. In den Zeiteinstellungs- und Konfigurationsmodi

werden am Knopf die gewünschten Änderungen an den dargestellten Daten vorgenommen.
Es muss eine Möglichkeit vorhanden sein, den nächsten Alarm bis zu 48 Stunden im Voraus zu aktivieren. Vielleicht könnte man durch einen Druck auf den Schlummer-Taster (natürlich wenn gerade kein Alarm aktiv ist) in einen „Alarm-bestätigen"-Modus wechseln, in dem die Zeit und der Tag des nächsten Alarms angezeigt wird. Durch einen längeren (mehr als zum Beispiel zwei Sekunden währenden) Druck würde der mit dem Alarmereignis verbundene Ton ertönen, um die Lautstärke zu testen. Dann könnte man beispielsweise die Lautstärke am Drehencoder oder an zwei Tastern den Sound selbst ändern.
Echte Technik-Enthusiasten werden die Möglichkeiten der Weckerrückseite zu schätzen wissen: Eine USB-Schnittstelle für den PC, vier I/O-Buchsen, mit den sich zu programmierten Zeiten verschiedenste Accessoires wie Alarmglocke, Nachttischlampe, Radio/TV, Teekessel oder Kaffeemaschine, Computer samt Peripherie und so weiter ein- und ausschalten lassen. Jeder der vier Kanäle sollte unabhängig und für jeden Wochentag anders programmierbar sein.
Eine Backup-Batterie muss natürlich vorhanden sein, um alle Zeitinformationen zu erhalten, wenn einmal die Netzspannung kurzzeitig ausfallen sollte. Wenn die Batteriespannung unter einen bestimmten Minimalwert fällt, wird ein Warnsignal angezeigt.
Alle vom Anwender eingestellten Optionen, Parameter, das ganze Ein- und Ausschaltmuster über sieben Tage soll in einem nichtflüchtigen Speicher vor Stromausfall geschützt werden. Die gesamte Konfiguration muss über eine Windows-Software hoch- und heruntergeladen werden können. Wenn mit dem PC verbunden, soll sich Zeit und Wochentag automatisch der internen Echtzeituhr des PCs anpassen (die ihre Einstellung wiederum mit der Internet-Zeit synchronisiert).
Und schließlich müssten über den USB auch Firmware-Upgrades mit zukünftigen Funktionserweiterungen und „Bug-Fixes" auf den Wecker gespielt werden können.
Zugegeben: Viele der gewünschten Features sind noch nicht realisiert. Hier können Sie zum Zug kommen und durch eigene Soft- oder Hardwareerweiterungen zu diesem Projekt beitragen!

Technische Einführung

Die Hardware besteht aus einer Haupt- und einer Displayplatine, dem Drehencoder/Schalter, einem Lautsprecher sowie dem Schlummer/Abbruch-Taster. Auf SMDs auf den beiden Platinen wurde verzichtet, so dass der Aufbau des Weckers auch Lesern möglich ist, die über wenig Löterfahrung und/oder über kein entsprechendes Equipment verfügen. Auch der Mikrocontroller ist aus diesem Grund „gesockelt" und lässt sich in eine 52-polige PLCC-Fassung stecken.
Die Hauptplatine beherbergt einen einfachen, preisgünstigen Sound-Synthesizer. Der Audio-Pegel (Attac/decay, Hüllkurve und Amplitude) werden digital vom Controller mit Pulsweitenmodulation gesteuert. Mit einigen Kniffen in der Firmware, die eine Frequenzmodulation und/oder Amplitudenmodulationseffekte hinzufügen, kann eine Vielzahl von Sounds von wohlklingend bis ganz schrecklich erzeugt werden. In der ursprünglichen Firmware sind 16 Sounds vordefiniert, alternativ kann der Anwender mit den im EEPROM gespeicherten „Sound-shapes" herumspielen und so neue Sounds erzeugen.
Alle Teile lassen sich in einem selbst gefertigten Gehäuse nach Wunsch und eigenem Gusto oder einem käuflichen Standard-Gehäuse unterbringen. Wir haben das Modell CM6-225 von PAC-TEC verwendet, es gibt auch ähnliche (und preiswertere) Produkte chinesischer Provenienz. Die Platinen sind auf dieses Gehäuse zugeschnitten (siehe „Assembly Instructions" im Download). Die sieben kleinen Taster der Steuerung sitzen normalerweise oben an der Display-Platine, so dass sie entlang der Vorderkante der Oberseite erreichbar sind, können aber auch auf einem Stückchen Lochraster an anderer Stelle des Gehäuses angebracht und über Kabel mit der Display-Platine verbunden werden. So sind andere, vom Standard-Weckerdesign abweichende Gehäusearrangements denkbar.
Hardware-Erweiterungen sind über eine optionale Mezzanine-Platine (Zwischengeschoss) möglich, die mit einem Flachbandkabel mit der Hauptplatine verbunden wird. So ist eine zusätzliche Funktionalität des Weckers denkbar oder auch eine völlig andere Anwendung. Der Autor denkt schon an ein Board mit MP3-Decoder und seriellem Datenspeicher für die Speicherung und Wiedergabe von hi-fidelen digitalen PCM/MP3-Alarmsignalen.
Die Firmware basiert auf dem Betriebssystem ALERT (Low-End Real-Time) des Autors. Wer das Projekt realisieren möchte, kann eine eigene Firmware schreiben (und verteilen) oder eine kostenlose Version herunterladen. Der Sorcecode ist ebenfalls unter [1] verfügbar; für Leser, die die Firmware modifizieren oder erweitern und eige-

Eigenschaften des 7-µC-Weckers / Zeitschalters

- Bequemer und einfacher Gebrauch durch intuitive lokale Steuerung und Setup-Menü
- Drehencoder für einfache Zeiteinstellung und Dateneingabe
- One-touch-Bestätigung des anstehenden Alarms und der Lautstärkeeinstellung
- Multiples Weck- und 7-Tage-Wochenschema
- Bis zu vier Weckzeiten pro Tag programmierbar
- Verschiedene Weckzeiten für jeden Tag (wenn gewünscht)
- Vielfältige Schlummer/Ausschaltoptionen
- „Klassischer Modus“ — Alarm ertönt kontinuierlich, bis er durch Tastendruck abgebrochen wird
- „Auto-repeat-Modus“ — Alarm ertönt in periodischen Intervallen, bis er abgebrochen wird
- Einstellbares Schlummerintervall in beiden Modi
- Variable Displayhelligkeit, einstellbar durch Umgebungslichtsensor
- aktiviert durch Alarm- oder Timer-Ereignis
- Manuell am Drehknopf änderbar
- Hochqualitative, programmierbare Wecksignale
- Wahl aus einer Reihe vorgegebener Alarmsounds
- Individuelle Anpassung der vorgegebenen oder Erzeugung eigener Sounds
- Jedes Alarm-Ereignis erhält individuellen Sound
- USB-Link zum Host-PC [1]
- Synchronisation des Weckers mit PC-Zeit/Datum (Internet-Zeit)
- Speichern und Wiederherstellen der Optionseinstellungen, Alarmereignisse und Zeitschaltmuster[2]
- Download der Firmware-Upgrades (mit Erweiterungen und Bug Fixes)
- Anschlussmöglichkeiten für Geräte an Zeit- und Alarmschalter-Ausgänge
- Vier unabhängige Zeitschalt-Kanäle mit 7-Tage-EIN/AUS-Muster
- Steuerung von 230-V-Geräten (Lampen, Radio, TV/AV, Computer und Peripherie)[3]
- Alarm-aktivierter Ausgang zur Steuerung externer Geräte (Sirene, elektrisches Ventil an der Wasserleitung)
- Hilfs-Countdown-Timer mit Steuerausgang für externe Geräte
- Schaltet PC-Peripherie automatisch ein, wenn USB-Spannung erkannt wird.

1. PC-Link verwendet USB-CDC Virtual COM port API, auch erreichbar über HyperTerminal.

2. Unterstützt von Windows-GUI-Applikationssoftware (noch zu entwickeln)

3. Benötigt ein optionales Solid-State-Relais mit vier Ausgängen zur Steuerung von Netzspannungsgeräten.

nen Vorstellungen anpassen wollen.

Die Hardware basiert auf dem Mikrocontroller AT89C5131 USB von Atmel, der in einem PLCC-Gehäuse lieferbar ist und über einen großzügigen Flash-Programmspeicher (32°kB) mit separatem Daten-EEPROM (1°kB) verfügt. USB ist „on-chip“, der Controller ist weit verbreitet, leicht erhältlich und preiswert und verfügt – ganz wichtig – über einen Bootloader für eine Flash-Programmierung über den USB-Port, so dass ein zusätzliches Programmiergerät nicht erforderlich ist.

Der einzige Nachteil des Controllers ist nach Meinung des Autors sein ineffizienter 8051-Kern. Allerdings macht das nichts aus, wenn man in C programmiert und genügend Flash-PROM zur Verfügung steht, um den Verlust des ineffektiven 8051-Befehlssatzes zu kompensieren. Fairerweise muss man sagen, dass das RAM unterhalb der Adresse 0xFF noch effizient angesprochen werden kann, für das „Extended Data RAM“ gilt das aber nicht.

Wer selbst die Firmware entwickelt, findet einen kostenlosen C-Compiler für 8051er im Netz (SDCC 8051 compiler). Insgesamt stellt der 89C5131 eine gute Wahl für den Selbstbau dar, für einen „professionellen“ Wecker würde der Autor aber zum Beispiel einen Freescale MC9S08-JM60 oder Microchip PIC18F66J50 oder einen Controller mit 32-bit ARM-Kern einsetzen.

Allgemeiner Betrieb

Spannungsversorgung

Der Wecker wird von einem Gleichspannungs-Steckernetzteil mit 9°V versorgt. Er benötigt weniger als 100°mA, wenn das Display mit normaler Helligkeit erstrahlt. Selbstverständlich muss hier (weil Dauerbetrieb) ein geschaltetes Steckernetzteil verwendet werden, um Strom und Kosten zu sparen. Schließlich verwendet auch der Wecker einen geschalteten 5-V-Spannungsregler für eine optimale Effizienz.

Der Wecker kann auch über den USB-Port versorgt werden (Vbus = 5 V DC). Dann ist die externe DC-Versorgung (+VEX) aber nicht verfügbar, da sie direkt mit dem 9-V-Versorgungseingang verbunden ist. Natürlich funktioniert nun auch angeschlossene Gerätschaft, die 9°V DC benötigt, nicht. Fehlen sowohl Steckernetzteil als auch USB-Versorgung, läuft die Uhr auf Kosten der Backup-Batterie weiter.

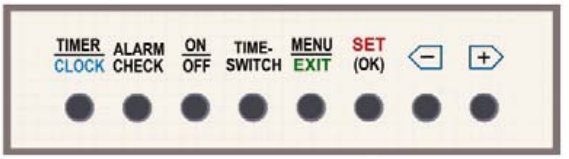


Bild 1.
Legende für die Tasten auf der Oberseite.

Backup-Batterie

Der Wecker wurde nicht entworfen, um dauerhaft von der Batterie versorgt zu werden. Diese ist nur vorhanden, um während kurzzeitiger Unterbrechungen der Netzspannung (sei es aufgrund eines Fehlers des E-Werks oder, um das Gerät von einem Raum in den anderen zu tragen, beispielsweise um es an den PC zu koppeln) Zeit, Wochentag und Einstellungen zu erhalten. Während der Batterie-Versorgung sind auch nur die absolut notwendigen Funktionen aktiv.

Die Backup-Batterie besteht aus drei Mignonzellen (AA), entweder vom Alkaline-Typ (1,5°V) oder aufladbar (1,2°V). Der Controller arbeitet bei Batteriespannungen im Bereich 3,3 5°V. Der Mikrocontroller überwacht den Füllstand der Batterien kontinuierlich, wenn die Schaltung mit 9°VDC versorgt wird. Wird der Wecker für längere Zeit ausgeschaltet, kann man per Schalter die Backup-Batterie abkoppeln.

Versorgung für die Peripherie

Wie schon angedeutet, steht für an der Alarm- und an der Zeitschalterbuchse angeschlossene Geräte eine geschützte Gleichspannungsversorgung zur Verfügung, die das Steckernetzteil bereitstellt.

Dazwischen liegt nur eine (rücksetzbare) Polyswitch-Sicherung, die die Gesamtstromaufnahme aller angeschlossenen Geräte (Glocke, Lampe, Radio) auf 0,5°A begrenzt.

Bild 2.
Die Anzeigeeinheit.

Firmware-Download

Die Firmware wird über den USB-Port in den Flash-Speicher des Controllers geladen. Dazu benötigt man Atmels Programmierhilfsmittel FLIP (Flexible In-system Programmer), das man in seiner aktuellen Version samt Gebrauchsanweisung von der Website von Atmel beziehen kann. Um den USB-Bootloader des Controllers zu starten, drückt (und hält) man den ISP-Taster, drückt den Reset-Taster und lässt ihn wieder los, ebenso danach den ISP-Taster. Das Display sollte leer sein. Das USB-Kabel wird mit dem PC verbunden und die FLIP-Applikation gestartet. Dann folgt man den Anweisungen im Anwenderhandbuch.

Host Command Interface (HCI)

Die Firmware enthält ein Kommunikationsprotokoll, das in erster Linie für das Versenden von Daten vom und zum Host-PC über USB gedacht ist. Das Protokoll basiert auf einem einfachen Kommandozeileninterpreter, daher der Ausdruck „host command interface“ (HCI). Da das Protokoll druckbare ASCII-Zeichen verwendet, kann das HCI vom Anwender mit einem Terminal-Emulator (wie HyperTerminal) erreicht werden. Das HCI lässt sich interaktiv zur Einstellung von Wecker-Optionen und Parametern einsetzen. Erweiterte Konfigurationsoptionen, die nicht über das „local user interface“ (der Tastensatz) erreichbar sind, lassen sich ebenfalls mit dem HCI und Hyperterminal oder einer selbst geschriebenen Windows-Applikation setzen.

Der USB-Port des Controllers erscheint am Host-PC als virtueller COM-Port. Windows-Applikationssoftware kann mit dem Wecker über die gewöhnlichen Funktionsaufrufe der Windows COM-Port-API kommunizieren (open, read, write, etc.).

Eine Befehlszeile besteht aus einem 2-Zeichen-Befehl und einer Anzahl von Benutzer festgelegten Argumenten (Parametern). Manche Befehle kennen keine Argumente (wie im wirklichen Leben). Zwischen den Argumenten (inklusive 2-Zeichen-Befehlsname) muss, sofern es mehrere sind, ein einzelnes Leerzeichen gesetzt werden. Die HCI-Eingabe beachtet prinzipiell keine Groß-/Kleinschreibung. Es gibt keine Möglichkeit, die Befehlszeile zu editieren, aber Escape oder Strg-X löscht die ganze Zeile.

Eine knappe Befehlserläuterung wird nach dem Hilfe-Kommando „HE“ angezeigt. Debug-Befehle unterstützen den Programmierer bei der Entwicklung der Firmware. Weitere Details zu dem HCI können dem User Guide im freien Download unter [1] entnommen werden.

Arbeiten am Steuerpult - eine Übersicht

Bild 1 zeigt das Standard-Layout der Steuer-Taster auf der Oberseite des Geräts. Auch die große Schlummer/Abbrech-Taste befindet sich dort. Zusammen mit dem LED-Display und dem Drehencoder bildet sie das lokale Benutzerinterface. Die meisten Funktionen und Einstellungen lassen sich mit diesem Interface vornehmen, nur für weitergehende Optionen muss das USB-Kabel angeschlossen werden.

Anders als die meisten Wecker zeigt das Display auch den Wochentag an. Wird ein Alarm eingestellt, kann der Benutzer entscheiden, für welche Tage der Woche der Alarm gelten soll und für welche nicht. Andere LEDs zeigen die AN/AUS-Zustände der vier Alarm-Ereignisse (A1…A4) und der vier Zeitschalter (S1…S4). Die Zustände der Zeitschalter lassen sich manuell mit den Tastern (**Bild°2**) „überschreiben“.

Die meisten der Taster haben mehr als nur eine Funktion. Die Aktion, die auf einen Tastendruck erfolgt, ist abhängig vom Kontext der Operation. Im normalen Tageszeit-Anzeige-Modus beispielsweise bewirkt ein Druck auf die Tasten [+] oder [−] die Darstellung eines anderen Datums, Sekunden, Jahr, Tag im Monat und so weiter. Im Einstellungs-Modus dagegen werden die Tasten [+] und [−] zum In- oder Dekrementieren eines numerischen Werts oder zum Scrollen in einer Liste (Wochentage) verwendet.

Prinzipiell weist im Einstellungsmodus ein blinkendes Digit oder eine blinkende LED auf das Datum hin, das aktuell durch Tastendruck oder Drehen am Encoder verändert werden kann. Die Exit-Taste verlässt den Einstellungsmodus, ohne Änderungen vorzunehmen.

Tabelle 1 zeigt eine Übersicht der Möglichkeiten des Steuerpults („local user interface“). Eine vollständige und ausführliche Liste finden Sie im *User Guide* im Download-Paket.

In der nächsten Ausgabe geht es weiter: Der zweite Teil dieser Projektbeschreibung wird sich ausführlich mit dem Schaltplan und dem Bau des Weckers beschäftigen.

(100149)

Weblinks

[1] www.elektor-magazine.de/100149

[2] www.elektor-labs.com

Tabelle 1. Funktionen der Taster

Taste	Resultat
TIMER/CLOCK Tastendruck	Countdown-Timer (anfängliche oder verbleibende Zeit mm.ss) wird angezeigt. Von hier kann der Countdown-Timer gesetzt und gestartet werden.
ALARM CHECK Tastendruck	Eintritt in den Alarm-Check-Modus. Die programmierte Zeit des selektierten Alarms (A1…A4) erscheint. Von hier kann der selektierte Alarm (Zeit und Wochentag) neu programmiert, ein- oder ausgeschaltet werden. Mit den Tasten [+] und [−] bestimmt man den Wochentag, die ON/OFF-Taste wechselt den Alarm-Status. Mit der SNOOZE-Taste ändert man den Soundeffekt und die Lautstärke.
TIME-SWITCH Tastendruck	Eintritt in den Zeitschalter-Steuermodus. Ein selektierter Zeitschalt-Kanal (S1…S4) kann ein- oder ausgeschaltet werden, das programmierte Zeitschaltmuster überschreibend.
SET – Taste für >3 s gehalten	Eintritt in den Tageszeit-Modus (TOD). Der Drehencoder stellt die Stunden, nach nochmaligen Druck auf die SET-Taste die Minuten ein. Nochmals SET zur Bestätigung oder EXIT zum Abbruch des Modus ohne Änderung drücken.
MENU – Taste für >3 s gehalten	Eintritt in den Menu-Setup-Modus. Von hier können diverse Anwenderoptionen und Parameter eingestellt werden. Die Tasten [+] und [−] rollen durch eine Liste.
[+] or [−] Tastendruck	Der Displayinhalt wechselt von der normalen Tageszeit zu Sekunden oder einer Datumanzeige (Jahr, Monat, Tag). Die [+]-Taste scrollt durch die Listeneinträge Sekunden (ss.cc), Jahr (20xx), Datum (MM dd) zurück zur Tageszeit, die [−]-Taste in die andere Richtung. Die SET-Taste kann bei jedem angezeigten Eintrag gedrückt werden.
SNOOZE/CANCEL Taste gedrückt	Alarmbestätigung: Die Zeit des anstehenden (freigegebenen) Alarms erscheint bis zu 48 Stunden im Voraus. Wird die Taste länger als 2°s gedrückt, ertönt der Alarmsound. Wird jetzt gleichzeitig die SET-Taste gedrückt, können Alarmsound und Lautstärke geändert werden.
Encoder-Drehung während normaler Tageszeitanzeige	Die Displayhelligkeit wird eingestellt. Abhängig von den gewählten Optionen bleibt die Helligkeit bis zur nächsten Einstellung gleich oder ändert sich wieder automatisch.

Flugfunk-Scanner

Für die Freunde der Luftfahrt kann das Beobachten naher Flugzeuge und das Verfolgen der Kommunikation zwischen Luft und Boden ein überaus informatives Freizeitvergnügen sein. Unser Flugfunk-Scanner ist ein Ohr für das Geschehen zwischen Himmel und Erde. Der Scanner ist schnell aufgebaut, er hat lediglich ein mechanisches Abgleichelement, und er ist über USB steuer- und einstellbar.

Von **Gert Baars** (NL)

„Frankfurt Approach, LH 171 with information delta, descending to flight level 70 just passed artip inbound spy, speed 250 knots…",

- es lohnt sich, auch einmal in diesen Bereich des Äthers hinein zu hören. Was dort an Informationen ausgetauscht wird, hat zwar nicht immer hochdramatischen Inhalt. Oft sind es aktuelle Positionsdaten oder Anweisungen an die Piloten beim Starten und Landen, reine Routine. Doch schnell entsteht das unterschwellige Gefühl, in den Cockpits neben den Piloten zu sitzen und rasant von der Piste abzuheben oder dort behutsam aufzusetzen.

Flugfunk-Scanner sind zwar schon recht preiswert im Handel käuflich, doch es macht mehr Spaß, ein solches Gerät selbst zu bauen und damit Versuche anzustellen. Unser hier vorgestellter Flugfunk-Scanner ist einerseits unkompliziert in Konzept und Aufbau, andererseits verhelfen ihm der Mikrocontroller und die zugehörige Software zu einer Reihe bemerkenswerter Eigenschaften. Als Beispiel sei das Programmieren der Empfangsfrequenzen über die integrierte USB-Schnittstelle genannt.

Vorbetrachtungen

Damit die Hardware und Software überschaubar bleiben, haben wir den Empfangsbereich unseres Scanners auf das zivile Flugfunkband 108...137 MHz zugeschnitten. In diesem Bereich war das Frequenzraster ursprünglich in 25-kHz-Schritte unterteilt, später wurde ein Raster im Abstand 25/3 kHz = 8,33 kHz eingeführt. Die feinere Unterteilung in 8,33-kHz-Schritte wird jedoch nur selten angewendet. Unser Scanner ist in einem 1-kHz-Raster abstimmbar, so dass die Empfangsfrequenz um höchstens 330 Hz von der tatsächlichen Flugfunkfrequenz abweicht. Im Flugfunkbereich beträgt die Kanalbandbreite 6 kHz, die Modulationsart ist die Amplitudenmodulation (AM). Diese Eckdaten legen fest, welche grundlegenden Eigenschaften ein Scanner für das zivile Flugfunkband mindestens haben muss. Unser Scanner kommt ohne Eingabetastatur und ohne Display aus, die Funktionen dieser Bedienelemente übernimmt ein anzuschließendes Terminal. Über die USB-Verbindung werden Daten mit einem PC ausgetauscht, auf dem ein Terminalprogramm wie zum Beispiel das zu Windows gehörende „Hyperterminal" läuft. Die Empfangsfrequenzen werden am PC-Bildschirm program-

Portabler Betrieb, konfigurierbar über USB

Eigenschaften

- Programmier- und einstellbar unter Windows
- 100 programmierbare Frequenzspeicher
- Empfangsbereich 108...137 MHz, AM
- Empfindlichkeit 0,2 µV bei 6 dB S+N/N
- Scan-Geschwindigkeit 5 Kanäle/s
- Belegte Flugfunk-Frequenzen werden selbsttätig gespeichert
- Handabstimmung mit Up-/Down-Tastern möglich

miert, auch die übrigen Einstellungen können von dort vorgenommen werden.

Die Empfängerschaltung ist überschaubar aufgebaut, sie arbeitet mit einem integrierten Baustein, der einen Einfachsuper für FM-Empfang an Bord hat. Das demodulierte AM-Signal ist das Signal, das bei FM-Empfang die Signalstärkeanzeige steuern würde. Einfachsuper haben die Eigenschaft, nicht nur für die Empfangsfrequenz, sondern auch für die Spiegelfrequenz empfindlich zu sein. Die unerwünschten Spiegelsignale lassen sich umso wirkungsvoller unterdrücken, je weiter der Abstand zwischen Empfangsfrequenz und Spiegelfrequenz ist. Die Zwischenfrequenz (ZF) muss folglich möglichst hoch liegen. Beim verwendeten Empfängerbaustein darf die ZF laut Datenblatt bis 25 MHz betragen, auch bei 27 MHz sind noch keine nachteiligen Auswirkungen zu beobachten. Leider sind ZF-Filter für 27 MHz nur mit Mühe beschaffbar, sie sind relativ kostspielig. Unser Flugfunk-Scanner arbeitet deshalb mit zwei Kettenfiltern mit insgesamt vier Quarzen, die zusammen nur etwa zwei Euro kosten. Die Zwischenfrequenz 27 MHz hat den Vorteil, dass die Spiegelfrequenzen bei Untermischung in den Bereich 54...83 MHz fallen, ein Bereich, der unterhalb

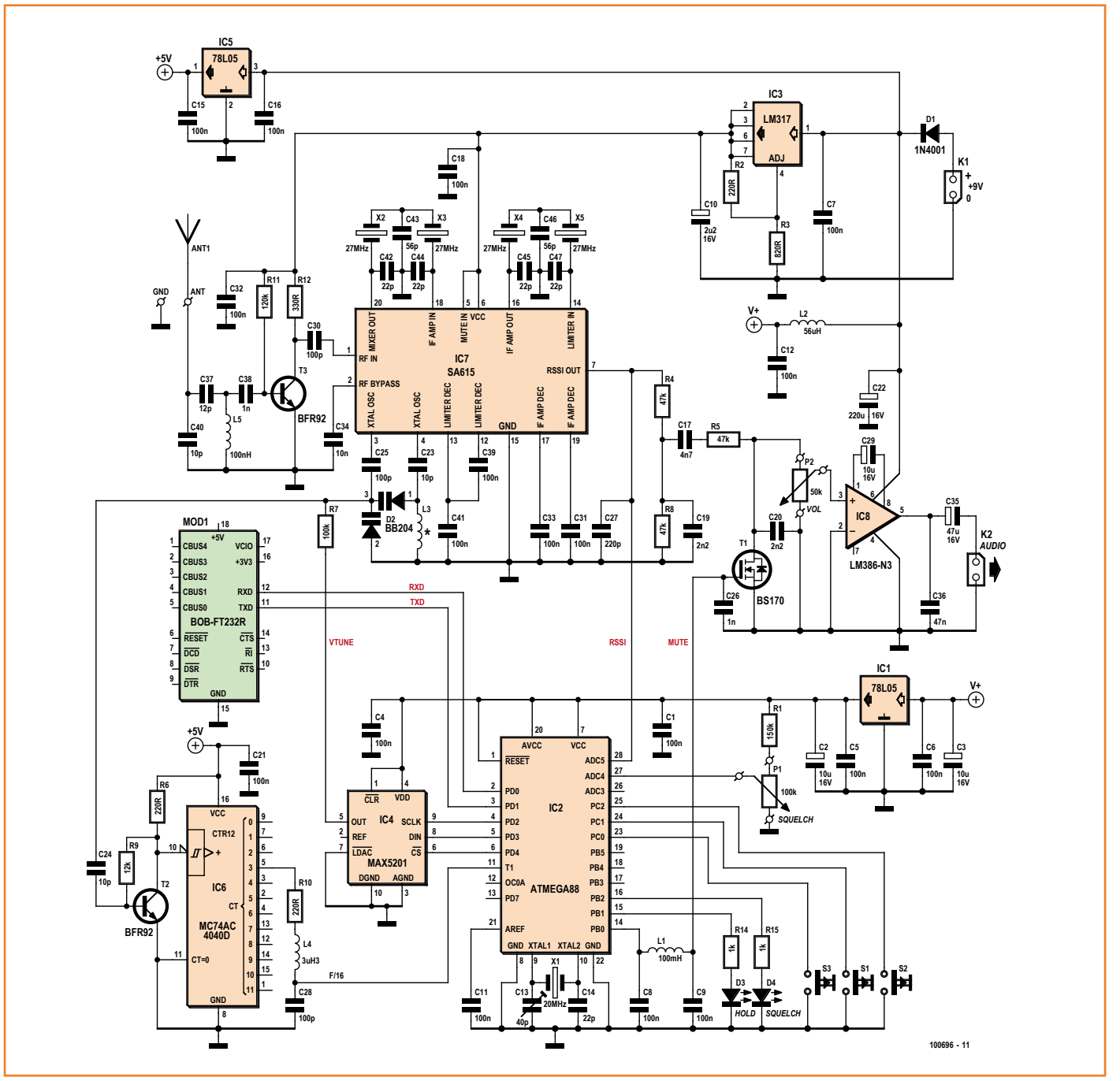


Bild 1.
Die Schaltung besteht aus einem analogen Teil (oben) und einem digitalen Teil (unten). Drei Spannungsregler entkoppeln die Betriebsspannungen.

des FM-Rundfunkbands liegt.

Für die digitale Frequenzwahl werden in Empfängern häufig die PLL oder die DSS eingesetzt. Ein anderes, weniger komplexes Verfahren ist die FLL (Frequency Locked Loop), bei dem der Mikrocontroller die Frequenz des VFO misst, sie mit der Sollfrequenz vergleicht und gegebenenfalls über einen DAC korrigiert. Verschiedene im Elektor-Labor durchgeführte Tests haben gezeigt, dass dieses Verfahren zum Ziel führt, sofern die Genauigkeit des DAC für die schmalen Frequenzschritte ausreicht. Allerdings kann der Zähler des Mikrocontrollers die auftretenden hochfrequenten Signale nicht direkt verarbeiten, aus diesem Grund ist ihm ein Frequenzteiler vorgeschaltet. Das unkomplizierte Konzept hat auch den Vorteil, dass der Flugfunk-Scanner über nur drei Drucktaster sowie ein Squelch- und ein Lautstärke-

Potentiometer bedienbar ist. Nachdem die Empfangsfrequenzen über den PC eingegeben und gespeichert sind, ist der Scanner für den portablen Betrieb gerüstet.
Ein Scanner hat definitionsgemäß die Fähigkeit, belegte Frequenzen selbsttätig zu finden und zu speichern. Nachdem der Squelch unseres Flugfunk-Scanners auf die gewünschte Signalschwelle eingestellt ist, genügt ein Tasterdruck, um den Scan-Vorgang zu starten.

Empfangsteil und Betriebsspannungen

Die Eingangsstufe (siehe Schaltung in **Bild 1**) arbeitet mit einem SA615 oder auch NE615. Dieser Baustein ist eine weiterentwickelte Version der Kombination aus NE602 und NE604, wobei im NE602 der Mischer mit VFO und im NE604 die ZF-Stufe, der Begrenzer und der FM-Demodulator untergebracht waren. Da hier ein FM-Demodulator nicht zum Einsatz kommt, bleiben einige Anschlüsse des SA615 unbeschaltet. Der SA615 stellt an seinem Ausgang RSSI eine Spannung bereit, die zur Stärke des empfangenen Signals proportional ist. Dieses Signal steuert über ein Rauschfilter die NF-Ausgangsstufe. Das gleiche Signal wird dem Mikrocontroller zugeführt, so dass der Mikrocontroller den Empfang eines Flugfunk-Signals während des Scan-Vorgangs erkennen kann.
Das von der Antenne kommende Signal wird gefiltert und verstärkt, bevor es zum Mischer mit dem VFO gelangt. Eine zweifache Varicap-Diode stellt die VFO-Frequenz ein. Das resultierende ZF-Signal mit der Mittenfrequenz 27 MHz durchläuft zwei zweistufige Kettenfilter (X2...X5), die Bandbreite beträgt ungefähr 6 kHz. Genau betrachtet liegt die ZF bei 26,998 MHz. Falls nötig ist dieser Wert im Setup der Software modifizierbar. Die Quarze der Kettenfilter müssen identische Typen mit der Grundschwingung auf 27 MHz sein. Mögliche geringe Toleranzen können im Software-Setup kompensiert werden.
Das VFO-Ausgangssignal steuert über T2 einen Zähler des Typs 74AC4040, der als Frequenzteiler durch 16 geschaltet ist. Da die VFO-Frequenz im Bereich f_{HF} - f_{ZF} = 81...110 MHz variiert und der Mikrocontroller lediglich Signale bis 8 MHz verarbeitet, muss der Teilfaktor mindestens 14 betragen.
Das NF-Signal an Pin 7 des SA615 steuert über einige Widerstände, den Stummschalte-FET T1 und den Lautstärke-Poti P2 die NF-Endstufe mit dem LM386. An diesen Endverstärker kann ein Lautsprecher (4 Ω oder 8 Ω) unmittelbar angeschlossen werden.

Frequenzbereiche

Für das Konzept eines Flugfunk-Scanners ist der Empfangsbereich ein entscheidendes Kriterium. Die wichtigste Bedeutung haben das Frequenzband 108...137 MHz der zivilen Luftfahrt sowie das Frequenzband 230...400 MHz der Militärs. In beiden Bändern ist die Amplitudenmodulation (AM) am gebräuchlichsten. Ferner sind im Frequenzraum 3...30 MHz einige Bänder reserviert, in denen Fluggeräte und Bodenstationen Informationen austauschen. Für die Sprachkommunikation wird dort meistens SSB (Single Side Band Modulation) angewendet.
Der Empfangsbereich dieses Flugfunk-Scanners überdeckt die Frequenzen 108...137 MHz der zivilen Luftfahrt. Hier wird nicht nur der Informationsaustausch zwischen den Piloten von Passagiermaschinen und Fluglotsen im Tower abgewickelt. Auch die Luft-zu-Luft-Kommunikation und die Kommunikation von Dienststellen der Flugaufsicht finden in diesem Bereich statt. Bodengebundene Fahrzeuge sind dort ebenfalls am Funkverkehr beteiligt.

Wegen des Zusammenwirkens eines analogen und digitalen Schaltungsteils (HF-Stufen sowie Mikrocontroller mit Frequenzteiler) ist eine saubere Entkopplung der Betriebsspannungen unverzichtbar. Dies geschieht mit drei separaten Spannungsreglern: IC3 liefert die Spannung +6 V für den Empfangsteil, IC1 ist für die Spannung +5 V des digitalen Teils ohne IC6 zuständig, Frequenzteiler IC6 wird von IC5 mit +5 V versorgt.

Mikrocontroller-Steuerung

Der Mikrocontroller, ein ATmega88 (IC2), hat neben dem Speichern der Empfangsfrequenzen eine weitere wichtige Funktion: Er misst die durch 16 geteilte VCO-Frequenz. Dazu wird ein interner Zähler für die Torzeit 16 ms konfiguriert, so dass das Zählergebnis die gemessene Frequenz in kHz ist. Ein 16-bit-DAC vom Typ MAX5201 steuert den VCO, die Auflösung beträgt hier 29000 kHz / 2^{16} = 440 Hz. Bei der Bandbreite 6 kHz ist dies ein akzeptabler Wert, die Empfangsfrequenz kann genügend genau abgestimmt werden. Der MAX5201 hat ferner den Vorteil, dass er seriell gesteuert werden kann und das Ausgangssignal den „Rail-to-Rail“-Bereich abdeckt.
Im Mikrocontroller ist so viel EEPROM verfügbar, dass außer diversen Einstellparametern bequem 100 Frequenzwerte speicherbar sind. Die Frequenzen können Bänken zugeordnet werden, zum Beispiel 5 Bänken mit 20 Frequenzwerten.
Der Mikrocontroller erkennt über zwei ADC-Eingänge (Pin 27 und 28) die Stärke des Empfangssignals und den Stand des Squelch-Potis. Er ver-

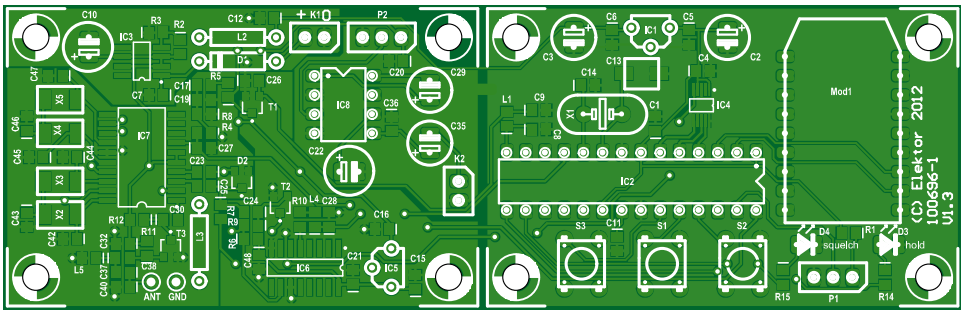


Bild 2.
Auch auf der Platine sind der analoge Teil und der digitale Teil voneinander getrennt. Die Platinenteile können mit einer Säge getrennt werden.

gleicht beide Werte und entscheidet, ob der Scan-Vorgang unterbrochen oder fortgesetzt wird. Die LEDs D3 und D4 signalisieren den Ablauf. Während des Scan-Vorgangs sind gefundene Frequenzen über die Bedientaster (S1...S3) blockierbar, sie werden zwar gespeichert, aber beim Betrieb im Empfangsmodus übersprungen.
Das inzwischen in mehreren Elektor-Projekten eingesetzte Breakout-Board (BOB) USB-FT232R [1] ist mit den Anschlüssen 2 und 3 des Mikrocontrollers verbunden. Dieses Modul, das die Verbindung mit einem USB-Port des PCs herstellt, hat seinen Platz auf der Scanner-Platine.

Software

In der Software steckt gewissermaßen die Intelligenz unseres Flugfunk-Scanners. Die Hauptschleife liest die Empfangsfrequenz aus dem EEPROM, von dem Wert wird die ZF subtrahiert, was die erforderliche VFO-Frequenz ergibt. Wegen der FLL-Abstimmung ist es nicht möglich, absolute Frequenzwerte direkt einzustellen. Deshalb wird von einem Näherungsverfahren Gebrauch gemacht, bei dem der DAC insgesamt 16 Messungen mit der Messdauer 16 ms durchführt. Der Scan-Vorgang wird dadurch zwar verlangsamt, doch andererseits ist die Scanner-Konstruktion deutlich weniger komplex als bei Scannern, die mit der PLL- oder DSS-Methode arbeiten.
Nachdem die Frequenz eingestellt ist, prüft der Mikrocontroller, ob an Eingang RSSI ein Signal liegt. Falls dies zutrifft, geht der Stummschaltausgang (Mute) auf niedrige Spannung, so dass das NF-Signal zum Lautsprecher durchgeschaltet wird. Nach Wegfall des Signals folgt eine programmierbare Wartezeit ohne Frequenzänderung, anschließend wird die nächste Frequenz aus dem EEPROM gelesen und eingestellt.
Solange der Scanner ein Signal im Flugfunk-Band empfängt, bleibt die eingestellte Frequenz unverändert. Der Puffer der seriellen Schnittstelle wird periodisch abgefragt, so dass Informatio-

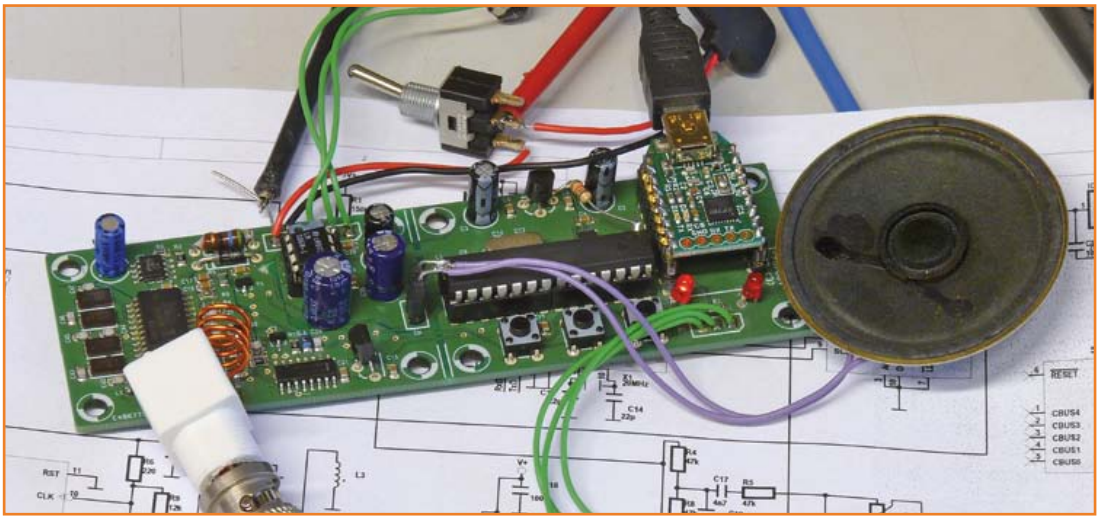


Bild 3.
Dem Labormuster fehlt zwar noch der letzte optische Schliff, trotzdem ist die Leistung überzeugend.

nen gelesen werden, die das Terminalprogramm des PCs sendet. Außerdem wird in jedem Zyklus der Status der Bedientaster abgefragt. Durch Drücken des Tasters S1 lässt sich die aktuelle Frequenz blockieren, auf dem Bildschirm wird die Frequenz mit dem Merkmal „B“ markiert. Das Blockieren einer Frequenz ist beispielsweise dann sinnvoll, wenn der Scanner wiederholt auf einem unmodulierten Träger hängen bleibt. Die Markierung kann über das Terminalprogramm aufgehoben werden.
In der Software unseres Flugfunk-Scanners sind für diverse Parameter Standardwerte vorgegeben, sie werden beim ersten Start vom Programmspeicher in das EEPROM übertragen. Maßgeblich für den Betrieb ist die im EEPROM stehende Kopie, sie ist über das Terminalprogramm des PCs modifizierbar.

Aufbau

Die Platine für den Flugfunk-Scanner ist in **Bild 2** wiedergegeben, sie ist in einen analogen und einen digitalen Bereich unterteilt. Die beiden Teile können mit einer Säge voneinander getrennt werden, wenn diese Kombination besser in ein vorgesehenes Gehäuse passt. In diesem Fall sind fünf Signalleitungen und eine Masseleitung notwendig, um die Platinenteile miteinander zu verbinden.
Am Anfang der Platinenbestückung stehen die SMDs, dies ist der schwierigere Part. Dann folgen die IC-Fassungen und die bedrahteten Bauelemente. Die Potis werden in das Gehäuse eingebaut und mit der Platine über flexible Leitungen verbunden.
Induktivität L3 muss in Eigenregie angefertigt werden, indem auf die glatte Seite eines 7-mm-Spiralbohreinsatzes vier Windungen versilberter Kupferdraht (Durchmesser 1 mm) gewickelt werden. Um mechanischen Schwingungen entgegenzuwirken, wird die Wicklung nach der Montage mit einem Kleber auf der Platine fixiert. Zum Schluss erhalten der programmierte ATmega88, der LM386 und das USB-Breakout-Board ihren Platz auf der Platine.
Bei der Wahl des Gehäuses ist zu berücksichtigen, dass außer den Potentiometern auch der Lautsprecher und eine 9-V-Batterie (oder ein 9-V-Akku) Raum beanspruchen. Die Drucktaster müssen von außen zugänglich sein, eventuell helfen mechanische Verlängerungen weiter. Die LEDs können ihre Funktion nur erfüllen, wenn sie von außen sichtbar sind. Übrig bleibt noch die Antennenbuchse, auch dafür muss ein geeigne-

Wichtiger Hinweis

Der legale Besitz und Betrieb dieses Empfängers ist in Deutschland einem Personenkreis vorbehalten, der auch im Besitz einer gültigen Zulassung zur Teilnahme am Amateurfunkdienst ist (was im Prinzip eine Amateurfunklizenz voraussetzt). Aber auch diesem Personenkreis ist das Abhören des Flugfunks nicht erlaubt, da es sich um einen Funkdienst handelt, der sich nicht an die Allgemeinheit wendet und der deshalb aus Gründen des Fernmeldegeheimnisses geschützt wird.
Voraussetzung für die Teilnahme am Flugfunkdienst ist in Deutschland ein dafür gültiges Sprechfunkzeugnis (BZF I, BZF II oder AZF). Aber auch dann darf der Flugfunk nur im Rahmen einer entsprechenden fliegerischen Betätigung und nur mit einem dafür zugelassenen Gerät empfangen werden. Im Ausland gelten natürlich die dortigen Gesetze, Richtlinien und Bestimmungen.

Stückliste

Widerstände (SMD 0805):
R1 = 150 k
R2,R6,R10 = 220 Ω
R3 = 820 Ω
R4,R5,R8 = 47 k
R7 = 100 k
R9 = 12 k
R11 = 120 k
R12 = 330 Ω
R14,R15 = 1 k
P1 = Trimmpoti 100 k lin.
P2 = Poti 50 k log.

Kondensatoren (SMD 0805, falls nicht anders angegeben):
C1,C4...C9,C11,C12,C15,C16,C18,C21,C31...C33,C39,C41,C48 = 100 n
C2,C3,C29 = 10 µ/35 V stehend
C10 = 2µ2/63 V stehend
C13 = Trimmer 8p5...40p (Murata TZB4P400AB10R00)
C14,C42,C44,C45,C47 = 22 p
C17 = 4n7
C19,C20 = 2n2
C22 = 220 µ/16 V stehend
C23,C24,C40 = 10 p
C25,C28,C30 = 100 p
C26,C38 = 1 n
C27 = 220 p
C34 = 10 n
C35 = 47 µ/16 V stehend
C36 = 47 n
C37 = 12 p
C43,C46 = 56 p

Induktivitäten:
L1,L5 = 10 nH (SMD 0805)
L2 = 56 µH
L3 = 4 Windungen Cu-Draht versilbert 1 mm, h = 8 mm, Durchmesser innen = 7 mm
L4 = 3,3 µH (SMD 0805)

Halbleiter:
D1 = 1N4001
D2 = BB207 (TO236)
D3,D4 = LED 3 mm, Low Current
T1 = BS170F (SOT23)
T2,T3 = BFR92A (SOT23)
IC1,IC5 = 78L05
IC2 = ATMEGA88-20PU (programmiert, EPS 100969-41)
IC3 = LM317 (SOIC-8)
IC4 = MAX5201AEUB+
IC6 = 74AC4040 (SOIC-16)
IC7 = SA615D (SOIC-20)
IC8 = LM386 (DIP-8)

Außerdem:
X1 = Quarz 20 MHz
X2...X5 = Quarz 27 MHz (Grundschwingung, z. B. Citizen CS1027.000MABJ-UT)
USB-FT232R Breakout-Board 110553-91 [1]
S1...S3 = Drucktaster 6 mm (Multicomp MC32830)
2 Stiftkontaktleisten 2-polig
DIP-Fassung 28-polig, für IC2
DIP-Fassung 8-polig, für IC8
2 Kontaktleisten 10-polig einreihig, für USB-FT232R Breakout-Board
Kleinlautsprecher 4 Ω oder 8 Ω
Platine 100696-1

Bedienung

Beim Einschalten des Flugfunk-Scanners gibt das Terminalprogramm auf dem PC folgende Meldung aus:

```
********* Airbandscanner startup ********

Keyfunctions:

H         -help (this page)
S         -show stored frequencies
B <fn>    -block on/off, fn is freq.number
T F(6)    -tune to frequency(Khz)
R         -scanner run/stop
F fn F(6) -store frequency(Khz) fn=freq.nr.
I Fif(5)  -Fif (Khz)
N <nf>    -#frequencies to scan (1-100)
W <Tw>    -scanning waittime (x 0.5 sec.)
M <fn>    -memory start from fn=freq.nr.
P <nr>    -#passes (2-15) for autom. search.

Current settings:

Fif       = 26998
Twait     = 2
#Freqs.   = 20
Mem.start = 1
#Srch.pas = 5


Fn 1
```

Der Scanner befindet sich im Scan-Modus, der Platz des Frequenzspeichers erscheint in der letzten Zeile. Bevor der Scanner ein Kommando ausführen kann, muss der Scan-Modus mit R gefolgt von <Enter> unterbrochen werden. Die meisten, aus nur einem Buchstaben bestehenden Kommandos haben eine leicht zu merkende Bedeutung. Nach S <Enter> gibt der Scanner die Frequenzen in Form einer Tabelle aus. Die erste Frequenz ist mit dem Kommando M einstellbar, die Anzahl der Frequenzen wird mit dem Kommando N festgelegt. Diese Einstellungen benutzt der Scanner auch im Scan-Modus. Das Scannen von Frequenzbänken ist möglich, indem mit M und N beispielsweise die Frequenzen 1...20 gewählt werden. Über das Terminalprogramm sind die Frequenzen der Bank änderbar, sie kann auch die Frequenzen 40...59 oder andere Frequenzen umfassen.

Der Scanner kann Frequenzen selbsttätig gruppieren. Angenommen, es wurden Frequenzen eingegeben, die nicht überschrieben werden dürfen, beispielsweise die Frequenzen 1...40. M wird jetzt auf 40 gesetzt, während N ebenfalls den Wert 40 erhält. Die Squelch-Schwelle muss für den Scan-Betrieb auf eine Höhe eingestellt sein, die den Empfangsbedingungen angepasst ist.

Nach Einschalten des Scanners bei gedrücktem Taster S2 leuchtet LED D3 (Hold) auf. Der Scanner durchläuft nun mehrfach (137000 – 108000) / 25 = 1160 Kanäle, die Anzahl der Suchläufe ist mit dem Kommando P festlegbar. Die gefundenen Frequenzen werden auf den Speicherplätzen 40...80 abgelegt. Nachdem das geschehen ist, verlischt LED D3. Beim nächsten Einschalten des Scanners kann das Ergebnis nach Eingabe des Kommandos S betrachtet werden. Ein Test in ungefähr 50 km Entfernung von einem Großflughafen ergab, dass bei 20 Frequenzen und P = 6 innerhalb einer halben Stunde praktisch alle Frequenzen gefunden wurden. Dazu gehörten die Kanäle Tower, Approach, Departure, Radar sowie weitere in der zivilen Luftfahrt gebräuchliche Kanäle. Die Frequenzen wurden selbsttätig

ter Einbauort gefunden werden.

Für den portablen Betrieb unterwegs genügt eine etwa 60 cm lange Teleskopantenne. Bei stationärem Einsatz hat sich die leistungsstarke Flugfunk-Antenne bewährt, die an anderer Stelle in dieser Elektor-Ausgabe beschrieben wird.

Einstellungen

Der Quarz des Mikrocontrollers ist mit einem 40-pF-Trimmer beschaltet. Wegen der notwendigen Genauigkeit der Frequenzmessung muss mit dem Trimmer die Frequenz 20 MHz präzise eingestellt werden.

Bringen Sie den Trimmer in eine Stellung, bei der er etwa halbe Kapazität hat. Stimmen Sie den Scanner mit dem Kommando T auf eine feste Frequenz ab, messen Sie gleichzeitig die Frequenz am Ausgang des 74AC4040. Stellen Sie diesen Wert abzüglich der ZF, dividiert durch 16, ein. Nachdem diese Frequenz exakt eingestellt ist, wählen Sie mit dem Kommando T eine bestimmte Empfangsfrequenz. Anschließend überprüfen Sie, ob die ZF im Setup stimmt. Eine eventuell notwendige Korrektur ist mit dem Kommando I möglich.

Die Grenzen des Empfangsbereichs sind überprüfbar, indem Sie mit dem Kommando T die Frequenzen 108.000 MHz und 137.000 MHz ein-

sortiert, der am häufigsten gefundenen Frequenz wurde Speicherplatz 1 zuwiesen, gefolgt von den weniger häufig gefundenen Frequenzen.
Zusammen mit dem Terminalprogramm arbeitet der Flugfunk-Scanner auch als Empfänger ohne Scan-Funktion. Wird auf dem PC beispielsweise das Kommando T 122400 <Enter> eingegeben, stimmt der Scanner auf die Frequenz 122,400 MHz ab. Auf dem PC-Bildschirm erscheint die zugehörige Signalstärke als ein Wert zwischen 1 und 9.

Betriebsarten

Scanner:
Nach dem Einschalten scannt der Scanner die gespeicherten Frequenzen. LED D4 (Squelch) leuchtet auf, sobald ein Signal die Squelch-Schwelle übersteigt, LED D3 signalisiert den Stopp des Scan-Vorgangs und das Durchschalten des NF-Signals zum Lautsprecher. Falls der Empfang abbricht, verlischt LED D4, die genannten Funktionen bleiben jedoch noch kurze Zeit unverändert. LED D3 wird erst nach Ablauf einer Wartezeit inaktiv, anschließend setzt der Scanner den Scan-Vorgang fort. Der Scanner bleibt auf einer beliebigen Frequenz stehen, wenn die Squelch-Schwelle auf ihren niedrigsten Wert eingestellt ist. Während eines Scan-Vorgangs werden die Nummern der aktuellen Speicherplätze zum Terminalprogramm des PCs übertragen. Auszublendende Empfangssignale, beispielsweise Trägerwellen ohne Informationsinhalt, können durch Drücken des Tasters S3 mit dem Merkmal „B“ blockiert werden. Dieses Merkmal ist nur über das Terminalprogramm löschbar.

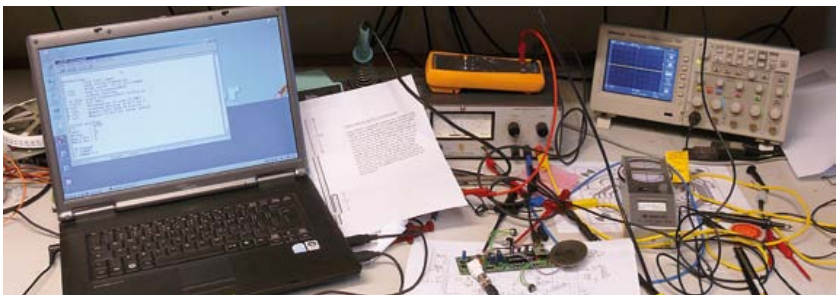
Empfänger:
Wenn Taster S3 nach dem Einschalten mindestens eine Sekunde gedrückt wird, schaltet der Scanner in den Empfänger-Modus. In diesem Modus kann die Empfangsfrequenz mit den Tastern S1 (Down) und S2 (Up) von Hand eingestellt werden. Der Squelch behält seine Funktion bei. LED D3 signalisiert das Erreichen der oberen oder unteren Grenze des Empfangsbereichs. Beim Start im Empfänger-Modus ist zunächst die untere Grenze eingestellt. Ein Druck auf Taster S3 weist den Scanner an, die Empfangsfrequenz so lange selbsttätig zu erhöhen, bis ein Empfangssignal die Squelch-Schwelle übersteigt. In diesem Fall leuchtet LED D3 auf, der Scanner bleibt auf der gefundenen Frequenz stehen. Wird S3 noch einmal gedrückt, verlischt LED D3, der Scanner setzt die Suche fort. Beim Erreichen der oberen Bereichsgrenze folgt ein Rücksprung zur unteren Bereichsgrenze.

Search-Modus:
Wenn Taster S2 nach dem Einschalten mindestens eine Sekunde gedrückt wird, leuchtet LED D3 auf. Der Scanner befindet sich nun im Such-Modus. Beim Abtasten des Flugfunk-Frequenzbands werden konstante Trägersignale ausgeblendet, die Suche wird im Frequenzraster 25 kHz durchgeführt. Vor Beginn des Suchvorgangs ist die Squelch-Schwelle mithilfe von LED D4 auf einen Wert einzustellen, der an die Empfangsbedingungen angepasst ist.
Die Anzahl der Durchläufe durch das gesamte Band ist mit Kommando P im Terminalprogramm auf dem PC festlegbar. Ein einzelner Search-Vorgang von der unteren zur oberen Bandgrenze dauert rund sieben Minuten. Ein bewährter Wert für P ist 6, das sechsfache Durchlaufen des Bands nimmt $6 \cdot 7 = 42$ Minuten in Anspruch. Optimale Suchergebnisse sind mit P = 15 zu erwarten, allerdings beträgt die Wartezeit eine Stunde und 45 Minuten.
Wenn der Suchvorgang abgeschlossen ist, verlischt LED D3, die gefundenen Frequenzen sind gespeichert. Jetzt muss der Flugfunk-Scanner neu gestartet werden. Im Speicher wurden N Frequenzwerte beginnend mit Speicherplatz M abgelegt (vergleiche Kommandos M und N). Bevor der Scanner die gefundenen Frequenzen durchlaufen kann, muss der Mikrocontroller die Werte in den Frequenzspeicher übertragen. Dies geschieht erst, nachdem die LED D3 verloschen ist.

stellen. Wenn im Terminalprogramm höhere Werte erscheinen, muss L3 geringfügig zusammendrückt werden, sind die Werte niedriger, ist L3 um einen geringen Betrag auseinander zu ziehen. Die vom Terminalprogramm ausgegebenen Werte müssen nach dem Kommando T exakt mit den eingegebenen Werten übereinstimmen.

Der Quell- und Hex-Code der Mikrocontroller-Software steht kostenfrei auf der Projektseite im Web [2] zum Download bereit. Die Platine und der bereits programmierte Mikrocontroller sind ebenfalls über diese Website erhältlich.

(100696)gd



Weblinks

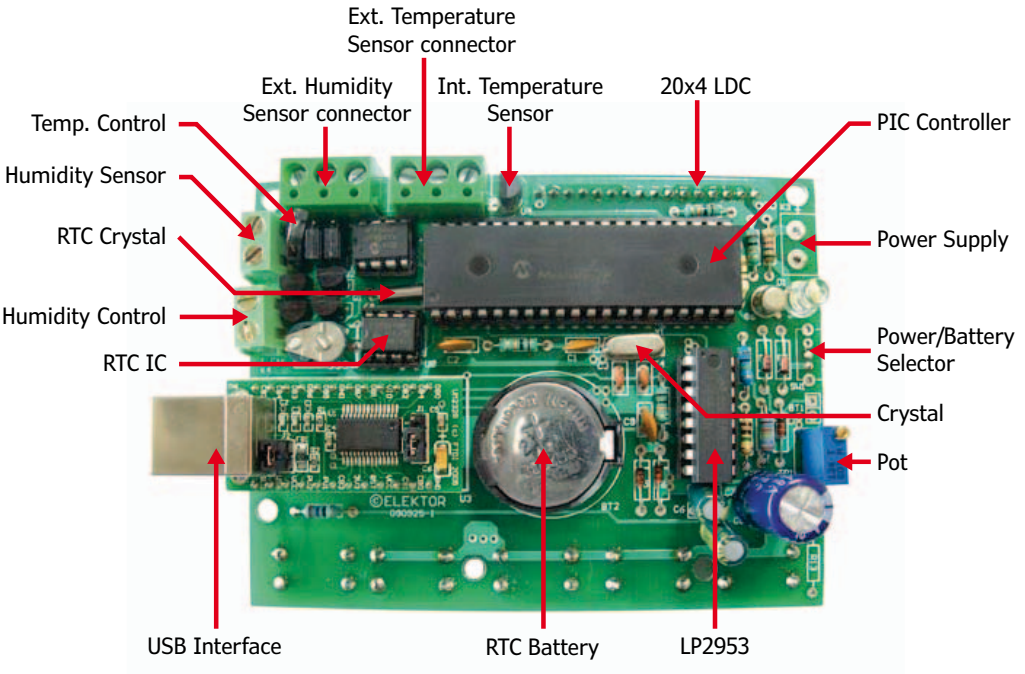
[1] www.elektor-magazine.de/110553
[2] www.elektor-magazine.de/100696

Tropen oder Arktis?

Keine Bange! Dieses Hygro/Thermometer merkt sich alles!

Von
Manoel Conde de Almeida
(Brasilien)

Projekte mit digitalen Thermometern oder Thermostaten gibt es in Elektor seit Jahrzehnten. Dieses hier jedoch kann mit nie dagewesenen, sehr interessanten Features glänzen, wie man sie ansonsten nur bei professionellen Geräten vorfindet.



Das wichtigste Ziel dieses Projekts ist ein Gerät, das Temperatur und relative Feuchte messen kann. Der endgültige Entwurf umfasst aber neben diesem Hauptziel eine Reihe von hochinteressanten Eigenschaften. Wenn die Werte der Temperatur und der relativen Feuchte ermittelt sind, behält das Gerät auch die beobachteten Mini- und Maximalwerte im Auge. Es vergleicht jeden Messwert mit vom Benutzer definierten Grenzwerten und aktiviert bestimmte Ausgänge, wenn diese Grenzen unter- oder überschritten werden. So können externe Schaltungen gesteuert werden.

Die Temperatur wird entweder in Grad Celsius oder Fahrenheit ermittelt. Die Festlegung der Mini-/Maximalwerte für Temperatur und relative Feuchte wird vom PC aus vorgenommen. Die dazu erforderliche PC-Software steht frei zum Download zur Verfügung [1].

Eine Echtzeituhr fügt den Temperatur- und Feuchtedaten Uhrzeit und Datum hinzu. Dann werden die Daten im controller-eigenen EEPROM abgelegt. Temperatur und Feuchte werden in 7-Sekunden-Intervallen ermittelt. Bis zu 240 Messwerte (Temperatur und Feuchte) können

in Intervallen gespeichert werden, die man von 1 bis 99 Minuten im 1-Minuten-Abstand festlegen kann.

Steuerung

Das „User-Interface“ besteht aus einem LC-Display mit 20x4 Zeichen und einem Satz von sieben Schaltern (fünf Taster und zwei Schalter). Damit erhält man Zugriff auf alle konfigurierbaren Parameter. Das Gerät kann auch über den USB des PCs gesteuert werden. Die PC-Applikation [1] kontrolliert alle Funktionen des Geräts, stellt die Datensammlung in einer kleinen Grafik dar und speichert die Daten im .csv-Format (comma separated value), so dass sie leicht von Tabellenkalkulationsprogrammen importiert werden können.

Hardware-Highlights

Die Schaltung in **Bild 1** ist um einen Mikrocontroller PIC18F4525 von Microchip [2] aufgebaut. Er arbeitet mit einer Taktfrequenz von 8 MHz, die von einem Oszillator mit Quarz X1 und den Kondensatoren C3 und C4 erzeugt wird. Der Reset-Schaltkreis besteht aus R1, C2 und dem Taster S2. Das 20x4-LCD ist am Port D des Mikrocontrollers angeschlossen und arbeitet im 4-bit-Kommunikationsmodus. Mit dem Taster S1 aktiviert man die Hintergrundbeleuchtung des LCDs, an TP1 mit R13 lässt sich der Kontrast einstellen.

U3, ein UM232R-Modul, basiert auf einem FT232R-Chip von FTDI [3]. Das Modul enthält ein USB-nach-UART-Interface, das an den Ports C.6 und C.7 (die als Tx und Rx des Controller-eigenen UART konfiguriert sind) ein serielles Interface realisiert. Das Modul wird vom USB versorgt, um die wertvolle Batterieenergie zu schonen.

Die RTC (Real Time Clock) ist mit U8, einem PCF8583 von NXP [4], aufgebaut. Der Trimmer C8 und der 32,768 kHz-Uhrenquarz bilden den Oszillator. Die Kommunikationsleitungen SCL und SDA sind an den Ports C.18 und C23 angeschlossen. Die Drucktaster S3, S4 und S5 ermöglichen den Zugriff auf die Menüfunktionen, die im LCD angezeigt werden. Die Transistoren Q1…Q4 und die Widerstände R2…R5 stellen Open-Collektor-Ausgänge dar, die von den Ports B.4…B.7 gesteuert werden. Die Ports werden aktiviert, wenn die Alarm-Funktion aktiviert ist und der Temperatur/Feuchtewert die vorgegebenen Grenzen überschreitet. **Tabelle 1** zeigt, welcher Ausgang welchem Alarm entspricht.

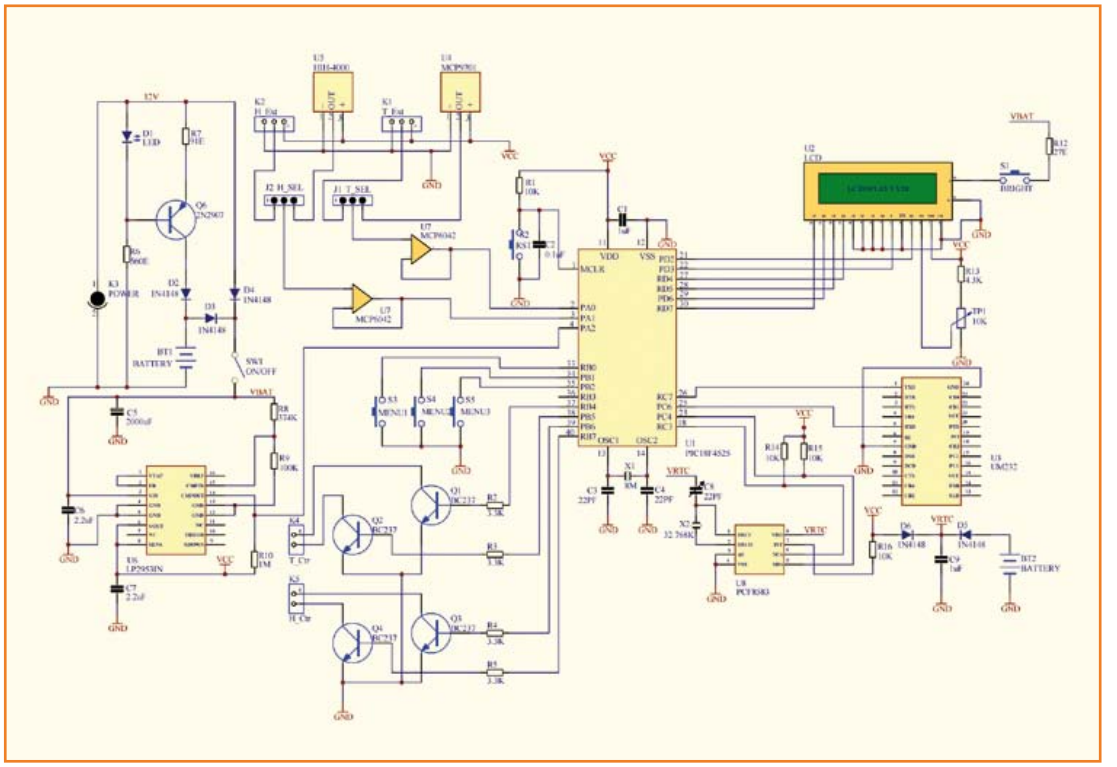


Bild 1.
Die vollständige Schaltung mit sieben leicht auszumachenden Teilen wie RTC und Batterie-Lader.

Eigenschaften

- PIC18F4525 (DIP-40-Pin) Mikrocontroller
- 20x4 (16-Pin) LC-Display
- US232 (USB-nach Seriell-Konverter) von FTDI
- MCP9701 Temperatursensor
- HIH4000 Feuchtesensor
- LP2953 Spannungsregler
- PC8583 Echtzeituhr
- Geeignet für 9-V-Akku, von der Schaltung geladen

Tabelle 1. Alarmausgänge

Portpin	Alarm
B.4	Temperatur Maximum
B.5	Temperatur Minimum
B.6	Feuchte Maximum
B.7	Feuchte Minimum

Die Schaltung kann von einem 9-V-NiMH-Akku versorgt werden, der seinerseits von einem 12-V_{DC}-Steckernetzteil an K3 wieder aufgeladen wird. Transistor Q6, Diode D1 und die Widerstände R6 und R7 stellen eine Konstantstromquelle dar, die für einen Ladestrom von etwa 10 mA vom Steckernetzteil zum Akku berechnet ist. Die Dioden D2…D4 sorgen für die saubere Verbindung der Betriebsspannung zur Schaltung. Ist das Steckernetzteil angeschlossen, fließt der Strom über D2 zum Akku und über D4 zur Schaltung, während D3 die höhere Betriebsspannung blockt. Damit kann der Akku über die Stromquelle geladen und die Schaltung direkt vom Steckernetzteil versorgt werden. Ist das Steckernetzteil nicht angeschlossen, so sperren D2 und D4, der Batteriestrom fließt über D3. Die Stromquelle ist abgeschaltet und die Schaltung wird von der Batterie versorgt.
Schalter SW1 verbindet die Batterie beziehungsweise das Steckernetzteil mit der Schaltung. Für die Stabilisierung der Betriebsspannung auf 5 V sorgt U6, ein Low-drop-Spannungsregler von National Semiconductor. Der Spannungsteiler R8/R9 führt einen definierten Anteil der Ausgangsspannung zum VIN-Eingang des Reglers zurück. Die Spannung wird intern mit einer Referenz verglichen, sodass bei einer Batteriespannung unter 6 V der Komparatorausgang an Pin 14 und damit Port A.2 auf Low geht. Die Firmware erkennt dies und sorgt für eine Low-Bat-Anzeige im Display. Die gesamte Stromaufnahme der Schaltung ist auf 10 mA begrenzt, was - abhängig von der Kapazität - für ungefähr 15 Stunden Dauerbetrieb mit einer vollgeladenen Batterie ausreicht. Die RTC wird von einer Back-up-Schaltung versorgt, wenn das Gerät ausgeschaltet ist. Der Back-up-Schaltkreis besteht aus BT2, einer 3-V-Lithiumbatterie, den Dioden D5 und D6 sowie dem Kondensator C9. Ist SW1 ausgeschaltet, sperrt D6 und BT2 versorgt die RTC über D5, ist dagegen SW1 eingeschaltet, sperrt D5 und der Strom zum RTC fließt von V_{CC} über D5.

Sensor-Spezifikationen

Verantwortlich für die Temperaturmessung ist ein *Linear Active Thermistor™* IC von Microchip mit der Bezeichnung MCP9701. Dieses dreibeinige IC erzeugt eine Spannung, die proportional zur Temperatur ist:

$$V_t = 0.5 + 0.01T$$

wobei V_t die Ausgangsspannung in Volt bei gegebener Temperatur T in Celsius ist.

Der Thermometerausgang wird von U7A gepuffert, einer von zwei Low-power-Opamps des Typs MCP6042 von Microchip. Beide Opamps sind als Spannungsfolger geschaltet.

Die Messung der relativen Feuchte wird vom analogen Hygrometer HIH-4000 von Honeywell [5] vorgenommen. Auch dieser Sensor erzeugt eine analoge, der Feuchte linear proportionale Spannung nach folgender Formel:

$$V_h = 0.7617 + 0.0297H$$

wobei V_h die Ausgangsspannung in Volt für eine gegebene relative Feuchte H (%rh) ist.

Der Hygrometerausgang wird von Opamp U7B gepuffert. Beide Opamps sind an den Ports A.0 und A.1 angeschlossen, die als 10-bit-A/D-Wandler konfiguriert sind.
Die Verbinder K1 und K2 ermöglichen den Anschluss von externen Temperatur- und Feuchtesensoren (gleiche Typen wie U4 und U5). Mit den Jumpern J1 und J2 wählt man zwischen internen und externen Sensoren. Mit dem PC-Applikationsprogramm kann man die Gleichungen der Sensoren leicht anpassen.

Firmware-Fakten

Um die Hardware so geradlinig wie möglich zu halten, übernimmt die Firmware den Löwenanteil am Handling der komplexen Gerätefunktionen. Sie wurde mit MikroBasic v7.2 von Mikroelektronika [6] entwickelt. Das Flussdiagramm in **Bild 2** beschreibt in Kürze die Hauptroutine des Programms, ein Großteil steckt aber in Subroutinen, die hier nicht beschrieben werden können.
Das Hauptprogramm beginnt mit einer Reihe von Initialisierungen einschließlich der Konfiguration der internen Register des Mikrocontrollers, des LCD-Moduls, des EEPROMs mit Standard-Konfigurationswerten (wenn nicht anders programmiert), der Programmvariablen sowie dem Zurücksetzen des TIMER0 und der Freigabe des TIMER0-Interrupts.
Nach den Initialisierungen fällt das Programm in eine Endlosschleife (die „On“-Schleife). In dieser Schleife läuft die Routine durch folgende Zustände:
Check der Batteriespannung und Ein-/Ausschalten des LowBat-Indikators,
Prüfen, ob sich die Zeit geändert hat, dann folgt eine entsprechende Änderung der Zeit/Datumanzeige im Display,
Prüfen, ob neue Messdaten zur Verfügung stehen und Displayinhalt entsprechend ändern,
Die Taster abtasten, um User-Anfragen für Display- oder Funktionsänderungen zu erkennen und danach entsprechend in das Programm springen,
Daten aufzeichnen, wenn die Datenaufzeichnung freigegeben ist,
Prüfen, ob der PC angeschlossen ist, um jede empfangene Anweisung auszuführen.

Obwohl nicht im Flussdiagramm beschrieben, ist es doch wichtig zu bemerken, dass zeitabhängige Aktivitäten vom Programm durchgeführt werden, die auf TIMER0-Overflow-Interrupts basieren. Wer am Aufbau der komplexen Programmstruktur interessiert ist, kann den kompletten Quellcode studieren, der in der Projektdokumentation [7] zu finden ist.

PC-Programm

Die PC-Applikation wurde unter Visual Basic 2008

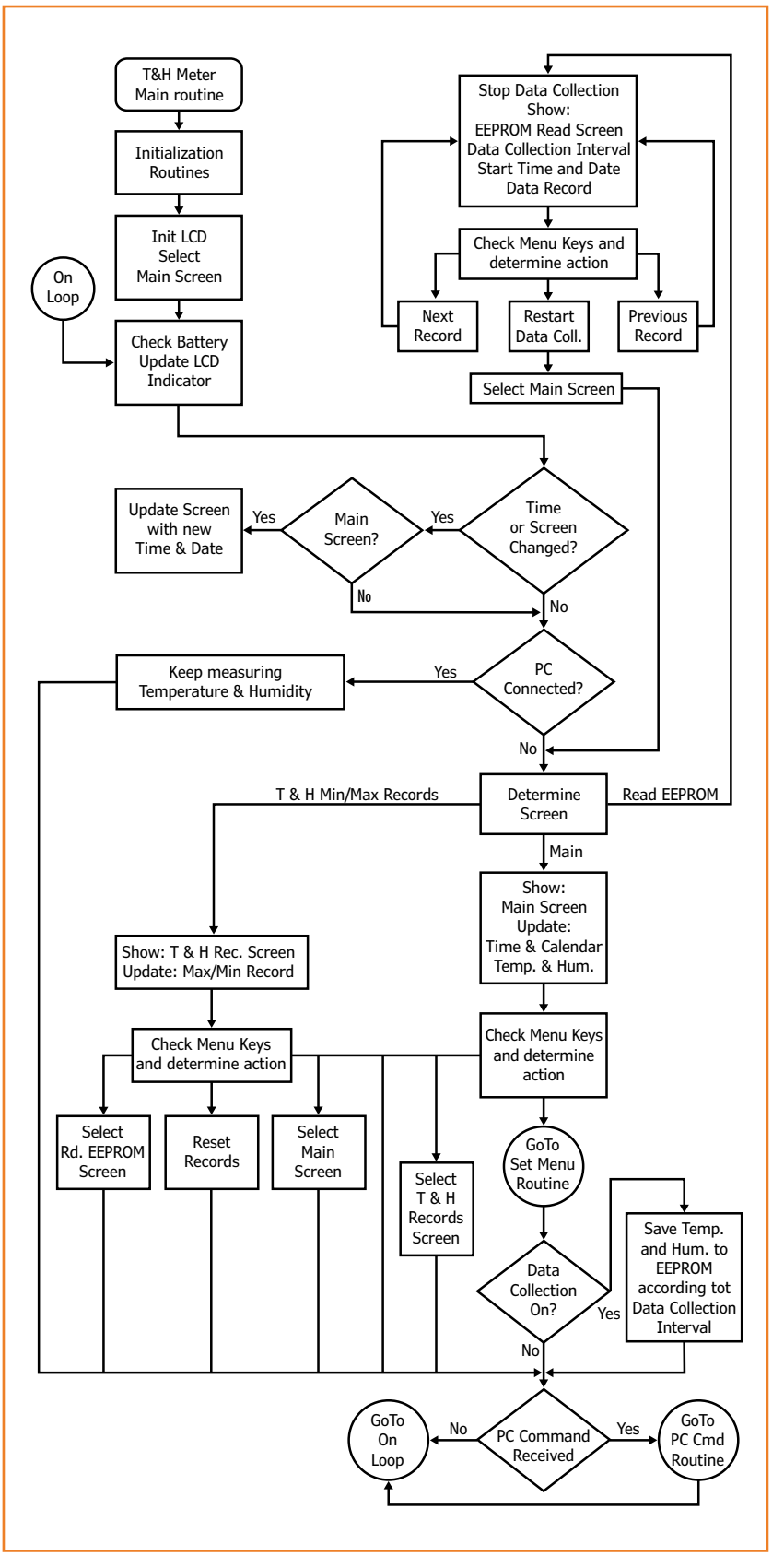


Bild 2.
Das Flussdiagramm zeigt die Hauptroutine, die vom Mikrocontroller ausgeführt wird.

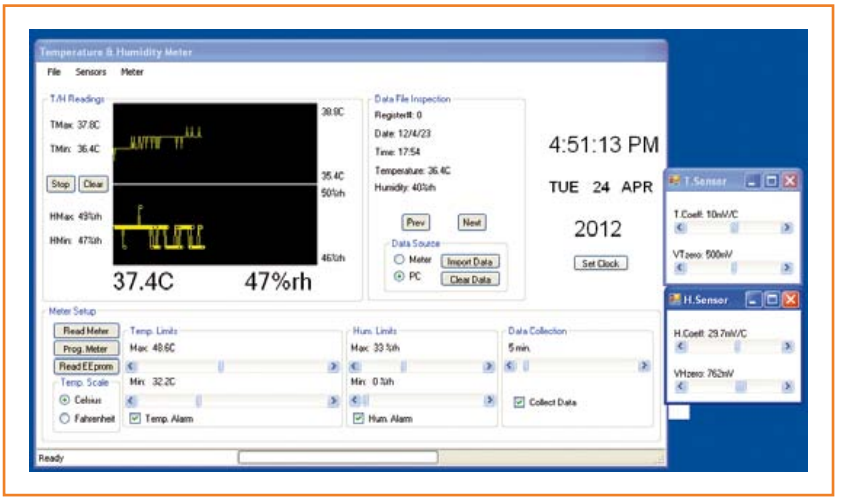


Bild 3.
Das Graphical User Interface ermöglicht einen einfachen Zugriff auf verschiedene Einstellungen des Feuchte/Temperaturmessers.

Express Edition entwickelt, die Microsoft kostenlos vertreibt. Die Applikation kann hier nur kurz umrissen werden, für mehr Details, eine ausgearbeitete Beschreibung der Softwareinstallation und Funktionsweise, konsultieren Sie bitte das Online-Dokument unter [1].

Um die Kommunikation zwischen Messgerät und PC über die USB-Schnittstelle zu realisieren, kommt der auf der FTDI-Website [8] kostenlos verfügbare D2XX-Treiber zum Einsatz. **Bild 3** zeigt das Hauptfenster des PC-Programms. Das Menü *File* ermöglicht dem User, die .csv-Dateien (Datensammlungen) zu bearbeiten (clear, copy). Das *Sensor*-Menü erlaubt die Feinjustierung der Parameter des Temperatur- wie auch des Feuchtesensors. Im *Meter*-Menü kann man das Gerät mit der PC-Applikation verbinden oder die Verbindung unterbrechen.

Das Fenster *T/H Readings* zeigt die Temperatur- und Feuchte-Messwerte im Format einer Liniengrafik. Links davon sind die höchsten und niedrigsten gemessenen Werte gelistet, in großer und fetter Schrift unter dem Fenster die aktuell gemessenen Werte. Die Echtzeituhr des Messgeräts kann mit einem Klick auf *Set Clock* mit der PC-Uhr synchronisiert werden.

Unter *Data File Inspection* kann man den Inhalt der .csv-Datensammlungsdateien lesen. Die Daten werden in den File-Inspector geladen, indem man die gewünschte Quelle aussucht und auf den *Import*-Knopf drückt. Mit den Knöpfen *Prev* und *Next* bewegt man sich durch die Datensätze. Die *Clear*-Taste löscht die Datenarrays im File-Inspector, lässt aber die originalen Datensätze unangetastet.

Die Abteilung *Meter Setup* fasst alle Möglichkeiten zusammen, mit denen man die Betriebsparameter des Messgeräts und des Programms einstellen kann. Jedes Mal, wenn das Programm gestartet wird und Verbindung mit dem Gerät aufnimmt, wird dessen Konfiguration automatisch heruntergeladen. Alle Steuerungen sind dann im Einklang mit den importierten Setup-Daten. Modifikationen des aktuellen Setups ändern den Betrieb der Applikation unmittelbar. Damit Änderungen (außer der Temperaturskala) vom Messgerät übernommen werden, klickt man einmal auf den *Prog. Meter*-Knopf und das neue Setup wird hochgeladen.

Die Applikation zeigt die eingestellten Maximal- und Minimalwerte für Feuchte und Temperatur (*Temp.limits* und *Hum. limits*) und markiert, wenn die Grenzen verletzt wurden, den entsprechenden Messwert im Grafikmonitor. Die Checkboxen *Temp. Alarm* und *Hum. Alarm* schalten diese Funktion ein und aus. Der Schieberegler *Data Collection* bestimmt das Zeitintervall der Messung, die in der Checkbox eingeschaltet wird.

Wenn die Datensammlung eingeschaltet ist, veranlasst die Applikation das Messgerät, im vorgegebenen Intervall Messungen der Temperatur und der relativen Feuchte vorzunehmen und in der Datei pc_cd.csv zu speichern. Für das Messgerät bedeutet dies, die Daten zu sammeln und im EEPROM abzulegen.

Wie bei jedem in Elektor beschriebenen Projekt kann sowohl die Firmware für den Controller als auch die Applikation für den PC von der Website des Projekts heruntergeladen werden [1]. Auch die Bauteilliste, das Platinenlayout sowie zahlreiche Hinweise zur Installation und Anwendung der Software sind online. Auf der Elektor-Projekts-Site [7] können Sie auch Verbesserungen und/oder Kommentare zu dem Temperatur/Hygrometer anfügen.

(090925)

Weblinks

[1] www.elektor-magazine.de/090925
[2] www.microchip.com
[3] www.ftdichip.com
[4] www.nxp.com
[5] http://sensing.honeywell.com
[6] www.mikroe.com/mikrobasic
[7] www.elektor-projects.com/090925
[8] www.ftdichip.com/FTDrivers.htm

Second Step

NEU!

Sie haben den „First Step“-Kurs erfolgreich durchgearbeitet und haben Lust auf mehr bekommen?
Dann machen Sie doch einfach den nächsten Schritt in die große Welt der Mikrocontroller!
Aufbauend auf Ihrem bisherigen Wissen beschäftigen Sie sich beim Fortsetzungskurs „Second Step“ mit weiteren interessanten und wichtigen Mikrocontroller-Anwendungen:
Steuern Sie zum Beispiel ein alphanumerisches LCD an, oder erweitern Sie Ihr System um einen Uhrenbaustein (Real Time Clock), oder lernen Sie eine doppelte UART-Schnittstelle und einen Watch Dog kennen und nehmen diese Bausteine in Betrieb!

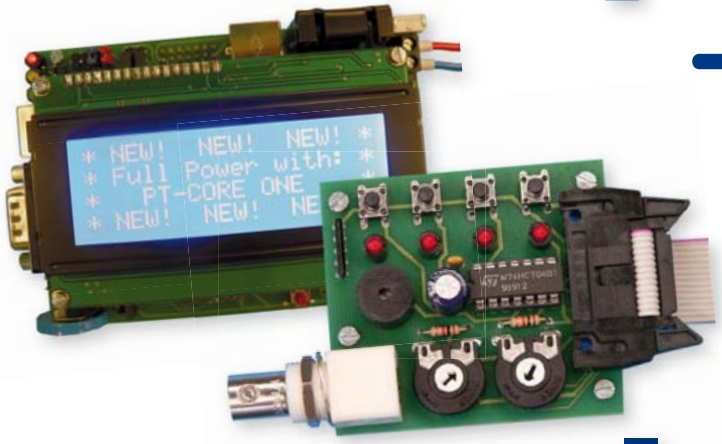
Bestandteile des Second Step -Pakets:

elektor ACADEMY
the school of electronics

3 Arbeitshefte

(inkl. passendem DIN A4-Ringbuch)

- Beschreibung der neuen Hardware
- Selbst geschriebene Funktionen, lokale und globale Variablen und Header-Dateien
- Externe parallele Peripherie-Einheiten: RTC, LCD und DUART
- Strings und Arrays
- Watch Dog und CS\-Logik
- Serielle Datenübertragung
- Umfang aller Arbeitshefte: 450 Seiten



1 PT-CORE-Mikrocontroller-Board + 1 PT-DAA-Zusteckkarte

- 8051er-Mikrocontroller AT89C51CC03
- 2,5-V-Referenzspannungsgeber für A/D-Wandler LT1009
- TTL/RS232-Pegelwandler MAX232
- CAN-Buskoppelstufe PCA82C250
- Watch Dog MAX807
- Uhrenbaustein mit Batteriepufferung RTC72421
- Alphanumerisches LC-Display (4x20 Zeichen)
- Doppelter UART-Baustein SCC2681
- Chip-Select-Dekodierung inkl. 4 freier CS\-Signale GAL16V8
- Datenlatch 74HCT573
- Karten-Format: 117 x 83 mm²
- Spannungsversorgung 9 V DC, Verpolungsschutzdiode und Miniatursicherung
- PT-DAA-Zusteckkarte enthält den kompletten optischen, akustischen und analogen Funktionsumfang des „First Step“-Boards

1 CD-ROM mit Zusatzinfos

- Datenblätter
- Systemdokumentation
- Entwicklungsumgebung
- Beispielprogramme

Das gesamte Second Step -Paket kostet nur 249,00 Euro.

Weitere Infos und Bestellung unter
www.elektor.de/second-step

Kapazitiver Näherungsschalter

Verschleißfreier Taster

Von
Martin Jepkens,
Dr. Thomas Scherer
und **Daniel Wunsch**
(D)

Beim Selbstbau von selbstbalancierenden Elektrorollern steht man vor dem Problem, dass zuverlässig erkannt werden muss, ob der Fahrer auf der Plattform steht oder nicht. Mechanische Lösungen konnten nicht richtig überzeugen. Ein rein elektronischer Fahrerdetektor in Form eines kapazitiven Näherungsschalters hingegen schon. Dieser kann natürlich noch für alle möglichen Anwendungsfelder adaptiert werden.

Die drei Autoren bauen an eigenen Versionen eines Wheelie, bzw. an selbstbalancierenden Zweirädern nach der Art eines Segway-Scooters. Damit der Scooter nicht ohne Fahrer losfahren kann, und damit er stoppt, falls der Fahrer absteigt (oder abgestiegen wird), muss zuverlässig detektiert werden, ob ein Fahrer auf der Plattform steht, oder ob der Scooter plötzlich solo ist. Die Firma Segway setzt hierbei auf mechanische Taster, die vom Gewicht des Fahrers betätigt werden.

Ähnliche Lösungen werden weltweit von Selbstbauern bevorzugt, doch so gut und zuverlässig wie das industriegefertigte Vorbild ist die Sache im Selbstbau nicht so leicht hinzubekommen. Fertige Taster sind entweder teuer, nicht wasserdicht, mechanisch nicht stabil oder schlicht nicht schön genug. Von daher drängte sich die Idee auf, den Fahrer kapazitiv zu detektieren.

Kapazitive Methoden

Zunächst wurde probiert, die Durchbiegung des oberen Gehäuseblechs zu nutzen und einen Mikroschalter zu betätigen, wenn sich das Blech aufgrund der Fahrerlast ausreichend biegt. Gute Mikroschalter benötigen hierfür nur etwa 1,5 mm Hub. Dass die Mikroschalter dann mechanisch justierbar angebracht sein sollten, machte die Sache schon wieder komplizierter als gewünscht. Eine sich biegende Metallplatte gibt aber einen super variablen Kondensator ab, wenn die zweite Kondensatorplatte fix montiert ist. Man muss also lediglich die sich ändernde Kapazität messen, wenn ein Fahrer auf der oberen Platte steht und die Sache ist gegessen. Notwendige Justierungen können dann nicht nur elektronisch, sondern auch gleich automatisch vorgenommen werden. Letzteres dann, wenn man zum Messen der Kapazität einen kleinen Mikrocontroller verwendet.
Soweit so einfach. Der Teufel lag hier aber wie so oft im Detail. Kapazitäten kann man ja bekanntlich auf viele verschiedene Arten messen. Man kann die Plattenkapazität als frequenzbestimmendes Bauteil eines astabilen Multivibrators (AMV) einsetzen. Man könnte auch den Ladestrom des Plattenkondensators erfassen. Oder man kann ein erprobtes und zuverlässiges Verfahren nutzen, das etwas komplizierter und patentiert ist, dafür aber stabile und zuverlässige Resultate liefert.
Der µC-AMV nach **Bild 1** ist eine sehr simple Sache: Über den Ausgang Pin A und einen Widerstand R wird die zu messende Kapazität Cx geladen und entladen. Da der Eingang (Pin B) üblicher Mikrocontroller eine Hysterese ähnlich einem Schmitt-Trigger aufweist, fällt die notwendige Firmware ebenfalls simpel aus: Ist die Spannung

am Eingang Pin B „low“, wird der Ausgang Pin A „high“ - ist der Eingang Pin B „high“, wird der Ausgang Pin A wieder „low“. An Pin B liegt somit ein Rechtecksignal mit von Cx abhängiger Frequenz. Man muss jetzt nur noch per Timer die Frequenz oder die Schaltzeiten messen und hat dann ein Maß für die Kapazität. Wenn jemand auf der Plattform steht, wird der Abstand zwischen der Deckplatte und der festen inneren Platte kleiner, demzufolge steigt die Kapazität und die Frequenz sinkt. Das kann man feststellen.

Getreu dem KISS-Prinzip („Keep it simple, stupid!“) wurde zunächst mit der Mikrocontrollerausführung eines astabilen Multivibrators experimentiert, bei dem die sich ergebende Frequenz bzw. die resultierenden Zeiten gemessen wurden. Doch leider war hier die Theorie besonders grau: Viele Experimente führten zum Schluss, dass es so einfach dann doch nicht geht. Die als Sensor dienende Gegenplatte mit ihren ca. 15 cm^2 kommt bei 1 mm Abstand zum sich biegenden Trittblech auf nur einige zehn pF. Um die sich ergebende Frequenz niedrig genug zu halten, muss R einen Wert im MΩ-Bereich haben. Der Eingang unterscheidet sich dann nicht mehr viel von einem offenen Eingang – jede Störung wirkt sich aus und nicht nur die Zuverlässigkeit fehlt, auch die Sensitivität ist in der Praxis nicht wirklich brauchbar.

Doch man muss das Rad ja nicht unbedingt neu erfinden. Die Firma Quantum hat sich nämlich ein gut funktionierendes Verfahren namens QTouch patentieren lassen. Da sich Atmel diese Firma vor ein paar Jahren samt QTouch einverleibt hat und sie für die eigenen Mikrocontroller zur Verfügung stellt, spricht nichts gegen die Übernahme dieser erprobten Technik.

QTouch

Das „Q“ im Namen spielt nicht nur auf Quantum an, denn Q ist gleichzeitig das Formelzeichen für die elektrische Ladung – und genau um die geht es beim QTouch-Verfahren. Prinzipiell wird dabei elektrische Ladung vom Sensor-Pad Cx in einen größeren Kondensator umgeschaufelt. Was man dazu braucht, das ist überraschend einfach. Im Prinzip muss man nur den Widerstand R von Bild 1 durch einen Kondensator C_L ersetzen und schon landet man bei der Prinzipschaltung von **Bild 2**. Das Prinzip ist schnell erklärt: Über die relativ kleine Kapazität Cx wird die relativ große Kapazität C_L zyklisch geladen, bis C_L „voll“ ist. Dabei werden die nötigen Zyklen gezählt. Zum Schluss wird C_L wieder entladen und die Sache beginnt

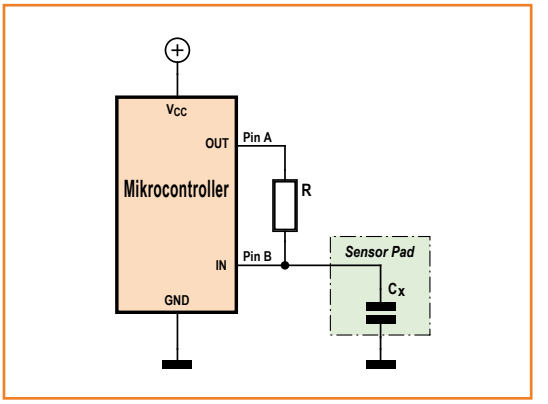


Bild 1.
Prinzipschaltung eines astabilen Multivibrators per Mikrocontroller, bei dem die Frequenz durch die Sensor-Kapazität Cx beeinflusst wird.

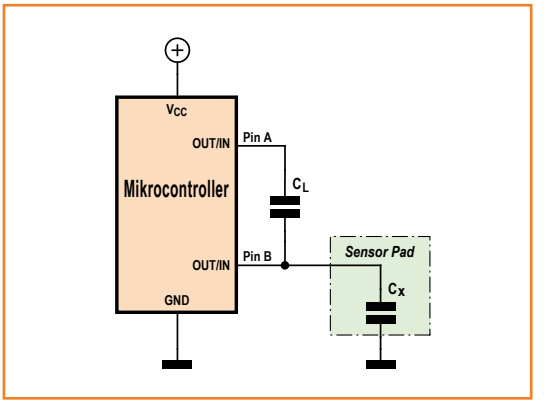


Bild 2.
Prinzipschaltung des QTouch-Verfahrens mit einem Mikrocontroller. Gegenüber Bild 1 ist R durch C_L ersetzt und der Eingang wird zeitweise auch als Ausgang geschaltet.

von vorne. Ganz klar wird das, wenn man sich den **Pseudo-Code** im Kasten anschaut. Die einzige (lapidare) Voraussetzung für den Mikrocontroller ist, dass er seine Pins im Betrieb zwischen Ein- und Ausgang umschalten kann. Da die Kapazi-

QTouch-Pseudo-Code

Am Ausgangspunkt sind C_L und Cx entladen (Zeile 2).

```
 1  Pin A und Pin B = out,Pins als Ausgang
 2  Pin A und Pin B = low,Entladen von CL und Cx
 3  Zyklen = 0 ,Löschen des Zyklenzählers
 4  Pin B = in ,Pin B als Eingang
 5  Pin A = high ,CL wird über Cx etwas geladen
 6  Zyklen = Zyklen + 1 ,Inkrement des Zyklenzählers
 7  If Pin B = low then goto 1 ,CL ist voll geladen, Neustart
 8  Pin A = in ,Pin A wird hochohmig
 9  Pin B = out  ,Pin B als Ausgang
10  Pin B = low  ,Cx wird entladen
11  goto 4 ,Nächster Zyklus
```

Wenn in Zeile 7 C_L geladen ist (Pin B = low), hat die Variable „Zyklen“ den der Kapazität von Cx entsprechenden Wert erreicht.

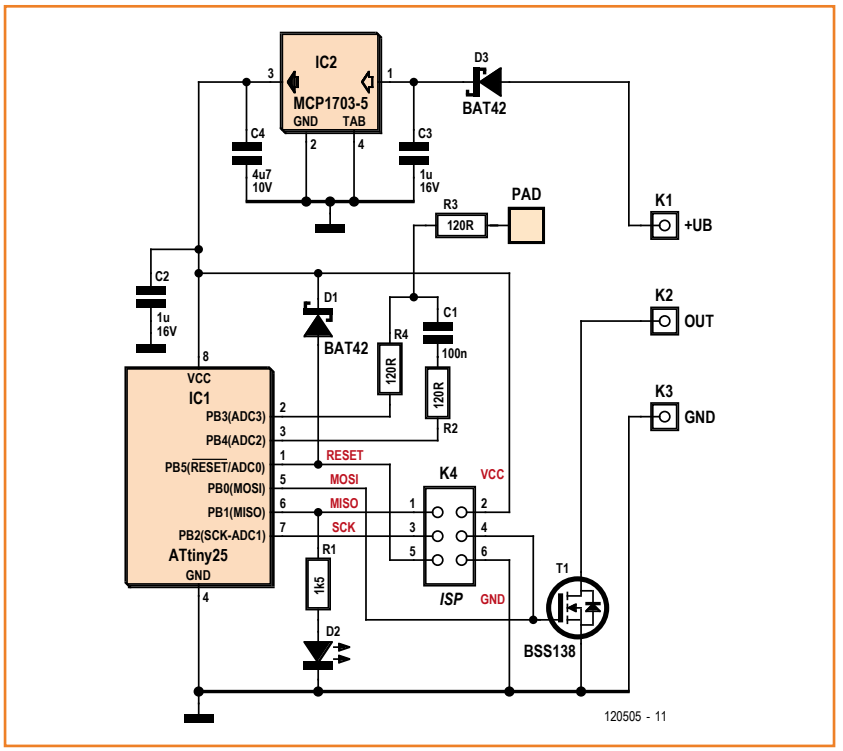


Bild 3.
Die Schaltung des kapazitiven Näherungsschalters nach dem QTouch-Verfahren.

tät von C_L erheblich größer als bei Cx ist, fällt die Impedanz an Pin B beim Messen des Pegels deutlich niedriger aus als bei Bild 1. Das QTouch-Verfahren ist daher ziemlich immun gegenüber Störsignalen.

Schaltung und Aufbau

Die konkrete Schaltung von **Bild 3** ist nicht viel komplexer als das Prinzip von Bild 2. C1 übernimmt die Rolle von C_L und ist mit seinen 100 nF wirklich sehr viel größer als die Kapazität des Sensor-Pads. Um die Ströme zu begrenzen sind mit R2...R4 noch drei Widerstände eingefügt. PB1 lässt die LED leuchten, wenn die Kapazität des Pads über dem Schwellwert liegt. Gleichzeitig wird T1 über PB0 durchgeschaltet. Der Open-Drain-Ausgang der Schaltung fungiert wie ein Schließer und verträgt problemlos 100 mA. D1 schützt den Reset-Pin vor zu hohen Spannungen. Damit die Schaltung als komplettes Modul z.B. im Elektor-Wheelie als Fußschalter eingebaut werden kann, ist mit IC2 gleich eine Stabilisierung der Stromversorgung auf der Platine untergebracht. D3 fungiert als Verpolungsschutz. Die Versorgungsspannung kann daher 7...14 V betragen.
Bild 4 zeigt den fertig aufgebauten Prototypen. Der sensible Teil rund um C1 ist von Masseflächen umgeben. Das Pad als zweite Kondensatorplatte eines Trittblechs befindet sich auf der Rückseite der Platine (siehe **Bild 5**). Warum? Die Rückseite wird gegen das Trittblech positioniert. Damit sie nicht stören, befinden sich die Bauteile auf der anderen Seite. Aus diesem Grund wird auch der sechspolige Pinheader K4 zur Programmierung des Controllers als SMD-Ausführung aufgelötet. Die Form des Pads ist nicht so entscheidend, lediglich die Fläche sollte in etwa beibehalten werden.
Nach Bestückung, Programmierung und Test sollte die Platine mit transparentem Lack besprüht werden, wenn man sie in ein Fahrzeug einbaut. Dabei aber K4 abdecken! Der Lack verhindert Störungen durch kondensierende Feuchtigkeit etc. Anschließend kann man auf die Rückseite der Platine eine dünne Gummischicht (ca. 0,6 mm Stärke) aufkleben, die quasi als Puffer dient, sollte sich das Trittblech allzu weit durchbiegen. Die freie Fläche der Bestückungsseite wird mit etwas dickerem Gummi (z.B. 1 mm Stärke) beklebt. Die so geschützte „top side“ wird mitsamt Gummischicht auf ein passendes Stück stabil fixiertes Metall als Unterlage geklebt, sodass die Platine quasi als Anschlag für das Trittblech dient.

Varianten und Anmerkungen

Zunächst zur Firmware: Sie ist in Bascom geschrieben und belegt nur etwas mehr als 1/3 des Speichers eines ATtiny25. Source- und Hex-Datei sind über die Elektor-Webseite zu diesem Artikel [1] kostenlos downloadbar. Der Code funktioniert so: Nach dem Einschalten wird zunächst

Über die Autoren
Martin Jepkens ist Elektro-Ingenieur und Geschäftsführer der Firma ME-Engineering, die sich mit der Prozessautomatisierung in der Industrie beschäftigt. Sein aktuelles Freizeitprojekt ist der Bau und die Optimierung eines selbstbalancierenden Scooters.
Thomas Scherer arbeitet schon über 30 Jahre als externer Autor für Elektor. Im Moment tüftelt er ebenfalls an einem selbstgebauten Scooter und staunt, wie komplex das Projekt in all seinen Facetten geworden ist.
Daniel Wunsch hat in früheren Jahren Haushaltsgeräte entwickelt und arbeitet heute bei einem namhaften Automobilhersteller. Er hat den Selbstbau-Scooter mit dem bislang wohl besten Fahrverhalten realisiert.

zur Kalibration die aktuelle Kapazität des Pads gemessen, die ja von der konkreten Einbausituation abhängt. Der Schwellwert, ab dem eine Betätigung des Näherungsschalters erkannt wird, ist im Code auf 2/3 des Einschaltwerts festgelegt. Da der Code ausführlich kommentiert ist, kann man diese Schwelle leicht ändern. Höhere Schwellen machen die Schaltung sensibler - allerdings darf man das nicht übertreiben, wenn die Schaltung störsicher sein soll.

Selbstverständlich kann man den Code auch an andere Mikrocontroller anpassen und für kleinere Pads (z.B. als Sensortasten) anpassen. Kleinere Pads bedeuten kleinere Kapazitäten, weshalb man dann den Wert von C1 proportional verkleinern muss, will man nicht sehr große Zyklenzahlen und eventuell einen Variablenüberlauf riskieren. Man könnte auch mehrere Pads an einen Mikrocontroller anschließen. Dabei ist dann im Prinzip nur ein Pin A für alle Pads und je ein Pin B pro Pad erforderlich.

Beim Einsatz als Fußschalter in einem Wheelie sollte man zwei Platinen verwenden – für jeden Fuß eine. Außerdem muss man schauen, ob das Trittblech nachgiebig genug ist, damit es sich etwa 1 mm bewegt, wenn mehr als 15 kg aufliegen. Schließlich will man ja nicht, dass das Gefährt erst dann aktiviert wird, wenn man mit vollem Gewicht drauf steht. Außerdem ist zu beachten, dass sich der Näherungsschalter beim Einschalten kalibriert. Beim Einschalten sollte also kein Fuß auf der Trittfläche stehen.

Die Schaltung kann ohne Probleme auch in 3,3-V-Technik betrieben werden. Hierzu muss man lediglich für IC2 eine 3,3-V-Version bestücken.

Die verwendete QTouch-Technik ist durch ein Patent geschützt, doch der Patentinhaber Atmel stellt eine QTouch-Library „royalty free“ zur Verfügung (siehe [2]). Die Verwendung der Atmel-Library hätte aber die Verwendung von C nach sich gezogen und den Einsatz kleinster Controller behindert, die ja nicht über so viel Speicher verfügen. Beim Einsatz für private Zwecke greift das Patentrecht nicht, von einer kommerziellen Nutzung unserer Schaltung müssen wir abraten!

Das Kleben der Platine ist beim Wheelie der Befestigung mit Schrauben vorzuziehen, da so keine überstehenden Teile zwischen Trittblech und Platine stören können. Für andere Zwecke hingegen mag eine Verschraubung sinnvoll sein. Dann kann man sich auch die Gummischicht auf der Bestückungsseite sparen.

(120505)

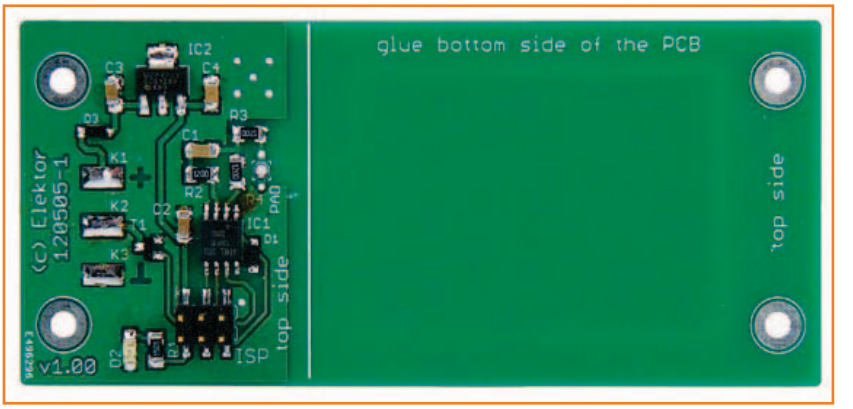


Bild 4.
Der Prototyp des kapazitiven Näherungsschalters (Bestückungsseite).

Stückliste

Widerstände:
(SMD 1206)
R1 = 1k5
R2…R4 = 120 Ω

Kondensatoren:
(SMD 1206, keramisch)
C1 = 100 n/25V
C2,C3 = 1 µ/16V
C4 = 4,7 µ/10V

Halbleiter:
D1,D3 = BAT42W, Schottky, SOD-123
D2 = LED, rot, 1206
IC1 = ATtiny25, SOIC-8, programmiert erhältlich 120505-41
IC2 = MCP1703-5, SOT-223
T1 = BSS138, SOT-23

Außerdem:
K4 = SMD-Pinheader 2x3, RM 1/10”
Platine 120505-1, erhältlich via [1]

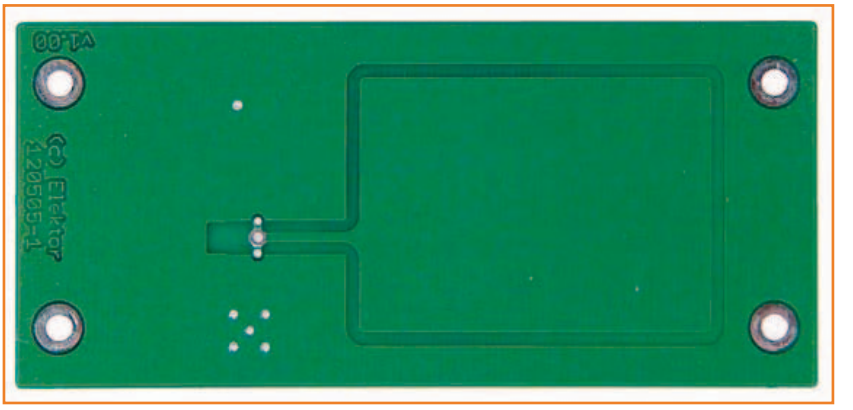


Bild 5.
Die Rückseite der Platine des kapazitiven Näherungsschalters zeigt, wie die Sensorfläche (das Pad) ausgeführt ist.

Weblinks

[1] www.elektor-magazine.de/120505

[2] www.elektor.de/elektronik-news/atmel-studio-6.2115137.lynkx

LED-Ringleuchte

Für Nahaufnahmen mit der Kamera

Von
Vincent Himpe (USA)

Eine Ringleuchte ist eine Lichtquelle, die bei Webcams, Mikroskopen und Fotoapparaten eingesetzt wird. Sie wird rund um die Linse angebracht und soll sehr nah gelegene Objekte gleichmäßig ausleuchten. Ringleuchten werden normalerweise in der Portrait- und Makrofotografie eingesetzt oder kommen bei der Beleuchtung von Objekten unter einem Inspektionsmikroskop zum Einsatz.

Traditionelle Ringleuchten verwenden ringförmige Fluoreszenzröhren, deren Helligkeit man nur mit einem aufwändigen und teuren Fluoreszenz-Dimmer einstellen kann. Dieser Nachteil lässt sich durch den Einsatz weißer Hochleistungs-LEDs umgehen. Die hier gezeigte Schaltung besteht aus einem Array von 36 LEDs, die von einem programmierbaren und speziell zur Ansteuerung von LEDs geeigneten Konstantstrom-Aufwärtswandler gesteuert werden.

Anatomie einer LED

Die oberflächenmontierten LEDs, die hier eingesetzt werden, stammen vom LED-Spezialisten Cree und tragen die Bezeichnung CLA1B-WKW. Sie stecken in PLCC4-Gehäusen (siehe **Bild 1**) und sind etwa 3,2 x 2,8 mm² groß. Sie produzieren einen Lichtstrom von 13,0 Lumen bei einem Abstrahlwinkel von 120 Grad. Die CLA1B-WKW verströmt Licht mit einer Farbtemperatur von typisch 5500 K, ein für Fotoleuchten idealer Wert. Der maximal zulässige Dauer-Durchlassstrom beträgt 80 mA, aber in dieser Anwendung bleiben wir weit unterhalb dieses Wertes.

Die Treiberschaltung

Um die LEDs mit Energie zu versorgen, kommt der LED-Treiber MIC3289 von Micrel zum Einsatz. Dieser Treiber ist im Prinzip ein pulsweitenmodulierter Aufwärtswandler, der aus einer niedrigen

eine hohe Ausgangsspannung erzeugt (**Bild 2**). Der Chip misst den Ausgangsstrom und stellt die Ausgangsspannung stets so ein, dass ein vorgegebener Strom durch die Last (in diesem Fall die LED-Ketten) fließt. Während ein normaler Aufwärtswandler eine konstante Spannung erzeugt, sorgt dieser mit einer variabel angepassten Spannung für einen Konstantstrom durch die Last. Und da LEDs mit einem Konstantstrom arbeiten, ist dieser Wandler die optimale Lösung.

Der MIC3289 besitzt noch weitere Eigenschaften, die bei der Ansteuerung von LEDs von Nutzen sind. So ist eine 16-stufige Stromsteuerung mit logarithmischer Skalierung integriert. Das menschliche Auge nimmt nämlich Helligkeit nicht linear, sondern logarithmisch wahr, sodass diese Stromsteuerung der Empfindlichkeit des Auges nahekommt.

Der MIC3289 benötigt nur wenige externe Bauteile, lediglich ein externer Widerstand zur Strommessung und eine Induktivität sind erforderlich. Der Chip schaltet auf einer sehr hohen Frequenz (1,2 MHz typisch), sodass die erforderliche Induktivität schön klein bleiben kann. Ein digitaler Steuereingang ermöglicht es dem Anwender, die Helligkeit der LEDs zu bestimmen und den Wandler ein-/auszuschalten.

Der Wandler ist durchaus in der Lage, eine so hohe Spannung zu erzeugen, dass die LED-Kette zerstört würde. Im Fehlerfall schaltet deshalb ein eingebauter Spannungsdetektor den Wandler ab, bevor irgendein Unglück passieren kann.

Der Wandler ist in zwei Versionen erhältlich, für 16 V oder 24 V Ausgangsspannung. Da die Schwellspannung der LEDs bis zu 3,8 V beträgt und sechs LEDs in Reihe geschaltet sind, benötigen wir die 24-V-Version (MIC3289-24YD6).

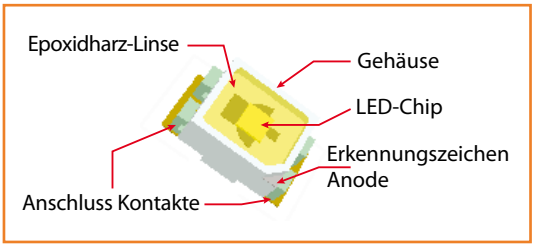


Bild 1.
Die Anatomie einer LED.

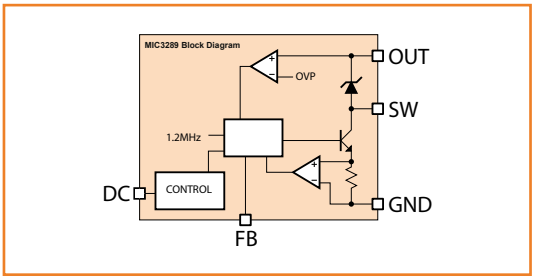


Bild 2.
Internes des LED-Treiber-ICs.

Der MIC3289 bestimmt den Strom anhand des Spannungsabfalls über dem Widerstand am Feedback-Eingang FB. Der maximale Wert wird bei 250 mV am Eingang erreicht. Die Rechnung ist einfach: Über dem 1-Ω-Widerstand fallen 250 mV bei einem Strom von 250 mA ab. Verteilt über vier LED-Ketten ergibt dies ungefähr 42 mA pro Kette, ein Wert weit unterhalb des Limits. Der MIC3289 ist in der Lage, über 500 mA zu liefern, sodass man durch Verringerung des 1-Ω-Widerstands (auf minimal 0,5 Ω) eine noch größere Helligkeit erreichen kann. Achten Sie aber darauf, den MIC3289 nicht in den „thermal shutdown" zu treiben – der Schalttransistor des Chips ist zwar intern geschützt, sodass man ihn eigent-

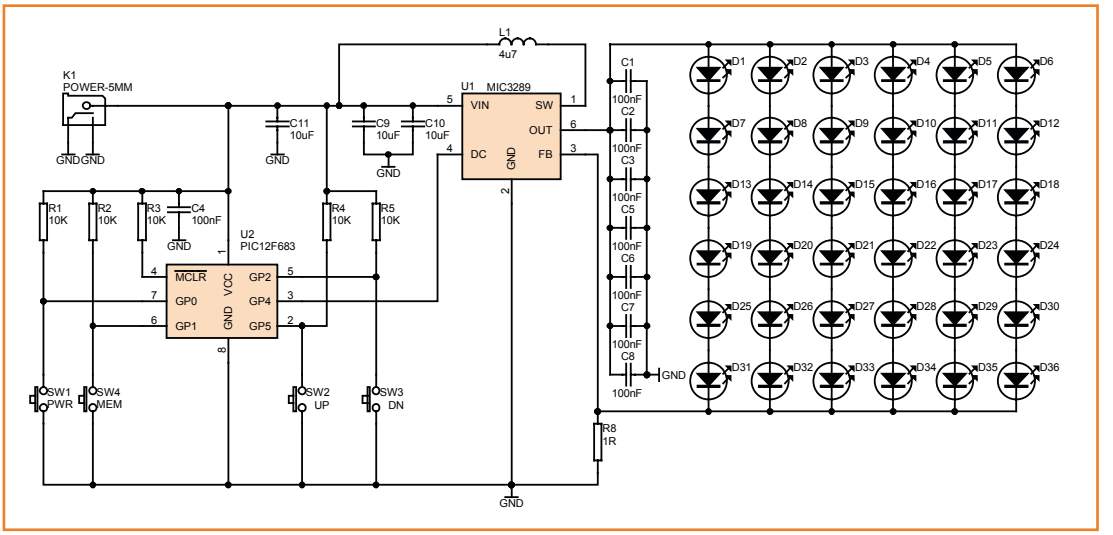


Bild 3.
Das Herz der Schaltung besteht aus U1 und R8, alles andere dient nur der Steuerung (U2, SW1…4), Entkopplung (C1…C11), Beleuchtung (D1…36) und als Pull-ups (R1…R5).

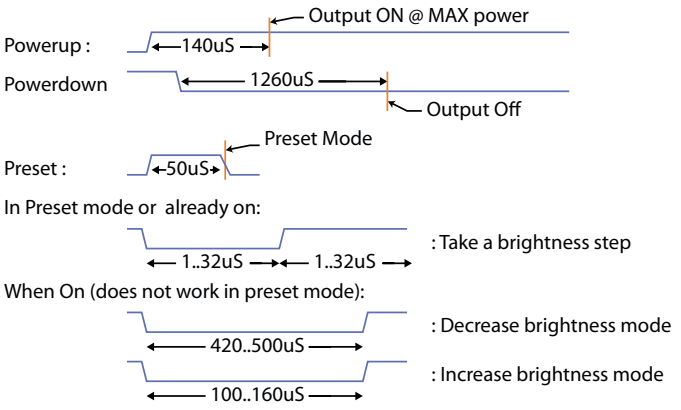


Bild 4.
Timing-Diagramm des MIC3289.

lich nicht zerstören kann, aber man muss es ja nicht herausfordern…

Schaltung

Die Schaltung in **Bild 3** stammt aus dem neuen englischsprachigen Elektor-Buch *Mastering Surface Mount Technology*. U1 sorgt für die rechte Energie, in Zusammenarbeit mit Induktivität L1 und Shuntwiderstand R8. Die Kondensatoren C1, C2, C3 und C5…C8 sind für eine adäquate Glättung der Ausgangsspannung des MIC3289 verantwortlich. Die sechs LED-Ketten bestehen aus jeweils sechs LEDs und sind einfach parallel geschaltet.

Mikrocontroller U2 erzeugt den Steuer-Bitstrom für den MIC3289. An den digitalen Eingängen sind vier Taster samt Pull-up-Widerständen angeschlossen. Das Entprellen der Tasten findet in der Software statt. Die Taster ermöglichen eine Einstellung der Helligkeit (UP/DOWN), das Ein- und Ausschalten (PWR) sowie das Speichern des Helligkeitswerts im EEPROM (MEM). Die ganze Schaltung wird mit 5 V/1 A (Netzteil oder Batteriepack) über Buchse K1 versorgt.

Auf das Timing kommt es an

Das Kommunikationsprotokoll ist bemerkenswert und verwendet nur eine Verbindung. Immer, wenn der MIC3289 einen Pegelwechsel am DC-Pin wahrnimmt, startet ein Timer, der 140 µs abwartet. Taucht kein weiterer Pegelwechsel auf, so schaltet der MIC3289 den Regler ein und liefert die volle Leistung ab, die der Shuntwiderstand erlaubt.

Wird der Pin für mehr als 1260 µs auf Low gehalten, schaltet der Ausgang ab. So ist es möglich, den MIC3289 als einfachen Ein/Aus-Schalter zu verwenden: Einfach den Eingang auf High legen und der Wandler startet nach 140 µs, genau so einfach auf Low legen und der Wandler schaltet nach 1260 µs ab.

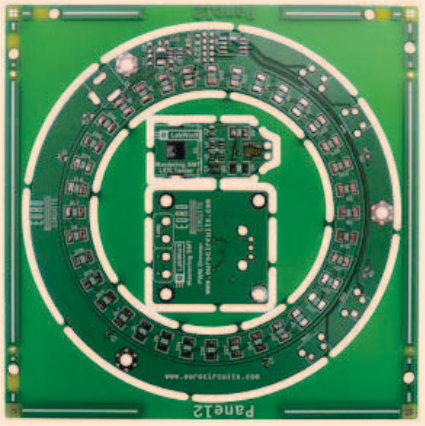
Wechselt aber der Pegel am DC-Pin innerhalb der ersten 50 µs des 140 µs-Fensters wieder auf Low, geht der MIC3289 in den Programmier-Modus. Ein anderer interner Timer startet. Sieht der MIC3289 nun ein Low-Signal mit einer Länge von 100…160 µs, geht er in einen Modus, in dem die Helligkeit erhöht wird, ist das Low-Signal dagegen 420…500 µs lang, wird die Helligkeit anschließend vermindert. Jeder Impuls zwischen 1 µs und 32 µs, der darauf folgt, erhöht beziehungsweise vermindert nun die Helligkeit um eine Stufe. Die programmierte Richtung derÄnderung und Helligkeitsstufe bleibt so lange erhalten, wie der Ausgang aktiv ist.

Beim Abschalten des MIC3289 wird die Helligkeit auf Maximum und die Zählrichtung auf „Vermindern“ (decrease) zurückgesetzt. Dies erlaubt es, während des Einschaltens einen Preset-Wert zu programmieren. Einfach den DC-Pin auf High legen und innerhalb der ersten 50 µs logische Nullen mit einer Dauer von höchstens 32 µs schreiben, und zwar so viele Impulse, wie man „Abwärts-

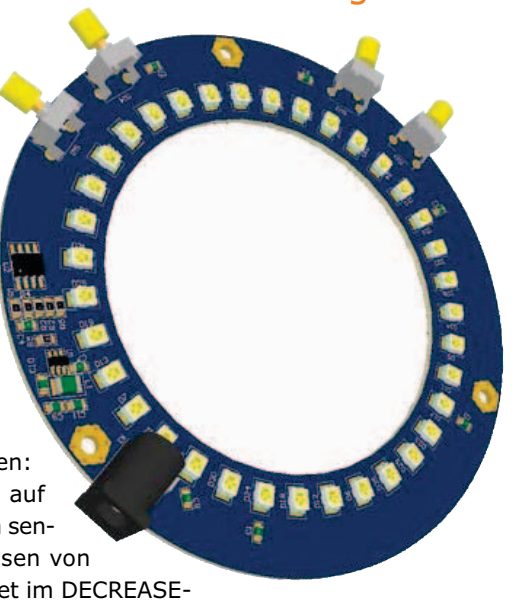
Bonus-Project: LED-Tester

Im Bausatz der LED-Ringleuchte ist ein kleines nützliches Gadget enthalten, mit dem man die Polarität von SMD-LEDs überprüfen kann. Diese Schaltung ist eine schöne kleine Übung zur SMD-Montage. Bringen Sie die Lötpaste mit der Schablone auf, richten Sie die SMDs korrekt aus und löten Sie diese auf der Platine fest. Eine Stückliste gibt es unter [1].

Bonus-Project: PWM-Dimmer für Glühbirnen



Das zweite Bonus-Projekt ist ein pulsweiten-modulierter Dimmer für ohmsche oder induktive Lasten wie Glühbirnen und Motoren und baut auf den Erfahrungen auf, die man beim beim Aufbau des LED-Testers gesammelt hat. Der Dimmer hat eine Eingangsspannung von 5…24 V und besitzt eine Softstart-Funktion. Mit einem Poti bestimmt man das Tastverhältnis. Eine Stückliste gibt es unter [1].



Bis 460 Lumen mit einem Abstrahlwinkel von 120 Grad

Schritte“ programmieren möchte. Dann lässt man den Pin auf High und der Wandler beginnt 140 µs nach dem letzten Impuls seine Arbeit.

Mit anderen Worten (siehe **Bild 4**):
Um den Ausgang einzuschalten:

- DC-Pin für mehr als 140 µs auf High legen.

Wenn der Ausgang schon eingeschaltet ist:

- DC-Pin für 100…160 µs auf Low, um den Einstellungs-Modus auf ERHÖHEN zu setzen.
- DC-Pin für 420…500 µs auf Low, um den Einstellungs-Modus auf VERRINGERN zu setzen.
- DC-Pin für 1…32 µs auf Low, um die Helligkeit um eine Stufe zu verändern.

DC-Pin für mehr als 1260 µs auf Low, um den Ausgang abzuschalten.

Für ein Preset beim Einschalten: Innerhalb 50 µs nach DC-Pin auf High eine Anzahl von Impulsen senden (Länge 1…32 µs mit Pausen von 1…32 µs). Der MIC3289 startet im DECREASE-Modus und bei maximaler Einstellung.

Beispiele

Beispiel Nr. 1

Nach dem Einschalten ist halbe Helligkeit erwünscht. DC wird für 40 µs auf High gelegt,

anschließend erfolgen acht Low-Impulse von 16 µs mit Pausen von ebenfalls 16 µs. Danach bleibt DC auf High, der Treiber startet 140 µs später. Die LEDs leuchten mit halber Kraft. 16 µs wurden gewählt, weil dies in der Mitte der erlaubten Spanne von 1…32 µs liegt. Da der voreingestellte Wert das Maximum ist, bewirken acht Pulse im DECREASE-Modus eine Reduzierung der Helligkeit um acht Stufen, die Hälfte der 16 verfügbaren Stufen.

Beispiel Nr. 2

Im Betrieb soll die Helligkeit um eine Stufe erhöht werden. Dies wird durch einen Puls mit der Länge von 100…160 µs erreicht, der den MIC3289 in den INCREASE-Modus schaltet, gefolgt von einem 16 µs-Puls, der die Helligkeit um eine Stufe steigert.

Beispiel Nr. 3

Im Betrieb soll die Helligkeit um eine Stufe verringert werden. Dies wird durch einen Puls mit der Länge von 420…500 µs erreicht, der den MIC3289 in den DECREASE-Modus schaltet, gefolgt von einem 16 µs-Puls, der die Helligkeit um eine Stufe absenkt.

Mikrocontroller

Als steuernden Mikrocontroller haben wir einen kleinen PIC12-Mikrocontroller mit acht Pins ausgesucht. Der Code tastet die vier Tasten ab und bestimmt die Befehle, die der MIC3289 erhalten soll. Die Tasten UP und DOWN erzeugen entsprechende Kommandos, um die Helligkeit in der gewählten Richtung zu ändern. Der Mikrocontroller behält den aktuellen Modus (Heller/Dunkler) und die aktuelle Helligkeit.

Der interne Zähler des MIC3289 läuft über, das heißt, wenn man beim Maximalwert eine Stufe erhöht, springt der Zähler auf das Minimum und umgekehrt. Die Steuerung (in unserem Fall der Mikrocontroller) kann dies aber verhindern, wenn er die Helligkeitseinstellung weiß und dann das Aussenden eines Impulses blockiert, der ein Überlaufen zur Folge hat. Genau dies geschieht auch in der Firmware des PICs.

Mit der MEM-Taste speichert man die aktuell eingestellte Helligkeitsstufe im controller-internen EEPROM. Mit der PWR-Taste schaltet man die Leuchte aus und (mit dem im EEPROM gespeicherten Wert) wieder ein.

Platinendesign

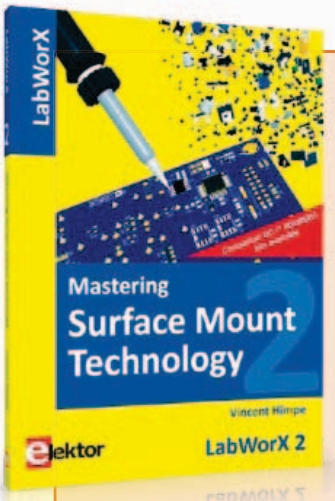
Ein reichhaltiges Baupaket ist für dieses Projekt verfügbar (siehe **Kasten**), angereichert durch ein paar kleine Extras. Die Platine ist natürlich ringförmig, so dass sie leicht an der Linse des Geräts (Kamera oder Mikroskop) angebracht werden kann. Die gesamte Elektronik befindet sich auf der Unterseite mit der möglichen Ausnahme der Drucktasten. An drei Montagelöchern kann das System am Objektiv befestigt werden, zum Beispiel mit Gummibändern, Kabelbindern oder einer selbst gebauten Feder-Spannvorrichtung.

An den gleichen Löchern kann man auch eine milchige Plexiglasscheibe als Diffusor befestigen, um eine gleichmäßigere Ausleuchtung zu erzielen.

Der Aufbau der Schaltung ist einfach: passive Bauteile wie Kondensatoren und Widerstände, danach die LEDs, die ICs und zum Schluss die Durchsteck-Taster und die Buchse, so lautet die richtige Reihenfolge beim Löten.

Und nun: Let your light shine bright… Zeigen Sie uns die interessanten und wohl-ausgeleuchteten Close-ups ihrer Schaltung!

(120701)



Mastering Surface Mount Devices

Möchten Sie mehr erfahren über das Projekt oder die Technik, wie man mit SMDs umzugehen hat, dann empfehlen wir die Lektüre des englischsprachigen Buches Mastering Surface Mount Technology aus der populären LabWorX-Serie von Elektor. Das Buch nimmt Sie mit auf einen Crash-Kurs durch die Welt der oberflächenmontierten Bauteile und gibt wertvolle Tipps und Know-how. Besuchen Sie www.elektor.com/labworx für weitere Informationen.

Bausatz verfügbar!

Alle Bauteile für die Ringleuchte bieten wir über unseren Business-Partner Eurocircuits an. Besuchen Sie www.elektor.com/ringlight, um einen Bausatz inklusive der beiden kostenlosen Bonus-Miniprojekte PWM-Dimmer und LED-Tester zu ordern. Im Bausatz befinden sich hochqualitative Platinen, alle Bauteile inklusive des programmierten Controllers und, einzigartig für Elektor, als Ergänzung eine Lötmaske, die das Auftragen der Lötpaste zum Kinderspiel macht.

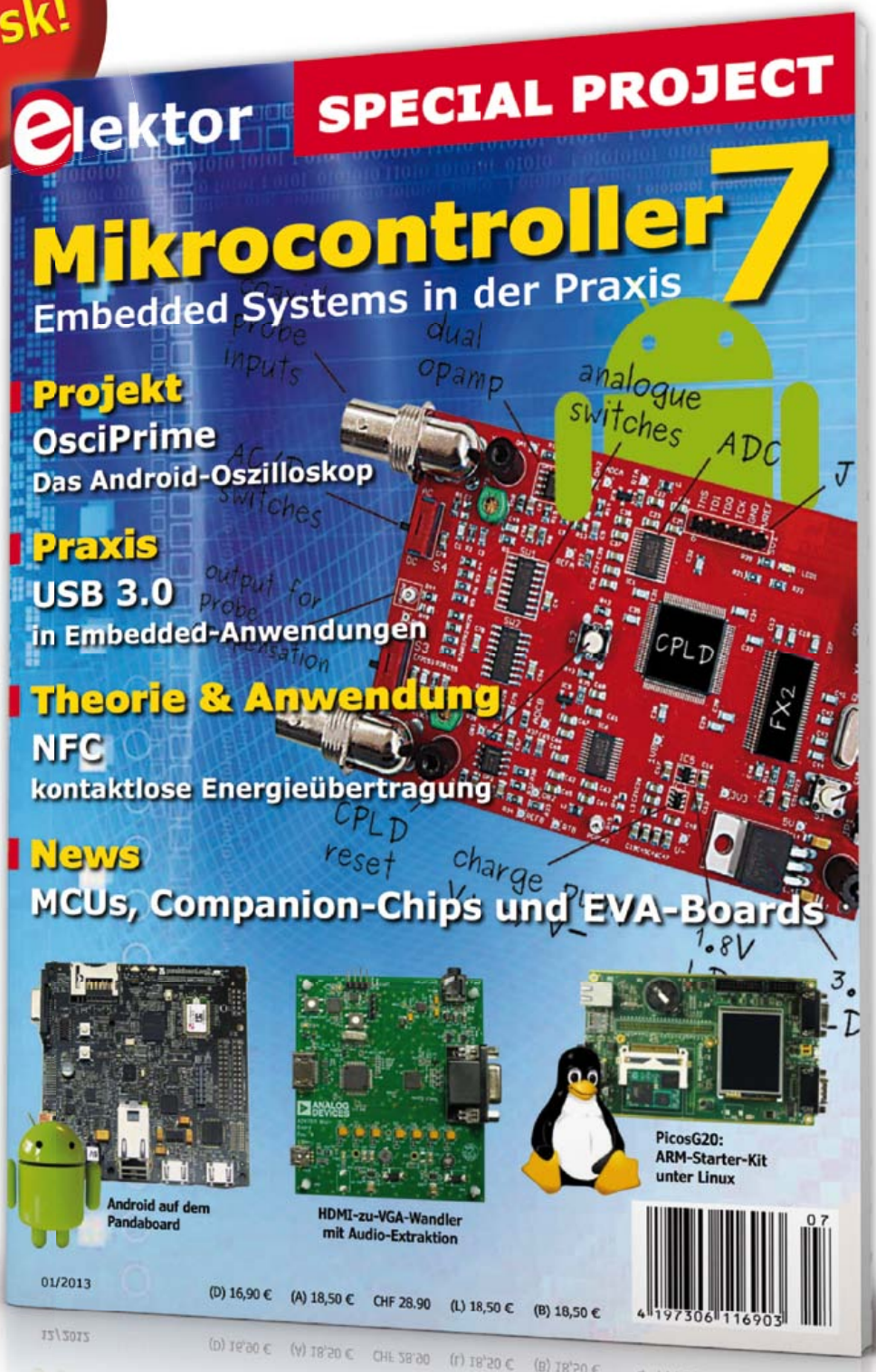
Weblinks

[1] www.elektor-magazine.de/120701

[2] www.elektor.com/labworx

Von den Machern von Elektor!

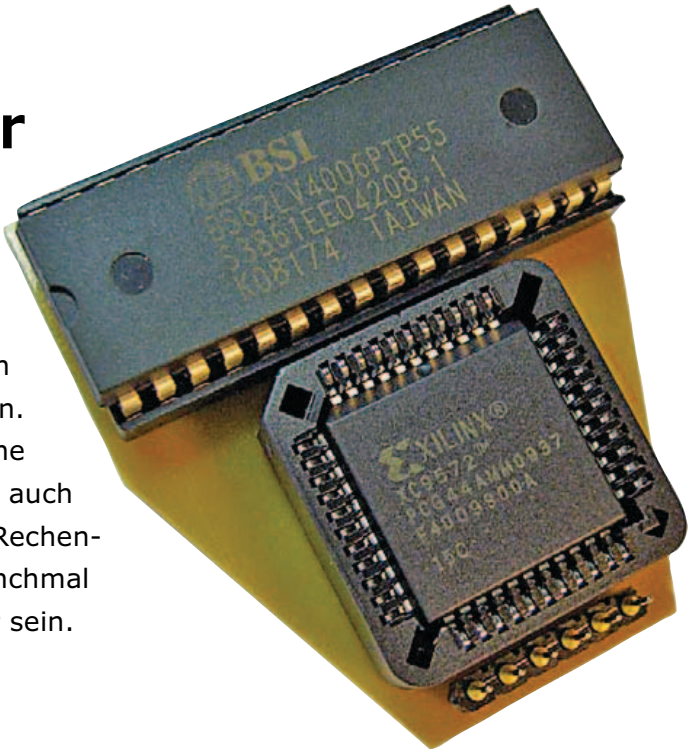
Jetzt neu am Kiosk!



Oder frei Haus unter
www.elektor.de/mikrocontroller7 bestellen!

RAMBO-S

SRAM-Controller mit SPI



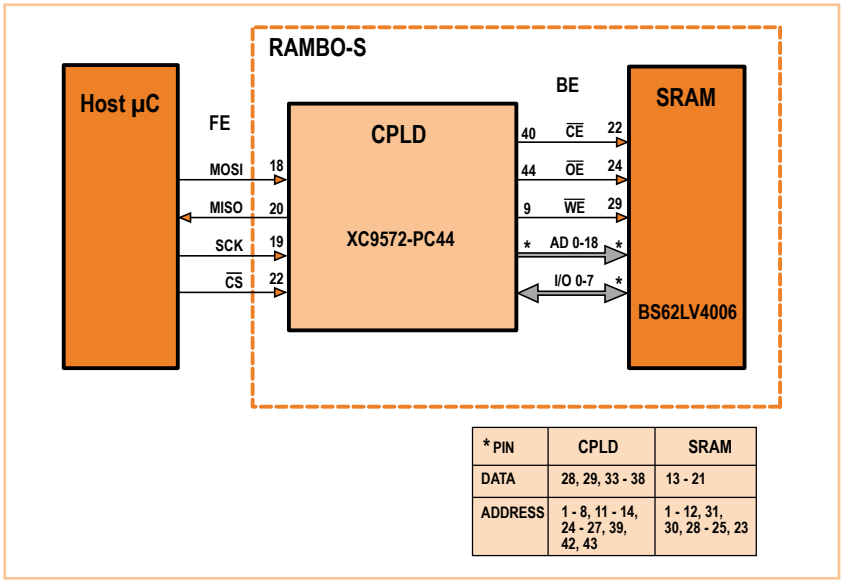
Von **Markus Hirsch** (D)

Mikrocontroller können immer mehr, und mit den Möglichkeiten wachsen auch die Anforderungen. Sinnvolle Anwendungen für kleine 8-bit-Controller gibt es dennoch auch heute noch jede Menge. Deren Rechenleistung reicht oft aus, aber manchmal dürfte es einfach mehr Speicher sein. Zeit für RAMBO-S!

Mehr Speicher bedeutet oft einen größeren (und komplexeren) Controller. Entweder hat man dann zu viel unnütze Leistung oder aber man verwendet doch den kleineren Controller und koppelt SRAM extern an.

Für kleinere Kapazitäten gibt es Chips mit seriellem Interface, was viele Leitungen für Daten und Adressen spart. Typisch ist z.B. das SRAM 23K256 mit 32 KB und SPI. Doch bei deutlich höherem Bedarf von 512 KB oder mehr sind lediglich SRAMs mit paralleler Adressierung erhältlich. Die dann notwendigen Pins für acht Datenbits und zusätzliche 19 oder mehr Adressleitungen überfordern so manchen kleinen Controller.

Eine Lösung wäre, auch viel SRAM per schnellem SPI anzuschließen. Man braucht also einen speziellen SPI-Controller, der eine Parallel/Seriell-Wandlung samt Speicheradressierung übernimmt. Genau dafür wurde RAMBO-S entwickelt. Es handelt sich dabei um ein Xilinx CPLD XC9572 im PC44-Gehäuse (44 Pins, davon 34 frei verfügbar), womit sich immerhin 512 KB an Speicher in Form von Standard-SRAM-ICs wie dem BS62LV4006 ansteuern lassen. Für noch mehr SRAM bräuchte man ein CPLD mit noch mehr Pins.



* PIN	CPLD	SRAM
DATA	28, 29, 33 - 38	13 - 21
ADDRESS	1 - 8, 11 - 14, 24 - 27, 39, 42, 43	1 - 12, 31, 30, 28 - 25, 23

Bild 1. Manchmal zeigt ein Blockschaltbild schon alles.

Hardware

Die komplette Elektronik besteht lediglich aus zwei Komponenten: CPLD und SRAM (siehe **Bild 1**). Die Adressleitungen kommen direkt an das CPLD; ein Zwischenspeicher ist nicht nötig. Es sind lediglich einige Kondensatoren zur Entkopplung vorgesehen. Das CPLD hat ein Front-End mit SPI und ein Back-End zum Anschluss von SRAM.

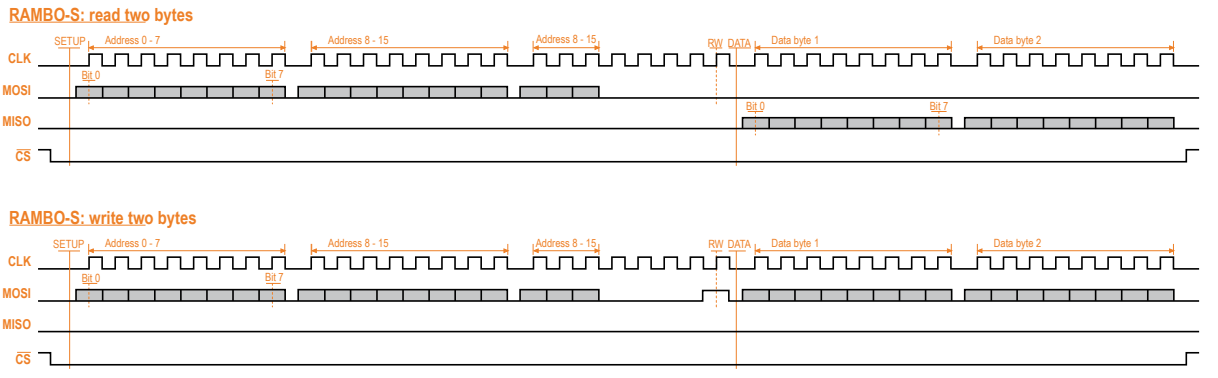


Bild 2.
Impulsdiagramme fürs Lesen und Schreiben mit RAMBO-S.

Interface

Die CPLD-Firmware ist so gestrickt, dass keine weiteren Bauteile notwendig sind. Die SPI-Taktleitung treibt auch die interne Logik. Nach dem Eintakten von zwei Adress-Bytes und einem Setup-Byte (in dem noch drei weitere Adress-Bits und R/W-Select stecken) können Daten Byte für Byte ein- oder ausgelesen werden. Die Adressen werden für sequentielles Lesen oder Schreiben automatisch inkrementiert. **Bild 2** zeigt die Impulsdiagramme für Lese- und Schreib-Operationen. Die Schaltung wurde mit Taktraten bis zu 2,2 MHz getestet. Hieraus folgt eine theoretische Datenrate von 275 KB/s. Das reale Maximum könnte durchaus noch höher liegen.

Firmware

Die vom Autor entwickelte Firmware ist wie immer über die Elektor-Webseite zu diesem Artikel [1] erhältlich. Mit diesen Dateien kann man sich sein eigenes CPLD für dieses Projekt brennen. Sie helfen auch beim Verständnis dessen, was in diesem Chip so vorgeht. Nur die private Nutzung ist frei, der Autor behält sich die kommerzielle Nutzung vor.

Entsprechend den gekennzeichneten Blöcken in **Bild 3** gliedert sich die Firmware in (1) SPI und Eingangs-Schieberegister, (2) Treiber für Eingang/Ausgang, (3) Ausgangs-Schieberegister, (4) Adresszähler, (5) Zähler-Steuerlogik und (6) SRAM-Steuerlogik.
Wenn $\overline{CS}$ „high“ ist, erfährt die interne Logik einen Reset und der Chip ist im Ruhezustand. Sobald aber $\overline{CS}$ „low“ wird, sind die Adress- und Setup-Bytes erforderlich. Die interne Logik lädt die ersten 19 bits in die Adress-Zähler und das letzte Bit des dritten Bytes steuert die Schreib/Lese-Logik.

Anschließend versetzt sich der Chip in den Streaming-Mode. Für jedes weitere Byte werden die Adress-Zähler einfach inkrementiert, womit die nachfolgenden Daten schnell via SPI gelesen oder geschrieben werden können.

Diese Prozedere funktioniert für eine (theoretisch) unbegrenzte Zahl an Bytes und endet erst, sobald $\overline{CS}$ wieder „high“ wird.

Um mehr als ein SRAM anzusprechen, können die restlichen freien Bits im Setup-Byte von einem Decoder zur Chip-Auswahl genutzt werden. Für jedes IC braucht es dann eine eigene CE-Leitung.

(091090)

Weblink

[1] www.elektor-magazine.de/091090

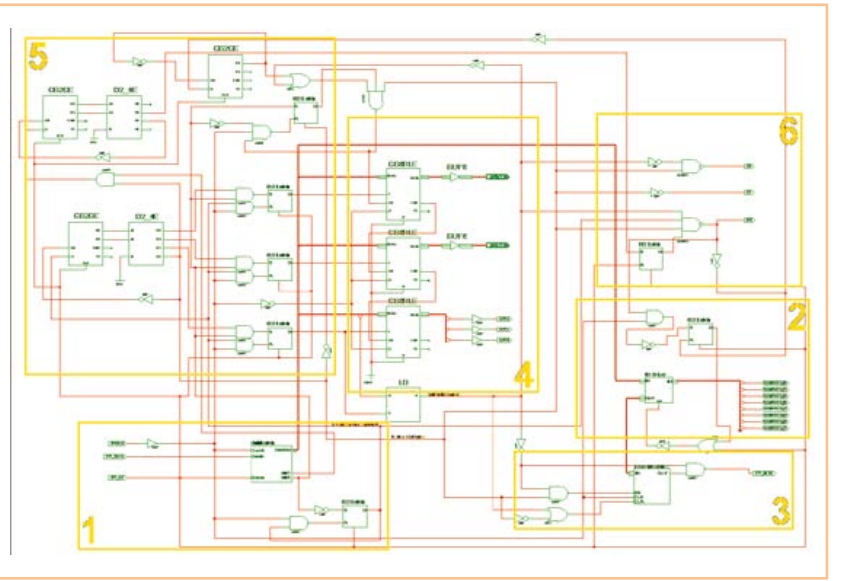


Bild 3.
Details des in RAMBO-S verwendeten CPLDs.

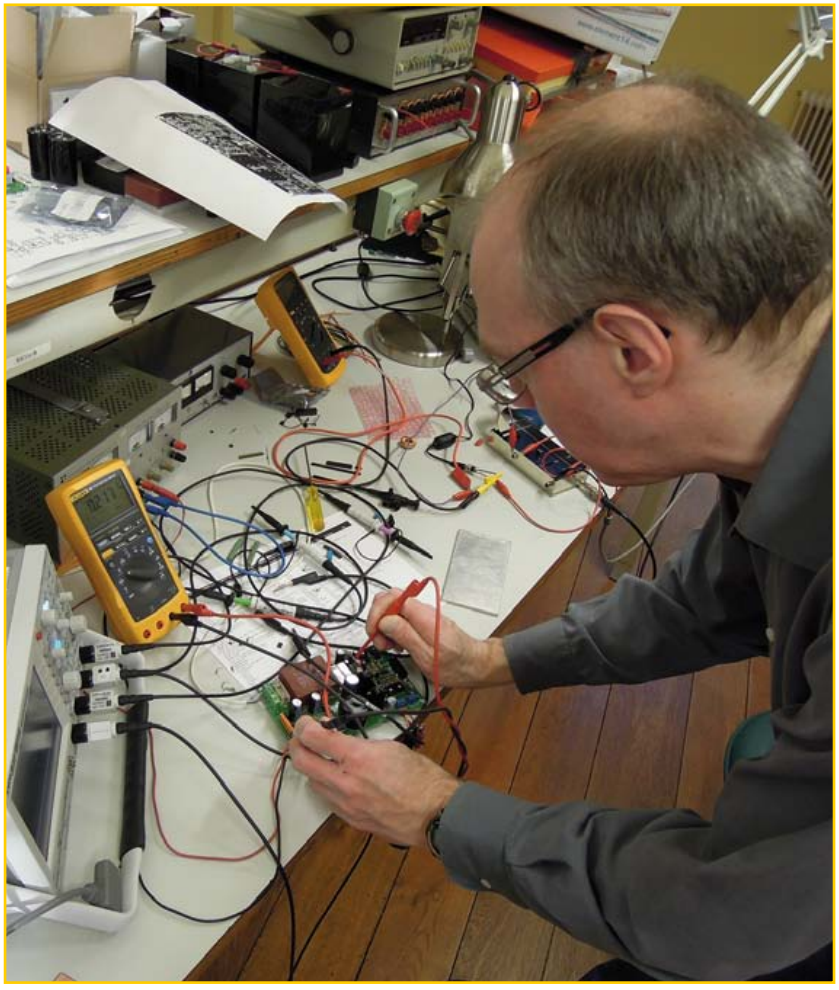
Wichtige Eilmeldungen!

Von **Clemens Valens**
(Elektor .Labs)

Elektor dot Labs ist das Herz von Elektor, wo alle Elektronik ihren Anfang nimmt. Projekte und Ideen werden hier gepostet, Schaltungen entwickelt, korrigiert, ausprobiert und in unserem Labor getestet. Wir haben hier wieder eine kleine Auswahl an interessanten Projekten zusammengestellt, die man auf dieser Seite finden kann!

Höher hinaus mit Volt und Ampere

Eines der ersten Projekte, das auf **.**LABS gepostet wurde, war eigentlich gar kein Projekt, sondern der Wunsch nach einem Labornetzteil. Keine stinknormale „30 volts max. lab PSU“, sondern eines mit bis zu 40 V. Offensichtlich war der „Original Poster“ (OP) nicht in der Lage, ein solches Netzgerät zu entwerfen und so kam er (oder sie) zu uns. Und weil Ihr Wunsch uns Befehl ist, haben wir uns der Sache angenommen und einen unserer Entwickler darauf angesetzt.



Nun wollte ich bemerken, dass das Projekt ein wenig nachlässig behandelt wurde und noch weit von seiner Fertigstellung entfernt ist, aber letzte Woche zeigte mir Ton Giesberts einen Prototyp und siehe da, er funktionierte! Nicht perfekt, mit ein wenig Oszillationen hier und Netzbrummen dort, aber es ging. Ton hat sich für ein unkonventionelles Konzept entschieden und statt eines variablen Spannungsreglers einen mit fester Ausgangsspannung eingesetzt. Dann hat er diesen überredet, wie ein variabler zu funktionieren. Oder so. Besser Sie fragen Ton, wenn Sie genau wissen wollen, wie sich die Sache abspielt… Übrigens: Solche Detailfragen werden auf **.**LABS auf der Seite zu diesem Projekt behandelt, also sehen Sie dort einmal nach, wenn Sie Netzgeräte interessant finden.
Einer der Gründe, warum das Projekt so langsam vorankam, war ein Bauteil, das auf der anderen Seite der Welt geordert werden musste. Ein Kondensator mit extrem niedrigem ESR chinesischer Provenienz. Die Chinesen haben wohl alle Exemplare selber gebraucht und dann doch noch einen entdeckt, der vom Labortisch gefallen und unter den Aktenschrank gerollt ist. Es hat ewig gedauert, ihn in einen Umschlag zu stecken und ihn Ton zu schicken.
Der OP hat 2 Ampere für das Labornetzgerät spezifiziert, für Ton nicht genug. 3 Ampere müssen es sein, oder 6 Ampere. So glaube ich, Sie müssen sich doch noch ein wenig in Geduld üben, bis der Entwurf fertig ist und in Elektor publiziert werden kann.

Sehen Sie Ton beim Entwickeln zu:
www.elektor-labs.com/120437

Höher hinaus mit meinem schönen Ballon

Wir alle wissen, dass unsere Welt drei Dimensionen hat. Zählen Sie die Zeit dazu, dann sind es vier. Aber wie mag wohl die fünfte Dimension aussehen? Mathematiker und Physiker arbeiten an diesem Thema seit Jahren und sind mittlerweile, wenn ich nicht irre, bei der 26. Dimension. Für uns Normalsterbliche ist das eine Spur zu abstrakt. Aber doch, seit 1967 steht die fünfte Dimension doch klar vor unseren Augen oder besser, vor unseren Ohren! Es war 1967, als die US-amerikanische Popband *The 5th Dimension* ihren Song *Up, Up and Away* vorstellte und damit fünf Grammies einheimste. Und wer hat den Song geschrieben? In der Tat, Jimmy Webb. Muss ich mehr sagen? Die fünfte Dimension ist schlichtweg das *world-wide jimmy web*! Nicht überzeugt? Ok, dann gehen Sie auf die Elektor dot LABS Website und was sehen Sie? Eine Anregung von Leser *ale* zu einer Wetterballon-Steuerung. Ballon, Web, fünfte Dimension, na, klingelt es?

Werfen Sie einen Blick auf *ale*s Wetter- oder, wie er es nennt, Sounding Balloon Main Unit Controller, ein sehr interessantes Projekt. Er schlägt ein System vor, das sechs analoge Signale plus Positionsdaten über eine FM-Funkverbindung übertragen kann. Damit lassen sich Parameter wie Temperatur, Infrarot- und Ultravioletteinstrahlung, Feuchte, Batteriepegel (obwohl, darüber sollte man sich keine Gedanken machen) und Luftdruck zur Erde übertragen. Und weil die Elektronik up, up and away in a beautiful balloon ist, sollte sie auch leichtgewichtig und billig sein.

Wo wir schon gerade beim Ballooning sind, wussten Sie schon, dass Elektor mit *Catch the Sun* das allererste Buch mit Nahfeldkommunikation (NFC) veröffentlicht hat? Es dreht sich ums Ballonfahren und enthält Links up and away ins Internet. Das weltweit erste Buch mit fünf Dimensionen!

Treten Sie ein in die fünfte Dimension:

www.elektor-projects.com/9121102594

Editor's Choice

Eine Reihe von .LABS-Projekten wurde von der Redaktion auserkoren, in nächster Zeit publiziert zu werden. Bei einigen Projekten haben wir aber feststellen müssen, dass die Original Poster (OP) nicht auf unsere Mitteilungen reagieren. Deshalb: Wenn Sie ein Projekt posten, achten Sie darauf, dass Ihre E-Mail-Adresse, die Sie zum Einloggen benötigen, auch richtig und aktuell ist. Wir können keine Projekte ohne Ihre ausdrückliche Zustimmung veröffentlichen, mögen sie auch noch so interessant sein. Hier eine Auswahl interessanter Projekte, die wie in der Zeitschrift Elektor publizieren möchten:

EDITOR'S CHOICE

Laser-TV

Das Projekt Laser-TV von OP hpt möchte ein Videobild mit Hilfe von 30 rotierenden Spiegeln auf einem VHS-Kopfmotor an die Wand werfen. Anfängliche Versuche waren vielversprechend, die größte Herausforderung ist die Phasensynchronisation der Kopfplatte. Der OP sucht interessierte BASCOM-Programmierer, die eine Motor-PLL (oder eine ähnliche Softwarelösung) entwickeln.

www.elektor-projects.com/9120502218

Analog-Theremin

Das Theremin war eines der ersten elektronischen Musikinstrumente. Es wurde gespielt, indem man die Hände dicht bei zwei Stabantennen hin- und herbewegte. Es gibt viele Designs im Netz, die aber auf simplen digitalen Rechtecksignalen beruhen. OP D. Philips möchte ein analoges Theremin mit echten Sinuswellen und hochqualitativer Steuerung bauen. Wollen Sie helfen?

www.elektor-projects.com/9120502157

Wasserbett-Energiesparer

Schlafen Sie in einem Wasserbett? OP Schwabix tut es, findet aber, dass der Energiebedarf zu hoch ist. Seine Idee ist es, die Heizung des Wasserbetts tagsüber automatisch abzuschalten. Natürlich müsste die Heizung rechtzeitig wieder eingeschaltet werden, damit das Wasser eine angenehme Temperatur erreicht hat, wenn Schwabix sich zum Schlafen niederlegt. Der OP wünscht Assistenz und ergiebige Diskussionen.

www.elektor-projects.com/9121102686

Portable Radio/Media Player

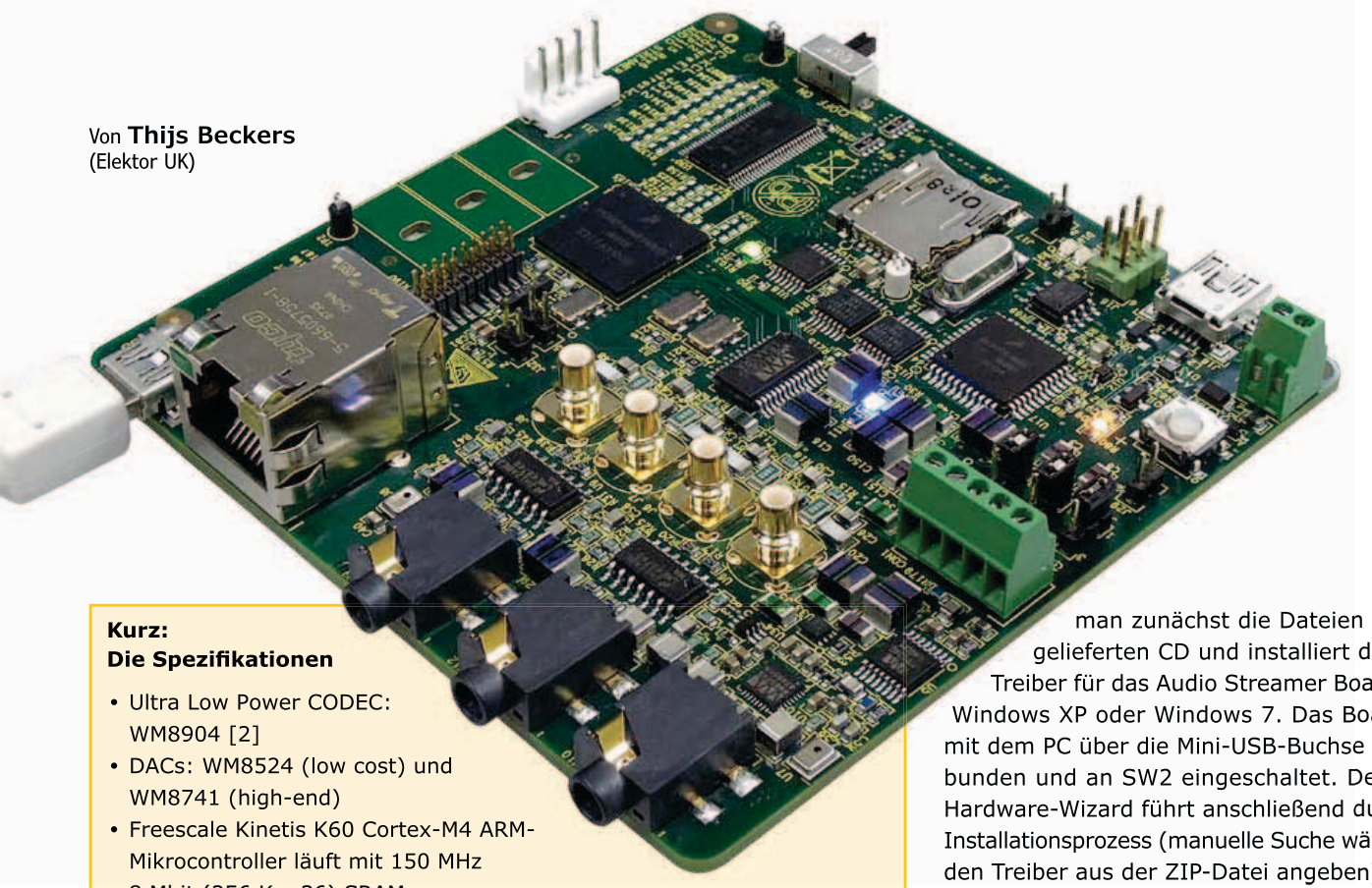
OP rsavas hat etwas entworfen, was er Portable Radio/Media Player nennt. Es ist ein Radio, ein MP3/AAC/WMA-Player für USB-Memorysticks oder SD-Memorycards und kann auch über den USB Codec (PC Stereo/sound card) an den PC angeschlossen werden. Es verfügt über Signalquellenauswahl, Lautstärkesteller, Kopfhörer- und einen 20-W-Klasse-D-Verstärker. Die Features sind nur von der Software limitiert, der OP sucht nach Programmierern, die ihm helfen, sein Projekt zu vollenden.

www.elektor-projects.com/9120502222

www.elektor-labs.com

Im Test: Audio Streamer

Von **Thijs Beckers**
(Elektor UK)



Kurz:
Die Spezifikationen

- Ultra Low Power CODEC: WM8904 [2]
- DACs: WM8524 (low cost) und WM8741 (high-end)
- Freescale Kinetis K60 Cortex-M4 ARM-Mikrocontroller läuft mit 150 MHz
- 8 Mbit (256 K x 36) SRAM
- JTAG on board
- SD-Karten-Slot
- 3 Touch-Sense-Tasten
- High-end-Kondensatoren (Nichicon Multilayer Polymer Film)
- 160-poliges Micro-Blox Interface
- Unterstützte Formate: WAV, MP3, FLAC [3] und Ogg Vorbis [4] (aus Lizenzgründen keine Apple-Formate).

Frisch auf den Tisch des Labors: eines der neuesten Entwicklungskits von Future Electronics, der Audio Streamer [1]. Dieses Mitglied der Micro-Blox-Familie ist ein „rapid proof-of-concept development system for OEMs“, das dem Entwicklungsingenieur auf schnelle und komfortable Weise die Implementierung neuer Applikationen mit der neuesten Generation digitaler Audio-DACs und Codecs von Wolfson Microelectronics [2] ermöglichen soll. Einen Überblick der wichtigsten Spezifikationen finden Sie im Kasten. Um das Board in Betrieb zu nehmen, entpackt man zunächst die Dateien der mitgelieferten CD und installiert den USB-Treiber für das Audio Streamer Board unter Windows XP oder Windows 7. Das Board wird mit dem PC über die Mini-USB-Buchse J20 verbunden und an SW2 eingeschaltet. Der Neue-Hardware-Wizard führt anschließend durch den Installationsprozess (manuelle Suche wählen und den Treiber aus der ZIP-Datei angeben).

Nach der Installation kann ein Audiosignal über den USB oder eine Netzwerkverbindung gestreamt werden. Eine umfassende Beschreibung, wie das Netzwerk einzurichten ist, findet sich im Quick Start Guide. Eine GUI mit dem Namen Audio-Demo erlaubt das Konfigurieren und das Herumspielen mit den Optionen des Boards.

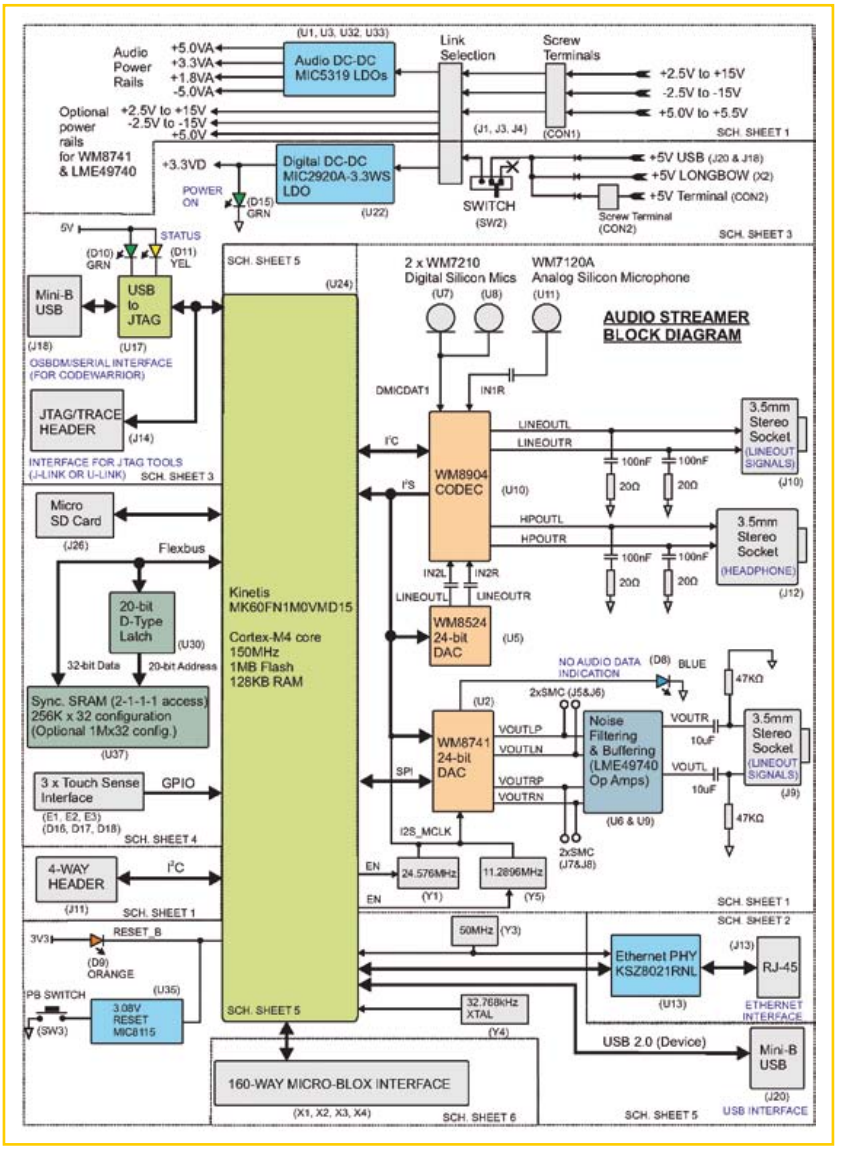
Nach Wahl der Verbindung ermöglicht ein Dropdown-Menü, das unter *Configure* erreichbar ist, die Wahl des DACs oder CODECs, der für die Wiedergabe des Audiostreams eingesetzt werden soll. Verschiedene Unter-Einstellungen sind erlaubt: Abhängig vom verwendeten DAC oder CODEC betreffen diese den Ausgang (LineOut oder Kopfhörer), einen Fünf-Band-Equalizer, diverse Filterkurven, den Anti-Clipping-Modus, das Stummschalten und die Quelle der Master-Clock (Kinetis K60 oder externer Oszillator).

Der Reiter *Play* zeigt eine Bibliothek mit vorhandenen Dateien; man kann hier auch eigene Testdateien vom PC oder einer SD-Karte im Slot hinzuladen. Zwei Schieberegler stellen Lautstärke

und Balance des Ausgangssignals ein.
Das Board ist ebenso in der Lage, im WAV-Format aufzunehmen und auf dem angeschlossenen PC zu speichern. Dazu gibt es zwei (macht Stereo) digitale „Silizium"-Mikrofone (Wolfson WM7210) und ein analoges. Beide Signalquellen speisen einen Low-cost-CODEC WM8904. Dies schafft eine gute Testplattform für alle Arten von Applikationen wie Security- oder Intercom-Kommunikation. In Sachen HiFi-Applikationen stellt der High-end-DAC WM8741 von Wolfson Microelectronics einen hervorragenden Weg zur Entwicklung von SD-, USB- und/oder Ethernet-Streaming-Applikationen dar. Der SMC-Verbinder J5-8 führt ein symmetrisches LineOut-Signal, das direkt vom DAC stammt und echte Messungen des Signal-Rauschverhältnisses zwischen DAC und den Opamps erlaubt (das S/N-Verhältnis kann durch Filterung an den Opamps verbessert werden). In der Audioschaltung kommen hochwertige Multilayer-Polymerfilm-Kondensatoren zum Einsatz. In Applikationen, die keine HiFi-Qualität erfordern, steht der WM8524 als Low-cost-Alternative zur Verfügung. Auch das komplexe Decodieren von FLAC- oder OggVorbis-Streams erfordert nur 30…40 % der Rechenkapazität des Kinetis-Chips, sodass noch ausreichend Reserven für parallel ablaufende Operationen vorhanden sind. Wenn man ausschließlich MP3 decodieren möchte, kann ein in puncto Rechenleistung kleinerer Kinetis-Mikrocontroller gewählt und auf das SRAM in der Endapplikation verzichtet werden.
IAR Embedded Workbench for ARM (Version 6.3 oder höher) ist die geeignete Entwicklungsumgebung für den Audio-Streamer. Der Quellcode der AudioDemo-Applikation ist in C# geschrieben und frei verfügbar bei Future Electronics. Das Board verwendet das MQX-Echtzeit-Betriebssystem, für das *Freescales MQX Software Solutions* alles zur Verfügung stellt, was des Entwicklers Herz begehrt. Es gibt noch eine Vielzahl von Features, die von der AudioDemo nicht genutzt werden, es gibt also noch viel zu entdecken.

Alles in allem stellt der AudioStreamer einen unkomplizierten Weg dar, dem Entwickler einen Streamer an die Hand zu geben. Mit der mitgelieferten Dokumentation und der (frei verfügbaren) Software kann eine vollständige Streaming-Applikation schnell und ohne einen eigens zu entwickelnden Prototyp gestartet und signifikant schneller abgeschlossen werden.

(120699)



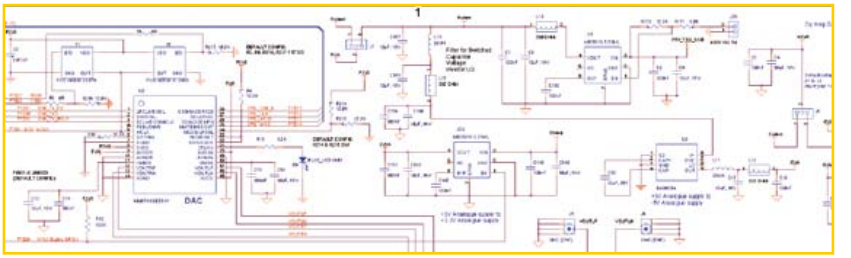
Weblinks

[1] www.my-boardclub.com/future_blox/miniblox.php?blox=32

[2] www.wolfsonmicro.com

[3] http://de.wikipedia.org/wiki/Free_Lossless_Audio_Codec

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



Ringkern mit Schnitt

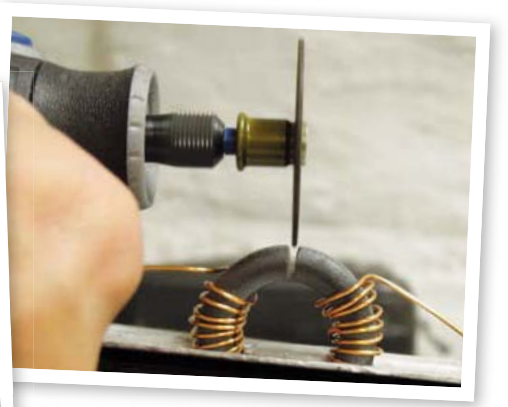
Von **Thijs Beckers**
(Elektor-Labor)

In dieser Ausgabe wird ein Einschaltstrombegrenzer beschrieben, der einen speziell angepassten Eisenringkern erfordert. Hier ist zu sehen, wie der Ringkern für den Einsatz in der Schaltung vorbereitet wird.

(120717)

Man nehme zuerst den (für die Applikation richtigen) Ringkern, winde die Wicklungen darum und spanne ihn vorsichtig (wirklich vorsichtig!) ein. Wir haben zum Schutz des Ringkerns weiches Alublech auf die Backen des Schraubstocks gelegt.

Mit einer Dremel oder einem ähnlichen Werkzeug, in das sich eine Trennscheibe für Metall einspannen lässt, beginnt man nun im rechten Winkel in Bezug auf den Ring zu schneiden. Tragen Sie einen Augenschutz!



Langsam arbeiten! Es braucht drei bis vier Minuten, um so weit zu kommen. Die Drehzahl sollte hoch sein, aber nicht die Nenndrehzahl der Scheibe überschreiten.

Spannen Sie den Ringkern nun erneut so ein, dass Sie bequem einen zweiten rechtwinkligen Schnitt durchführen können. Vorsicht: Der Ringkern verliert durch den Schnitt seine Festigkeit und das Material ist ziemlich spröde.

Nehmen Sie sich Zeit und drücken Sie nicht beim Weg durch den Kern. Die Trennscheibe ist zum Trennen und nicht zum Schleifen geeignet und bricht leicht, wenn sie rechtwinklig zum Radius belastet wird.

Geschafft! Wieder Vorsicht, der Ringkern kann vom Schneiden heiß sein. Nochmal, seien Sie sanft im Umgang mit dem Ringkern, er ist sehr spröde und bricht nun viel leichter als vor der Behandlung.

USB-Strom unlimited…
die Fortsetzung

Von
Raymond Vermeulen
(Elektor-Labor)

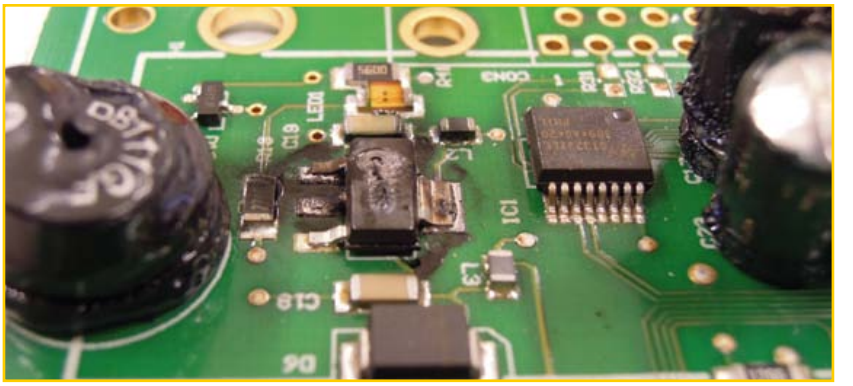
Kurze Erinnerung: Im September 2012 haben wir Sie ermuntert, ihre Kenntnisse über eine eventuelle Strombegrenzung des USB zu enthüllen. In der Zwischenzeit haben wir von Ihnen eine Vielzahl von Reaktionen erhalten und so langsam ergibt sich ein klares Bild. Es scheint aber, dass die Realität mehr Stolperdrähte bereithält als erwartet.
Auf der Host-Seite sind die zentralen Stromversorgungsschaltungen oft auf 500 mA pro Port limitiert. Zum Beispiel: Bei einem Motherboard mit zehn USB-Ports ist es nicht ungewöhnlich, dass ein einziger 5-A-Strombegrenzer alle Ports „schützt".
Viele Leser haben viele Geräte und Gerätchen getestet, inklusive Lasten mit einem Leerlaufstrom von 700 mA und mehr als 1 A im Betrieb. Es scheint Gründe zu geben, den Strom für passive Hubs zu begrenzen.

Einige interessante Beiträge:

„Das Missverhältnis zwischen dem erlaubten Strom von 500 mA (gewählt, weil dann noch als Spielzeug klassifiziert) und den anfänglich einkalkulierten 100 mA des USB liegt an den passiven Hubs. Dem Hub werden 100 mA zugeteilt, die restlichen 400 mA werden auf die Downstream-Ports verteilt, was erklärt, das sie vier Ports besitzen. Ein High-power-Gerät wird an einem passiven Hub versagen. Unglücklicherweise knausern die Hersteller immer bei Teilen, die sie als nicht essentiell empfinden. So findet man immer wieder Geräte ohne Strombegrenzung sowie Hubs, die nicht detektieren können, ob sie nun aktiv versorgt werden oder nicht." — Yann Vernier.

„USB ist nicht entworfen, um den Strom auf Geräteanfrage zu limitieren. Ein USB-Host sollte einen Treiber haben, der sich der gesamten Strom-Ressourcen aller Ports bewusst ist. Ein sehr wichtiger Aspekt dabei ist, dass physikalisch nichts diese 100 mA-Grenze erzwingt - sicherlich nicht die USB-Host-Hardware. ... Werfen Sie einmal einen Blick auf http://goo.gl/9vjD7 !"
— Ian G3ZHX

„Wir haben ähnlich wie Sie einen Stromshunt gebaut, den man von 10 mA bis etwa 2 A einstellen kann, ihn dann parallel zu einem Low-Power-Gerät geschaltet (eine USB-Maus) und den von der USB-Schnittstelle gelieferten Strom gemessen. Ergebnis: Ohne Nachfrage werden 500 mA geliefert; wir haben dann den Abnahmestrom weiter erhöht auf etwa 1,4 A. Erst ab diesem Wert hat die USB-Hardware wirklich abgeschaltet. Der User der USB-Schnittstelle muss also selber dafür sorgen, dass max. 500 mA fließen. Ein Kurzschluss wird auch erkannt, da wird auch erst ab etwa 1,4 A wirklich ausgelöst.



Weiterhin haben wir verschiedene USB-Devices getestet. Beispielsweise „verbrauchen" die USB-Scanner der Lide-Serie von Canon im Ruhezustand einen Strom von etwa 700 mA, der während des Scan-Vorgangs bis auf ca. 950 mA ansteigt. Also werksseitig eine Verletzung der USB-Vorgaben. Ebenfalls kein Schutz ist es, wenn man einen Hub (natürlich mit einem eigenen Netzteil, also self-powered) davorschaltet. In einem Versuch haben wir die Maus-Shunt-Schaltung an solch einen Hub gesteckt. Auch dabei wurde der Strom nicht auf 500 mA begrenzt. Und- noch viel schlimmer: Als wir das Netzteil dieses Hubs entfernt haben, holte sich der Hub den Betriebsstrom direkt vom Host (PC/Laptop). Eine Art Pufferung durch Verwendung von Hubs ist also nicht möglich gewesen."
— Gerald E. Riemer (Dipl.-Phys. Ing.).

Unsere Leser haben gesprochen.
Danke und gute Nacht!

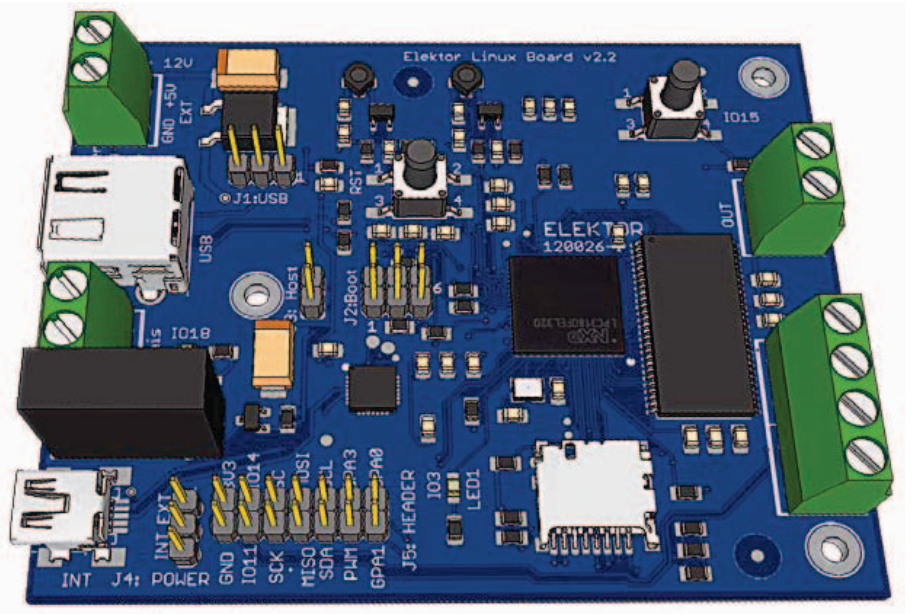
(120575)

Embedded Linux leicht gemacht (7)

I²C, UART, RS485

Von
Benedikt Sauter [1]

In der vorangegangenen Ausgabe haben wir unsere Leser dazu aufgerufen, den Inhalt des letzten Teiles dieser Artikelserie mitzubestimmen. Vielen Dank an die weit über 100 Leser, die an der Umfrage teilgenommen haben! Auf vielfachen Wunsch werden wir diesmal noch etwas genauer auf die Entwicklung eigener Anwendungen eingehen. Auch serielle Schnittstellen wie I²C und UART, die man in eigenen Projekten häufig benötigt, sind ein Thema.



3D-Modell des Elektor-Linux-Boards, erstellt mit dem Tool EagleUp [14].

Open-Source-Software lebt von der offenen Entwicklung in der Community. Eine Quelle für Updates, Patches oder Erweiterungen sind daher die Webseiten des jeweiligen Projektes, auf denen man wichtige Hinweise und meist auch ein User-Forum findet.

Basis des Elektor-Linux-Boards ist das Projekt Gnublin [2], das der Autor gemeinsam mit der Hochschule Augsburg 2009 als Plattform für die Embedded-Linux-Entwicklung gestartet hat. In der Zwischenzeit sind neben verschiedenen Board-Versionen, passenden Ergänzungs-Schaltungen (Schrittmotorsteuerung, CAN, …) auch diverse Erweiterungen der Software dazugekommen. Unter anderem existieren inzwischen Updates des Kernels und der Toolchain samt C/C++-Compiler. Diese wollen wir sogleich installieren!

Updates

Um Programme nicht immer direkt auf dem Elektor-Linux-Board übersetzen zu müssen, haben wir im dritten Teil der Serie [3] einen C/C++-Compiler auf dem PC installiert. Wir können die neueste Version dieses Compilers parallel mit der übrigen Toolchain installieren.

Am besten öffnet man dafür in unserem Ubuntu-Linux mit der Tastenkombination „Strg-Alt-t“ eine neue Konsole und gibt dort den folgenden Befehl ein:

```
wget http://gnublin.org/downloads/eldk-eglibc-i686-arm-toolchain-qte-5.2.1.tar.bz2
```

Ist das Archiv fertig heruntergeladen, dann kann man es einfach in das eigene Dateisystem entpacken:

```
sudo tar xjf eldk-eglibc-i686-arm-toolchain-qte-5.2.1.tar.bz2 -C /
```

Um den neuen Compiler jetzt nutzen zu können, muss man ein kleines Skript schreiben, das die Umgebungsvariablen zu den Verzeichnissen der neuen Installation automatisch setzt. Dafür legt man eine Datei „set.sh“ im Ordner „/opt/eldk-5.2.1“ an:

```
sudo gedit /opt/eldk-5.2.1/set.sh
```

In **Bild 1** sieht man, wie die Verzeichnisse zu der Umgebungsvariablen PATH hinzugefügt werden. Nach dem Speichern kann man die Datei auf der Konsole aufrufen (auf Syntax achten: Punkt Leerstelle Schrägstrich):

```
. /opt/eldk-5.2.1/set.sh
```

Dieses Kommando muss ab jetzt immer wieder aufgerufen werden, wenn man ein Programm auf dem Entwicklungs-PC für das Elektor-Linux-Board übersetzen möchte.
Der neue Compiler der Toolchain kann dann wie folgt aufgerufen werden:

```
arm-linux-gnueabi-gcc -v
```

C/C++-Entwicklungsumgebung

Viele Leser haben nach einer Möglichkeit gefragt, C/C++-Software für das Board mit einer komfortablen IDE wie zum Beispiel Eclipse entwickeln zu können. Wir beschreiben daher in diesem Abschnitt kurz, wie man auf eine einfache Art und Weise C/C++-Programme auf dem Entwicklungsrechner schreiben und direkt auf dem Board testen kann.

Was man dazu braucht ist:

- Konsole des Linux-Boards (Zugriff per UART oder Netzwerk)
- Netzwerk-Dateisystem (für den Zugriff vom Entwicklungs-PC aus)
- Optional natürlich die Entwicklungsumgebung

Da in einem Linux-Verzeichnisbaum Geräte abgebildet werden wie Dateien, ist es naheliegend, mit diesem Betriebssystem auch entfernte Speichermedien genauso in den Verzeichnisbaum einhängen zu können wie einen lokalen Ordner bzw. Unterordner. Hierfür gibt es unter Linux (natürlich) wieder viele verschiedene Möglichkeiten wie z.B. NFS, FTP oder Samba.

Im vorangegangenen Teil der Serie [4] wurde eine Möglichkeit zur Fernwartung des Linux-Boards per SSH vorgestellt. SSH ist ein Server-Programm, mit dem man Zugriff auf die Konsole des Boards per Netzwerk bekommt. SSH ist vergleichbar mit Telnet, nur bietet es eine erhöhte Sicherheit, da

```
P1=/opt/eldk-5.2.1/armv5te/sysroots/i686-eldk-linux/usr/bin/armv5te-linux-gnueabi/
P2=/opt/eldk-5.2.1/armv5te/sysroots/i686-eldk-linux/bin/armv5te-linux-gnueabi/
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
export PATH=$P1:$P2:$PATH
```

Bild 1.
Umgebungsvariablen für die neue Toolchain.

es die Verbindung vom Server bis zum Nutzer verschlüsselt. Außerdem lassen sich auf einfache Weise Dateien zwischen Nutzer und Server austauschen; auf Nutzerseite kann man dafür das Programm *scp* verwenden.

Hat das Linux-Board beispielsweise die IP-Adresse 192.168.0.190, so kann man vom Entwicklungs-PC eine Datei direkt in das Dateisystem des Linux-Boards übertragen:

```
scp test.c root@192.168.0.190:/root
```

Soll ein ganzer Ordner übertragen werden, dann schreibt man:

```
scp -r myproject root@192.168.0.190:/root
```

Als erstes gibt man die Datei oder den Ordner an. Im Falle des Ordners muss man ein zusätzliches -r angeben, damit alle weiteren Ordner und Dateien darunter rekursiv mit übertragen werden. Danach muss man noch den Benutzer-

namen angeben, gefolgt von einem @-Zeichen und der IP-Adresse. Nach dem Doppelpunkt gibt man schließlich den Zielpfad im Verzeichnisbaum des Linux-Boards an.
Jetzt könnte man auf diese Art und Weise in regelmäßigen Abständen die übersetzten Programme auf das Board übertragen. Aber es geht noch eleganter!
Ein Profi installiert sich unter Linux das Programm *sshfs*. Auf unserem Entwicklungs-PC kann man dies machen mit:

```
sudo apt-get install sshfs
```

Mit dem Tool können wir das komplette Dateisystem des Linux-Boards live in unserem Entwicklungs-PC freigeben. So müssen wir nie wieder Dateien von Hand auf die SD-Karte des Linux-Boards kopieren. Voraussetzung hierfür ist (wie bereits bei SSH oder scp) natürlich eine bestehende Netzwerkverbindung.
Auf dem Dateisystem des Entwicklungs-PCs muss man zuerst einen sogenannten „mount-Punkt" anlegen. An dieser Stelle kann man das entfernte Dateisystem einhängen. Der mount-Punkt ist ein gewöhnliches leeres Verzeichnis, das man mit dem Programm *mkdir* über die Konsole oder auch mit einem grafischen Dateiexplorer anlegen kann.

Zuerst wechseln wir in den Home-Ordner…

```
cd ~
```

… und legen den mount-Punkt an:

```
mkdir elektor-linux-board
```

Jetzt kann man mit dem nächsten Befehl das gesamte Dateisystem des Linux-Boards auf dem Entwicklungs-PC verfügbar machen:

```
sshfs root@192.168.0.190:/
elektor-linux-board
```

Nach dem Aufruf wird man wieder einmalig nach dem Passwort gefragt. Ist dies erfolgreich eingegeben, kann man mit `cd` und `ls` im Dateisystem herumnavigieren. Testet man ein paar Verzeichnisse durch, wird man ab und zu aber ein „Permisson denied" gemeldet bekommen. Denn beim Elektor-Linux-Board gehören die Systemprogramme natürlich dem Benutzer *Root*, doch man arbeitet aktuell mit der Benutzer-Identität des Entwicklungs-PCs. Im Wesentlichen wird ein Benutzer durch seine User-ID bestimmt. Um sich komplett frei in dem eingehängten Dateisystem bewegen zu können, muss man sich als *Root* anmelden. Am einfachsten startet man dafür eine Konsole, die mit Root-Rechten ausgeführt wird:

```
sudo /bin/bash
```

In dieser Konsole kann man sich ganz frei in dem eingehängten Dateisystem bewegen, Dateien einfügen und löschen.
Möchte man jetzt Programme für das Board auf dem Entwicklungs-PC schreiben, dann muss man den Projektordner einfach in dem eingehängten Dateisystem ablegen. Wie man Eclipse für die Entwicklung von C/C++ verwendet, ist im Wiki [5] beschrieben.
Idealerweise legt man sich vor dem Start eines neuen Projektes einen eigenen Benutzer auf dem Elektor-Linux-Board an. Dies macht man mit:

```
adduser elektor
```

Dann kann man auch nur das Home-Verzeichnis des Benutzers in das eigene Dateisystem einhängen:

```
sshfs elektor@192.168.0.190:/home/elektor
elektor-linux-board
```

Arbeitet man nur immer alleine am PC und mit dem Board, so erhalten beide Nutzer automatisch die User-ID 1000 und man hat in diesem Ordner nie Probleme mit den Rechten. Trotzdem kann man natürlich zwischendurch ein Programm auch als *Root* ausführen.

Kernel-Update

Der Kernel dient den Anwendungen als Schnittstelle zur Hardware. Möchte man zusätzliche oder neuere Hardware ansprechen, dann kann es immer wieder notwendig sein, sich eine neue Kernel-Version herunterzuladen und manuell zu übersetzen.
Für das Elektor-Linux-Board ist das Kernel-Archiv des Gnublin-Projektes die erste Anlaufstelle. Seit dem Beginn der Artikelreihe wurde der Kernel erweitert, so kann man beispielsweise jetzt auch CAN-, SPI- oder I2C-Geräte ansprechen.
Zentral wird der Quelltext in einer Versionsverwaltung gepflegt. Entwickler können hier Änderungen einfach und nachvollziehbar anderen Entwicklern

zur Verfügung stellen. Als Software zur Verwaltung des Softwarearchivs wird GIT verwendet. Um auf ein GIT-Repository zugreifen zu können, benötigt man das GIT-Softwarepaket. Hierzu gibt man am Entwicklungsrechner ein:

```
sudo apt-get install git-core
```

Anschließend kann man in einen beliebigen Ordner wechseln und dort das Softwarearchiv aus dem Repository „klonen“.

```
git clone http://code.google.com/p/
gnublin-develop-kernel/
```

Dies kann etwas dauern. Sobald das Herunterladen erfolgt ist, blinkt der Cursor wieder und ist bereit für neue Eingaben. Um den neuesten stabilen Kernel für das Board zu „bauen“, wechselt man in das Verzeichnis für den 2.6.33er Kernel (Experimentierfreudige können gerne mal die Version 3.3.0 des Kernels testen):

```
cd linux-2.6.33-lpc313x
```

In diesem Ordner findet man die passende Standardkonfiguration:

```
make elektor_defconfig
```

Möchte man manuelle Änderungen an der Konfiguration vornehmen, kann man dies wieder erledigen mit:

```
make menuconfig
```

Anschließend wird der Kernel übersetzt (Aufruf von „set.sh“ nicht vergessen):

```
make zImage
```

Ist der Kernel fehlerfrei erzeugt worden, kann dieser wieder mit einem SD-Kartenleser auf die SD-Karte übertragen werden (für „LABEL_SD_CARD“ den entsprechenden Ordner einsetzen):

```
sudo cp arch/arm/boot/zImage /media/
LABEL_SD_CARD/
```

Die Module übersetzen wir mit folgendem Befehl:

```
make modules
```

Danach installieren wir die Module in ein Zwischenverzeichnis:

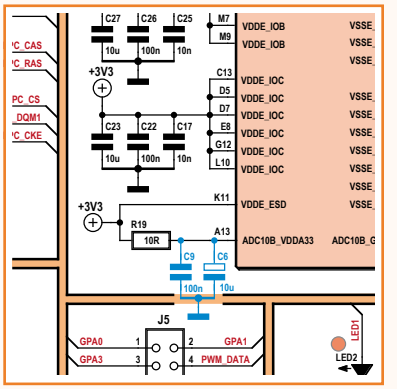
```
make modules_install INSTALL_MOD_PATH=/
tmp
```

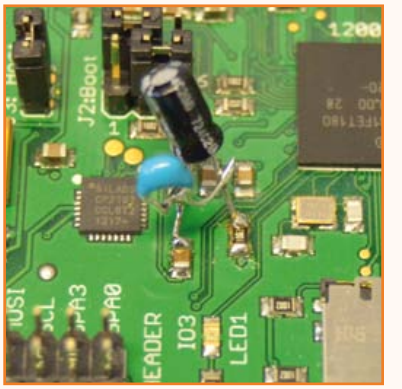
Vom Zwischenverzeichnis aus werden die fertigen Module schließlich auf die SD-Karte kopiert:

```
sudo cp -r /tmp/lib /media/LABEL_SD_CARD/
```

Genauere Spannungs-Messungen

Im fünften Teil der Serie [15] hatten wir gezeigt, wie man den Analog/Digital-Wandler des Controllers für Spannungsmessungen verwendet, und natürlich haben das viele Leser gleich ausprobiert. Dabei gab es vereinzelt Rückmeldungen, dass die 3,3-V-Board-Spannung, die vom ADC als Referenz genutzt wird, nicht stabil genug für präzise Messungen ist. Doch es gibt hierfür eine einfache Lösung: Um zu einer stabileren Spannung zu kommen, muss man bei R19 einfach zwei Kondensatoren mit ca. 10 µF und 100 nF nach Masse schalten (**Bild 18** und **Bild 19**). Schon sollten die Messungen wesentlich genauer sein.





I2C-Tools

I2C ist seit Jahrzehnten ein sehr beliebter Bus im Mikrocontroller-Bereich. Es gibt viele verschiedene Bausteine, vom A/D-Wandler über den I/O-Expander bis hin zum Temperatursensor reicht die Palette. Um I2C unter Linux verwenden zu können, müssen die entsprechenden Treiber im Kernel aktiviert sein.

```
Device Drivers → I2C support → I2C
Hardware Bus support → I2C bus support
for Philips PNX targets
```

In der Standardkonfiguration ist der Treiber bereits dabei.

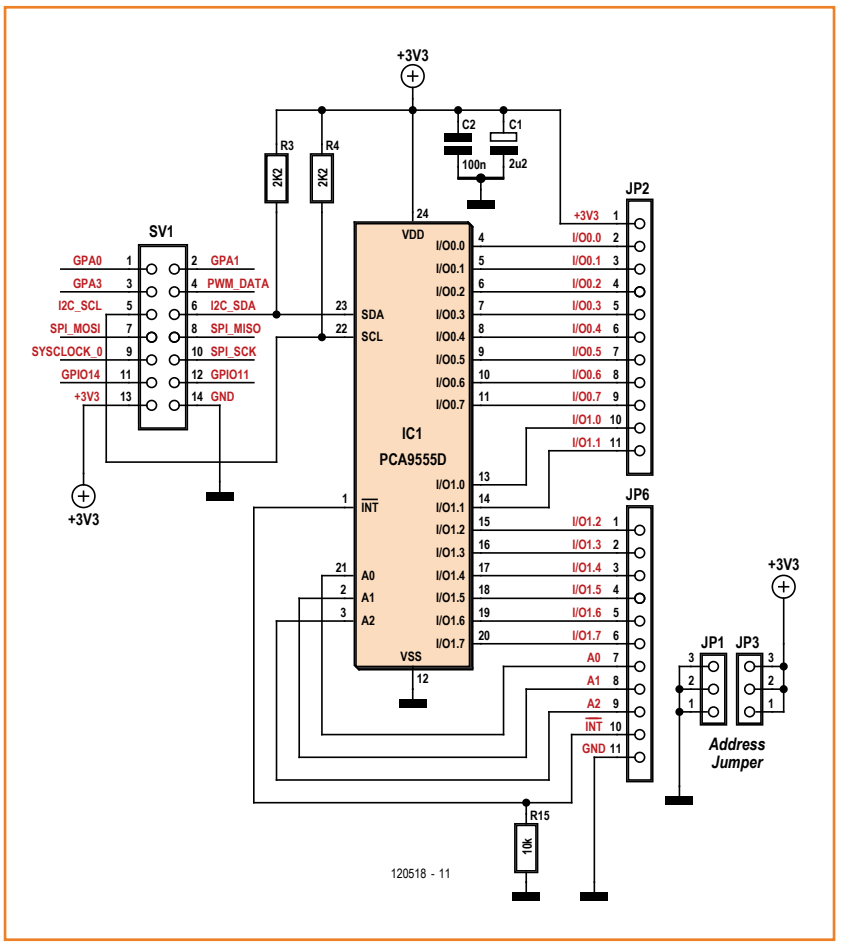


Bild 2.
Schaltung mit dem I2C-Portexpander PCA9555.

Für einfache Tests bietet Linux I2C-Programme an. Diese müssen aus dem Internet heruntergeladen, kurz umverpackt und anschließend auf die SD-Karte geschrieben werden.
Zuerst laden wir die I2C-Tools auf den Entwicklungs-PC herunter:

```
wget http://ftp.de.debian.org/debian/
pool/main/i/i2c-tools/i2c-tools_3.0.2-5_
armel.deb
```

Dann wandeln wir das Debian-Paket in ein Standard-Archiv um:

```
alien -t i2c-tools_3.0.2-5_armel.deb
```

Jetzt findet man im gleichen Ordner eine Datei mit dem Namen „i2c-tools-3.0.2.tgz“. Diese lässt sich per SD-Kartenleser auf die SD-Karte kopieren:

```
cp i2c-tools-3.0.2.tgz /media/
LABEL_SD_CARD/
```

Nach dem Neustart des Linux-Boards muss man einmalig das eben umgewandelte Paket entpacken und zusätzlich einmalig die Gerätedateien für I2C anlegen.
Zuerst ist also das Entpacken der I2C-Linux-Tools an der Reihe:

```
cd /

tar xvzf i2c-tools-3.0.2.tgz
```

Die Gerätedateien legen wir wieder mit dem Befehl `mknod` an [6]. Sicherheitshalber kann man die Major-Nummer ermitteln mit:

```
cat /proc/devices
```

Beim entsprechenden Eintrag sollte eine „89“ stehen. Steht hier eine andere Nummer, müssen die nächsten Befehle entsprechend angepasst werden.

```
mknod /dev/i2c-0 c 89 0
mknod /dev/i2c-1 c 89 1
```

Sind die Gerätedateien angelegt und ist die Software installiert, kann das erste I2C-Gerät angeschlossen und angesprochen werden. Für einfache Tests bietet sich der Baustein PCA9555 an, ein Portexpander, den wir wie in **Bild 2** an den 14-poligen Erweiterungsstecker des Boards anschließen.
Für unsere Tests benötigen wir eine Stromversorgung von 3,3 V, die am Erweiterungsstecker abgegriffen werden kann. An den Eingängen A0 bis A2 des Portexpanders kann man direkt Jumper anbringen, um die Leitungen entweder auf GND oder 3,3 V legen zu können. Hiermit stellt man die Adresse des Bausteins ein.
Ist der Chip entsprechend verschaltet und über SDA und SCL an das Elektor-Linux-Board angeschlossen, dann kann mit dem Tool „i2cdetect“ der komplette I2C-Adressraum am Bus gescannt werden:

```
i2cdetect 1
```

Als Ausgabe erhält man **Bild 3**. Die Adresse 0x6e stammt vom LPC3131-Interface. Da alle Adressleitungen des PCA9555 in unserer Schaltung auf High gelegt worden sind, haben wir für den Baustein die Adresse 0x27 gefunden.
Für einen einfachen Test haben wir drei LEDs an den

Port 0 des Bausteins gehängt (siehe **Bild 4**). Mit diesen LEDs kann man nun einen einfachen Funktionstest des I2C-Busses durchführen. Um kurze I2C-Nachrichten per Kommandozeile versenden zu können, verwendet man das Program „i2cset". Alle I/O-Pins als Ausgang definieren:

```
i2cset 1 0x27 0x06 0x00
```

Setzen der Ausgänge auf „High":

```
i2cset 1 0x27 0x02 0xff
```

Jetzt sollten die angeschlossenen LEDs leuchten. Nachdem wir jetzt schon einfache I2C-Tests direkt auf der Konsole durchführen können, wollen wir uns nun einem einfachen C-Programm für eine I2C-Steuerung widmen.

I2C in C

Der LPC3131 bietet zwei physikalisch getrennte I2C-Schnittstellen an, wobei am 14-poligen Stecker nur der zweite Bus verfügbar ist.
Die Gerätedateien heißen „/dev/i2c-0" bzw. „/dev/i2c-1". Der Zugriff auf die Hardware erfolgt wie immer; man öffnet die Gerätedatei und greift anschließend lesend oder schreibend auf diese zu. In **Listing 1** sieht man ein kleines C-Programm, mit dem wir wieder unsere LEDs leuchten lassen. Das Programm kann man auf dem Entwicklungs-PC übersetzen mit:

```
arm-linux-gnueabi-gcc -o i2ctest
i2ctest.c
```

Anschliessend nutzt man die SD-Karte (oder die Tools *scp* bzw. *sshfs*), um die übersetzte Datei auf das Board zu übertragen. Optional kann man die Datei auch direkt auf dem Elektor-Linux-Board mit dem integrierten C-Compiler *gcc* übersetzen. Idealerweise kopiert man das Testprogramm nach „/usr/bin", dann kann es von jedem Verzeichnis aus ohne explizite Pfadangabe aufgerufen werden:

```
cp i2ctest /usr/bin
```

Das Programm startet man auf dem Board, mit der Gerätedatei für den I2C-Bus als Parameter:

```
i2ctest /dev/i2c-1
```

Jetzt sieht man die angeschlossenen LEDs an- und ausgehen.

```
root@gnublin:~# i2cdetect 1
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-1.
I will probe address range 0x03-0x77.
Continue? [Y/n]
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- 6e --
70: -- -- -- -- -- -- -- --
root@gnublin:~#
```

Bild 3.
Ausgabe mit dem Programm „i2cdetect".

Im oberen Teil des Quellcodes (Listing 1) sieht man, wie mit `open()` ein File-Handle für die Gerätedatei angelegt wird. Mittels `ioctl()` (dem Befehl für die Konfiguration von Gerätetreibern) wird dem Treiber die aktuelle Geräteadresse des gewünschten Bausteins mitgeteilt. Ist die Adresse eingestellt, dann kann mit `write()` einfach eine I2C-Nachricht an das Gerät gesendet werden. Das I2C-Kommando für das Setzen der Portexpander-I/O-Pins als Ausgang kennen wir bereits:

```
buffer[0] = 0x06;

buffer[1] = 0x00;
```

Die Kommandos für das Ein- und Ausschalten der LEDs stehen in der Endlosschleife (mit jeweils 100 ms Pause zwischen den Aufrufen).
Auf die gleiche Art und Weise können weitere Bausteine am I2C-Bus angesprochen werden. Für C-Profis ist eventuell auch die Bibliothek *libi2c* [7] interessant, mit der sehr flexibel auf den I2C-Bus zugegriffen werden kann. Aber auch diese

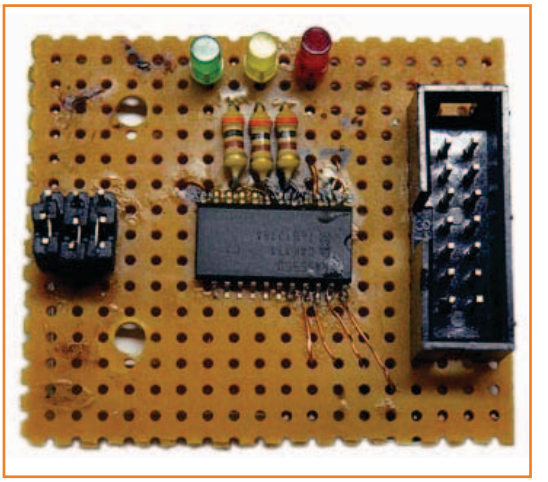


Bild 4.
Drei LEDs dienen zum Testen.

Listing 1: Ansteuerung eines I2C-Portexpanders

```
#include <stdio.h>
#include <fcntl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>

//Slave Address
#define ADDR 0x27

int main (int argc, char **argv)
{
  int fd;
  char filename[32];
  char buffer[128];
  int n, err;

  if (argc == 0) {
      printf(„usage: %s <device>\n“, argv[0]);
      exit(1);
  }
  sprintf(filename, argv[1]);
  printf(„device = %s\n“, filename);

  int slave_address = ADDR;

  if ((fd = open(filename, O_RDWR)) < 0) {
    printf(„i2c open error“);
    return -1;
  }
  printf(„i2c device = %d\n“, fd);

  //prepare communication
  if (ioctl(fd, I2C_SLAVE, slave_address) < 0) {
    printf(„ioctl I2C_SLAVE error“);
    return -1;
  }

  write(fd, buffer, 2);

  //Port-0 GPIO as Output
  buffer[0] = 0x06;
  buffer[1] = 0x00;
  write(fd, buffer, 2);

  n = 0;
  while (1)
  {
    buffer[0] = 0x02;  /* command byte: write
output regs */
    buffer[1] = 0x00;  /* port1 data  */
    write(fd, buffer, 2);

    usleep(100000);
    buffer[0] = 0x02;  /* command byte: write
output regs */
    buffer[1] = 0xff;  /* port1 data  */
    write(fd, buffer, 2);

    printf(„%d\n“, n++);
    usleep(100000);
  }
}
```

Bibliothek verwendet die üblichen ioctl-, write- und read-Funktionen!

UART auf der Konsole

Im vierten Teil der Serie [6] hatten wir bereits einen USB/RS232-Wandler installiert. UART-Schnittstellen können in ein Linux-System auf verschiedenste Wege integriert werden. Die UART-Schnittstellen des Controllers werden meist über die Gerätedateien „/dev/ttyS0“, „/dev/ttyS1“ usw. angesprochen. Die Standardkonsole, auf der man alle Systemmeldungen und den Login sieht, wird zum Beispiel über „/dev/ttyS0“ realisiert. USB/UART-Schnittstellen werden als „/dev/ttyUSB0“, „/dev/ttyUSB1“ etc. eingebunden. Doch unabhängig davon, wie die UART-Schnittstelle integriert wird, man kann auf die immer gleiche Art und Weise darauf zugreifen.

Mit den Linux-Gerätedateien kommt man auch zu einer gewissen Plattformunabhängigkeit: So ist es kein Problem, eine USB/UART-Anwendung komplett am PC zu entwickeln bzw. zu simulieren, denn auch dort werden die USB/UART-Konverter als „/dev/ttyUSB0“ usw. am System angemeldet. Steht die Software auf dem PC, so kann man diese einfach auf das Linux-Board umziehen, dort einmal cross-compilieren und dann mit den richtigen Parametern aufrufen und ausführen lassen. Sollte eine Gerätedatei fehlen, kann man diese wieder mit `mknod` anlegen, wobei immer zuvor der entsprechende Treiber geladen werden muss:

```
mknod /dev/usb/ttyUSB0 c 188 0
mknod /dev/usb/ttyUSB1 c 188 1
mknod /dev/usb/ttyUSB2 c 188 2
```

In vierten Teil der Serie haben wir *microcom* zur Kommunikation genutzt, ein kleines Terminalprogramm für serielle Schnittstellen. Als Parameter kann die gewünschte Baudrate und Gerätedatei angegeben werden. Ist das Programm gestartet, dann werden empfangene Zeichen direkt angezeigt. Gibt man Buchstaben oder Zeichen mit der Tastatur ein, werden diese automatisch über die serielle Schnittstelle ausgegeben.
Klar, dass wir jetzt auch einmal ohne solch ein fertiges Programm auskommen wollen. Das Empfangen und Senden von Daten ist nicht schwer, denn auf die Gerätedatei „/dev/ttyUSB0“ kann man ebenfalls wieder mit unseren einfachen Lese- und Schreibbefehlen zugreifen. Möchte man aber eine bestimmte Baudrate einstellen, dann muss man dies über eine gesonderte Methode machen. Unter Linux kann man dafür das Programm „stty“ verwenden. Um die Baudrate auf z.B. 38400 Baud zu setzen, ruft man folgenden Befehl auf:

```
stty -F /dev/ttyUSB0 38400
```

Jetzt kann man einfach Zeichen versenden mit:

```
echo “Hallo Welt” > /dev/ttyUSB0
```

oder auf Zeichen für den Empfang wie folgt warten:

```
cat < /dev/ttyUSB0
```

Am besten spielt man einfach einmal ein wenig mit diesen Befehlen herum. Wenn man nur einfache Protokolle verwendet, die mit darstellbaren ASCII-Zeichen arbeiten, so kann man manuell schon ganz gut testen und arbeiten. Möchte man aber auch binäre Werte übertragen, muss man zu einem kleinen C-Programm greifen.

UART in C

Der Schritt zu einem eigenen C-Programm ist aber nun nicht mehr besonders groß. Wie auch bereits auf der Konsole muss man zuerst die serielle Schnittstelle konfigurieren. Das wichtigste dabei ist die Baudrate. Im Download zu diesem Artikel [8] findet man ein C-Programm (bezeichnet mit „Listing 2“), dort sieht man das typische Vorgehen.
Als Vorlage für dieses Programm wurden die Beispiele aus [9] verwendet. Im oberen Teil des Programms wird mit Hilfe der Datenstruktur `struct termios options` die Grundkonfiguration vorgenommen. Mit `tcgetattr` wird zuerst die aktuelle Konfiguration in die Datenstruktur übernommen, wo sie dann modifiziert werden kann. Die Datenrate stellt man mit `cfsetispeed` (für eingehende Zeichen) und `cfsetospeed` (für ausgehende Zeichen) ein. Die Anzahl der Daten-, Start- und Stoppbits lässt sich hier ebenfalls einstellen. Anschließend wird die neue Konfiguration dem Treiber mit `tcsetattr` übergeben.
Das Versenden und Empfangen von Daten funktioniert wieder wie bei jedem Standard-Gerät. Mit `write()` kann man über das Handle – das zuvor mit `open()` geöffnet wurde – Daten senden und mit `read()` entsprechend Daten lesen. Im Beispiel erwartet unsere Gegenstelle als Kommando einen 6 Byte langen Wert. Man sieht beispielsweise, dass an die Variable `buffer[0]` eine 0 übergeben wird.

Als Antwort erwartet unser Programm von dem angeschlossenen Gerät ebenfalls wieder 6 Bytes. Das Empfangen ist hier mit einer Schleife gelöst, die insgesamt sechs Zeichen einliest. An der Stelle des read-Befehls innerhalb der Schleife wartet das Programm, bis das nächste Zeichen kommt (blockierender Aufruf). Das ist keine optimale Lösung,

Autor Benedikt Sauter im Elektor-Interview. Am Farnell/element14-Stand auf der Electronica 2012 waren gleich mehrere Kameraleute im Einsatz. Das Video kann man ansehen unter [13].

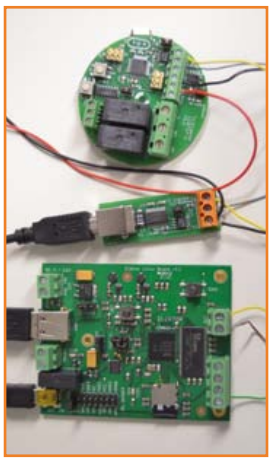


Bild 5.
RS485 mit dem ElektorBus.

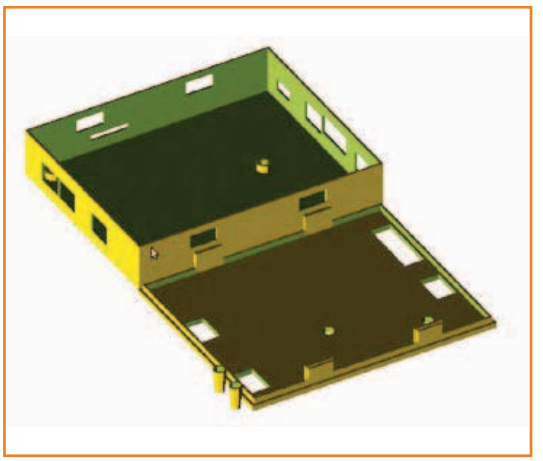


Bild 6.
Gehäuse für das Elektor-Linux-Board.

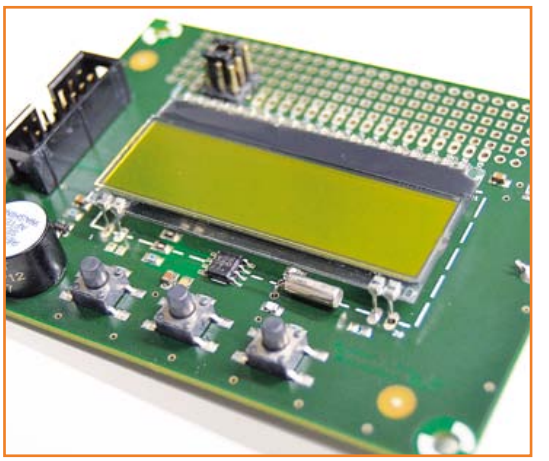


Bild 7.
Das Erweiterungsboard stellen wir im nächsten Heft vor.

da die Datenübertragung nur mit 19200 Baud erfolgt. Unser schneller Controller macht in der Schleife dann fast nichts anderes, als untätig auf Zeichen zu warten, doch trotzdem nimmt das kostbare Rechenzeit in Anspruch.

Hier wünscht man sich typischerweise einen Interrupt. Wenn Daten vorliegen, wird man benachrichtigt und greift auf den Empfangsspeicher zu. Mit einem Betriebssystem wie Linux lässt sich dann ein ressourcensparender Programmablauf erreichen. Sind keine Daten vorhanden, kann das empfangende Programm etwas anderes machen oder auch durch Linux schlafengelegt werden. Erst wenn wieder Daten eingehen, wird das Programm aufgeweckt, um die Daten zu verarbeiten. Im dritten Programm im Downloadordner [8] ist so ein Interrupt-Verhalten verwirklicht. Im Wesentlichen handelt es sich um das gleiche Beispiel wie eben, nur gibt es jetzt eine eigene Funktion `signal_handler_IO`, die aufgerufen wird, wenn Daten empfangen wurden.

RS485 mit dem ElektorBus

Wenn wir mit dem UART arbeiten können, dann sollten wir nun auch in der Lage sein, über eine RS485-Schnittstelle Bytes zu senden und zu empfangen. Nur dass wir jetzt statt eines USB/RS232-Konverters einen USB/RS485-Wandler an die USB-Schnittstelle unseres Boards anschließen müssen. Bei Elektor kann so ein Konverter unter der Nummer 110258-91 bestellt werden [8]. Er enthält einen FTDI-Chip; der entsprechende Treiber ist im Kernel bereits vorhanden und muss nur noch (so wie im vierten Teil beschrieben) im Kernel aktiviert werden.
Als RS485-Gegenstelle benutzen wir eine Elektor-Bus-Installationsplatine [8][10], die zwei Relais mitbringt (**Bild 5**). Für das ElektorBus-Protokoll muss man einfach Listing 2 etwas anpassen, zum Beispiel müssen die Baudrate auf 9600 Baud eingestellt und die Variable LENGTH auf 16 gesetzt werden. Das Array `Buffer` wird dann vor dem Versenden der RS485-Nachricht mit den entsprechenden 16 Bytes befüllt. Um das Relais 1 auf der Installationsplatine zu schalten, muss man folgendes Kommando senden:

```
170,0, 0,5,0,10, 96,1,0,0, 0,0,0,0, 0,0
(anziehen)
170,0, 0,5,0,10, 96,0,0,0, 0,0,0,0, 0,0
(abfallen)
```

Für Relais 2 lauten die Bytes:

```
170,0, 0,5,0,10, 0,0,96,1, 0,0,0,0, 0,0
(anziehen)
170,0, 0,5,0,10, 0,0,96,0, 0,0,0,0, 0,0
(abfallen)
```

Mehr zum ElektorBus findet man unter [11].

Ausblick

Wer auf der Suche nach einem passenden Gehäuse für das Elektor-Linux-Board (**Bild 6**) ist, wird mittlerweile im Internet fündig [12]. Unter dem Link findet man ein passendes 3D-Modell für einen 3D-Drucker. Es ist so konstruiert, dass man

keine Schrauben benötigt, sondern einfach mit einem Schnappmechanismus Deckel und Grundkörper zusammenstecken kann.
Dies war der letzte Teil unseres Embedded-Linux-Kurses, doch damit sind wir noch längst nicht am Ende des Projekts angekommen! Mit einer neuen Platine (**Bild 7**) erweitern wir das Elektor-Linux-Board um viele sehr nützliche Funktionen:

Display (2 x 16-Zeichen)
drei Taster als Bedieneinheit bzw. Steuerung für Menüs
16 zusätzliche digitale Ein- und Ausgänge
RTC (Echtzeituhr) mit Batterie zur Versorgung des Systems mit der Uhrzeit
Summer zur akustischen Ausgabe von Meldungen
Experimentierfeld für weitere Schaltungen

In der nächsten Ausgabe stellen wir dieses Board ausführlich vor!

(120518)

Weblinks

[1] sauter@embedded-projects.net
[2] www.gnublin.org
[3] www.elektor.de/120180
[4] www.elektor.de/120578
[5] http://en.gnublin.org/index.php/Eclipse
[6] www.elektor.de/120181
[7] http://opensource.katalix.com/libi2c/
[8] www.elektor-magazine.de/120518
[9] www.tldp.org/HOWTO/Serial-Programming-HOWTO/
[10] www.elektor.de/110727
[11] www.elektor.com/elektorbus
[12] www.thingiverse.com/thing:29314
[13] www.element14.com/community/community/events/electronica
[14] http://eagleup.wordpress.com
[15] www.elektor.de/120182

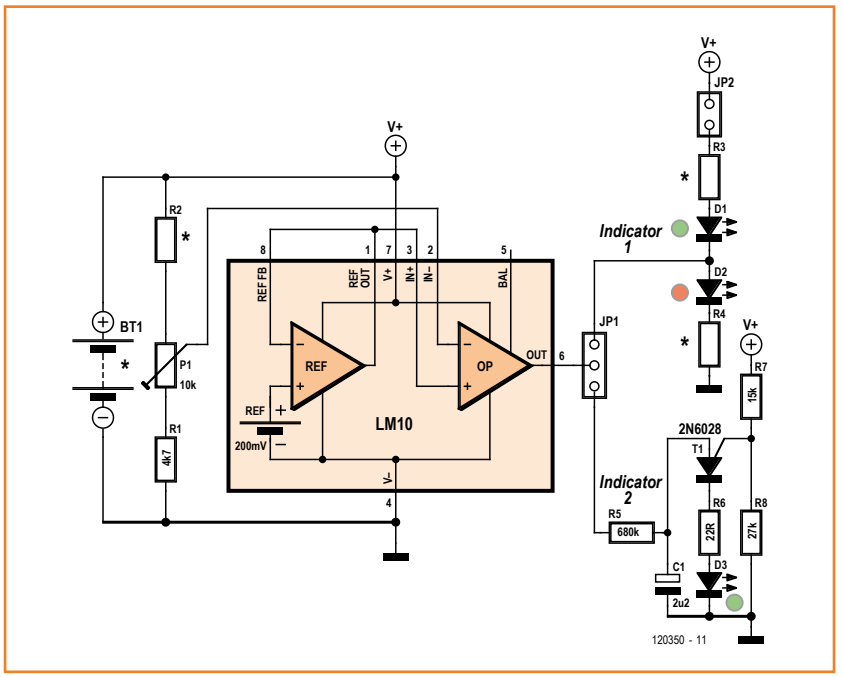
Batterie fast leer?

Schutz vor Totalausfall

Von **J.F. Verrij** (NL)

Dieser Wächter wacht über den inneren Zustand von Batterien oder Akkus. Wenn die Spannung unter einen voreinstellbaren Wert sinkt, warnt eine rote LED vor dem bevorstehenden Blackout.

Im LM10 sind eine Spannungsreferenz 200 mV, ein nachgeschalteter Puffer und ein Opamp integriert. Der Opamp vergleicht die herabgeteilte Batterie- oder Akkuspannung mit der Referenzspannung. Über den Opamp-Ausgang wird Alarm ausgelöst, sobald die zu überwachende Spannung den mit P1 festgelegten Wert U_{min} unterschreitet. Für die Anzeige stehen bei diesem Projekt zwei Versionen zur Wahl:



Die erste Version arbeitet mit einer roten und einer grünen LED. Wenn das Ende des Batterielebens naht oder der Akku nachgeladen werden muss, leuchtet die rote LED auf. Die grüne LED signalisiert, dass der Wächter in Betrieb ist.

Bei der zweiten Version steuert ein Unijunction-Transistor eine rote Low-Current-LED. Der Unijunction-Transistor lässt die LED periodisch aufblitzen, so dass der Alarm auch bei hellem Umgebungslicht noch in einiger Entfernung auffällt. Das ist von Vorteil, wenn der Wächter beispielsweise über ein ferngesteuertes Flug- oder Schiffsmodell wachen soll.

Bevor der Spannungsteiler (R2, P1 und R1) berechnet werden kann, muss die Spannungsschwelle U_{min} festgelegt werden. Für LiPo-Zellen ist 3,3 V pro Zelle ein Wert, bei dem die Zelle noch nicht vollständig entladen ist. Flug- oder Schiffsmodelle können den heimatlichen Hafen mit letzter Kraft erreichen, ohne dass die Zelle den kritischen Wert 3 V unterschreitet. Nur selten überstehen LiPo-Zellen unbeschadet Tiefentladungen unter der 3-V-Grenze. Ein Akku aus zwei LiPo-Zellen hat die untere Spannungsgrenze $U_{min} = 2 \cdot 3{,}3\ V = 6{,}6\ V$. Die untere Grenze einer NiCd-Zelle beträgt 1,1 V, so dass für einen Akku aus sechs NiCd-Zellen ebenfalls der Grenzwert $U_{min} = 6{,}6\ V$ gilt. R2 lässt sich wie folgt berechnen:

$$R2 = (48{,}5 \cdot U_{min} - 14{,}7)\ k\Omega$$

Daraus folgt für $U_{min} = 6{,}6\ V$:
$R2 = (48{,}5 \cdot 6{,}6 - 14{,}7) = 305{,}4\ k\Omega$.

Der nächste Wert der Normreihe E12 ist 330 kΩ. Die Spannungsschwelle U_{min} ist mit P1 justierbar, wenn anstelle von Akku oder Batterie ein Labornetzteil angeklemmt wird. Das Netzteil muss auf exakt 6,6 V eingestellt werden. P1 wird bis zu dem Punkt gedreht, an dem die LED ihren Zustand wechselt.

Auch die Strombegrenzungswiderstände R3 und R4 der LEDs lassen sich berechnen. Da der Spannungsunterschied zwischen roten und grünen LEDs klein ist, soll hier 2 V als Näherungswert dienen. Low-Current-LEDs leuchten schon beim Strom 3 mA genügend hell. Mit $U_B = 6{,}6\ V$ gilt:

$$R_S = (U_B - 2) / I_{LED} = 1533\ \Omega$$

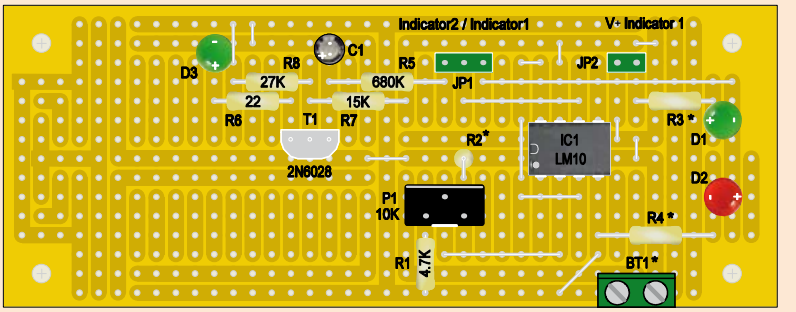
Der nächste Wert der Normreihe E12 ist 1,5 kΩ.

Stückliste

Widerstände:
R1 = 4k7
R2 = siehe Text
R3,R4 = siehe Text
R5 = 680 k
R6 = 22 Ω
R7 = 15 k
R8 = 27 k
P1 = 10 k Trimmpoti

Kondensatoren:
C1 = 2µ2/16 V

Halbleiter:
D1 = LED grün
D2 = LED rot
D3 = LED rot, Low Current
T1 = 2N6028
IC1 = LM10



Außerdem:
Elex-Platine

Auf einer Elex-Platine oder einem Stück Laborkarte ist die Schaltung schnell aufgebaut. Das Foto zeigt einen Prototyp, auf dem zur Demonstration beide LED-Anzeigen vorhanden sind.

Die kleine Schaltung findet in den meisten zu überwachenden Objekten schnell ihren Platz.

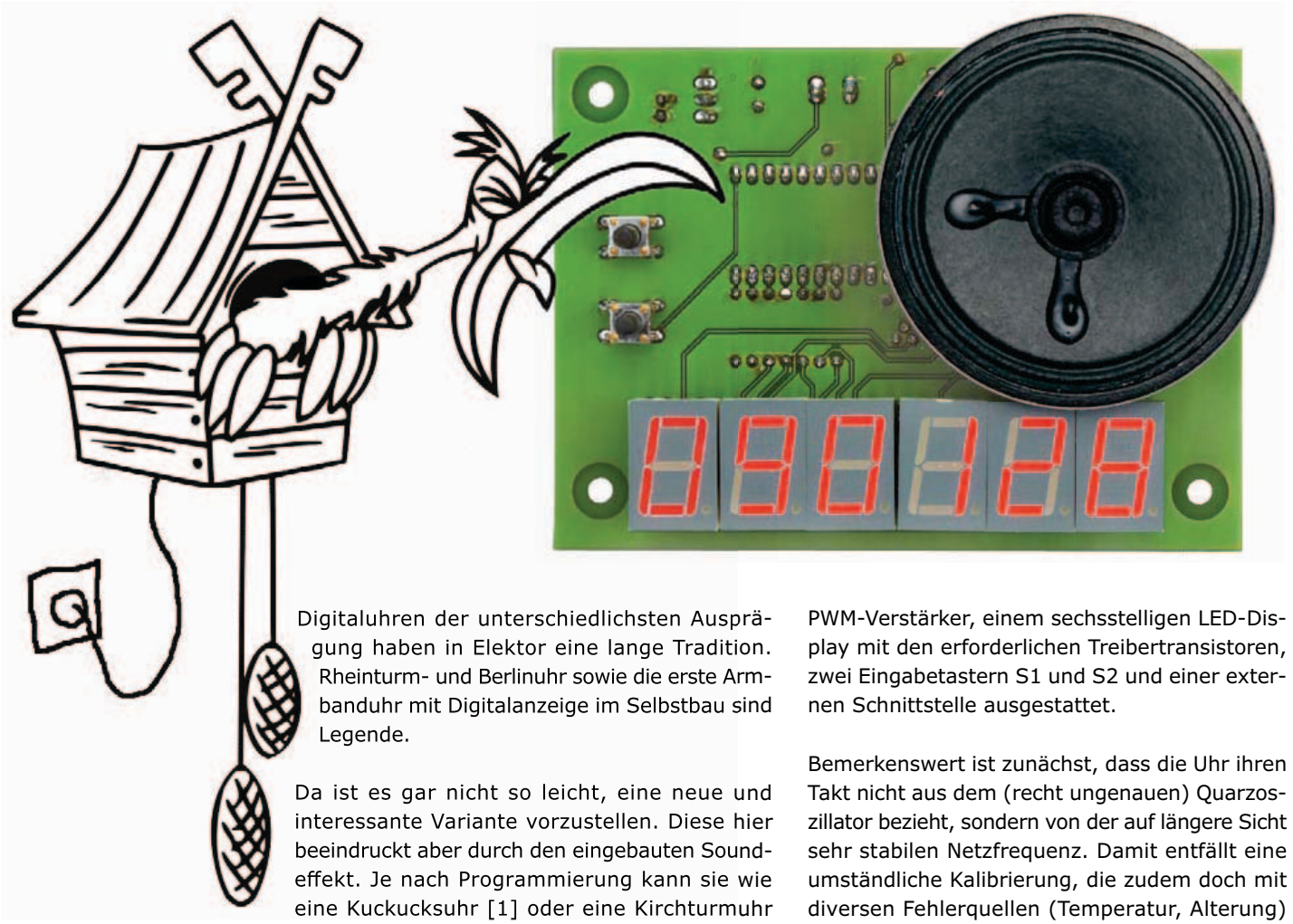
(120350)gd

Digitaluhr mit Sound

Kuckuck oder Glocke?

Von Martin Ossmann

Schon wieder eine Uhr? Diese hier ist zwar einfach aufgebaut, verkündet aber Viertel- und volle Stunden durch Glockengeläut, Kuckucks-Rufe oder beliebige andere Sounds.



Digitaluhren der unterschiedlichsten Ausprägung haben in Elektor eine lange Tradition. Rheinturm- und Berlinuhr sowie die erste Armbanduhr mit Digitalanzeige im Selbstbau sind Legende.

Da ist es gar nicht so leicht, eine neue und interessante Variante vorzustellen. Diese hier beeindruckt aber durch den eingebauten Soundeffekt. Je nach Programmierung kann sie wie eine Kuckucksuhr [1] oder eine Kirchturmuhr klingen. Dabei werden die Viertelstunden durch eins, zwei, drei, vier Glockenschläge angedeutet, zudem ertönt zur vollen Stunde eine andere Glocke, um die Tagesstunde zu signalisieren.

Die Schaltung

Schalt- und Waltzentrale der Kirchturmuhr in **Bild 1** ist ein Mikrocontroller ATmega32 von Atmel. Neben dem obligatorischen Taktgeber mit einem 16-MHz-Quarz und der 5-V-Spannungsstabilisierung mit IC1 ist der Controller mit einem PWM-Verstärker, einem sechsstelligen LED-Display mit den erforderlichen Treibertransistoren, zwei Eingabetastern S1 und S2 und einer externen Schnittstelle ausgestattet.

Bemerkenswert ist zunächst, dass die Uhr ihren Takt nicht aus dem (recht ungenauen) Quarzoszillator bezieht, sondern von der auf längere Sicht sehr stabilen Netzfrequenz. Damit entfällt eine umständliche Kalibrierung, die zudem doch mit diversen Fehlerquellen (Temperatur, Alterung) behaftet ist. Dies bedeutet aber auch, dass die Spannungsversorgung direkt aus einem Netztrafo beziehungsweise aus einem Wechselspannungs-Steckernetzteil erfolgen muss. Die Wechselspannung gelangt über R1 zum Interrupt-Eingang INT0 (PD2) des Controllers.

Das restliche „Netzteil“ fällt äußerst simpel aus und besteht nur aus drei Bauteilen. D1 sorgt für eine Einweggleichrichtung. Die pulsierende Gleichspannung wird von C1 einigermaßen geglät-

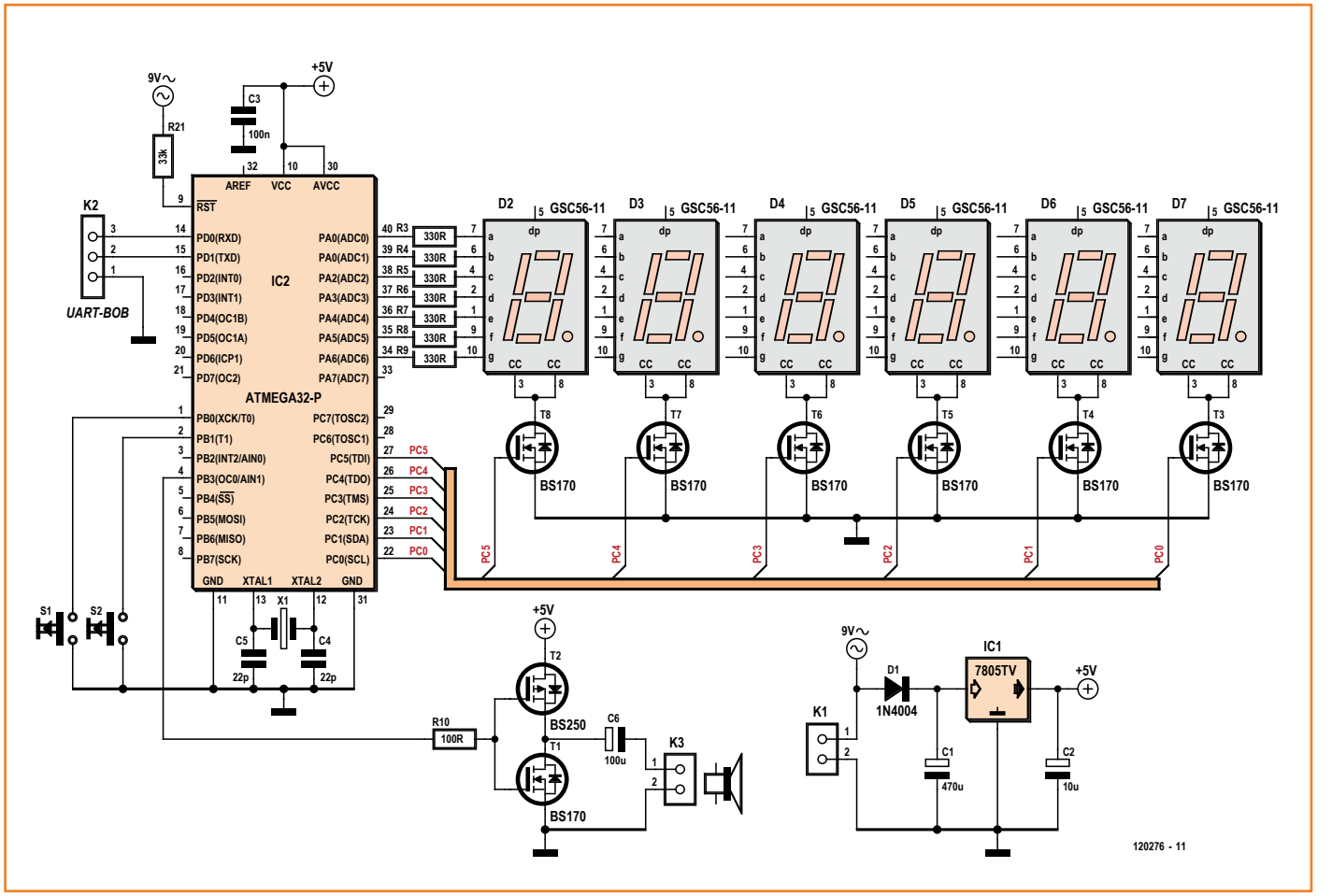


Bild 1.
Ein Mikrocontroller und sechs LED-Displays.

tet, so dass der Festspannungsregler 7805 daraus eine brauchbare 5-V-Gleichspannung für den Rest der Schaltung erzeugen kann.

Ein- und Ausgänge

Da die Ausgangsstufe des ATmega natürlich nicht viel Strom liefern kann, ist eine Treiberstufe für den Lautsprecher notwendig. Diese Stufe besteht einfach aus zwei MOSFETs (T1 und T2), einem P-Kanal-Typ BS250 und einem N-Kanal-Typ BS170. Die damit zu erreichende Lautstärke reicht vermutlich für alle Anwendungen.
Die Uhrzeit wird im Format hh:mm:ss von sechs LED-Siebensegmentdisplays angezeigt. Auch hier werden die Ports des Controllers von externen N-Kanal-Treibertransistoren vom Typ BS170 (T3...T8) unterstützt. Die Segmentströme werden von R3...R9 beschränkt.

Mit den Tastern S1 und S2 kann man die Uhrzeit einstellen. Wenn man im normalen Uhrzeitmodus Taste S2 drückt, gelangt man in den Stellmodus und die Stunden-Zehner-Ziffer blinkt. Mit Taster S1 kann man diese Stelle dann einstellen. Danach drückt man wieder S2, sodass die Stunden-Einer Stelle blinkt und so weiter. So stellt man alle Ziffern nacheinander ein und bringt durch einen letzten Druck auf S0 die Uhr wieder in den normalen Modus.

Wave-Sounds

Zur Analogausgabe wird Timer0 im Fast-PWM-Modus benutzt. Die PWM-Frequenz liegt dann bei 16°MHz/256 = 62,5°kHz, sie liegt also deutlich über der Hörbarkeitsgrenze. Der Sound wird mit 11025 Samples/Sekunde abgespielt, und zwar gesteuert von Timer 1, der immer bis jeweils 16°MHz/11025°Hz = 1451 zählt. Der Sound ist in einem Feld namens `sound` von 8-Bit-Werten im Programmspeicher gespeichert und hört sich so an wie eine Glocke. Die Interruptroutine für Timer 1 dazu sieht wie unten gezeigt aus. Das Abspielen wird einfach dadurch gestartet, dass man den Wert von `samplPtr` auf 0 setzt. Der zweite

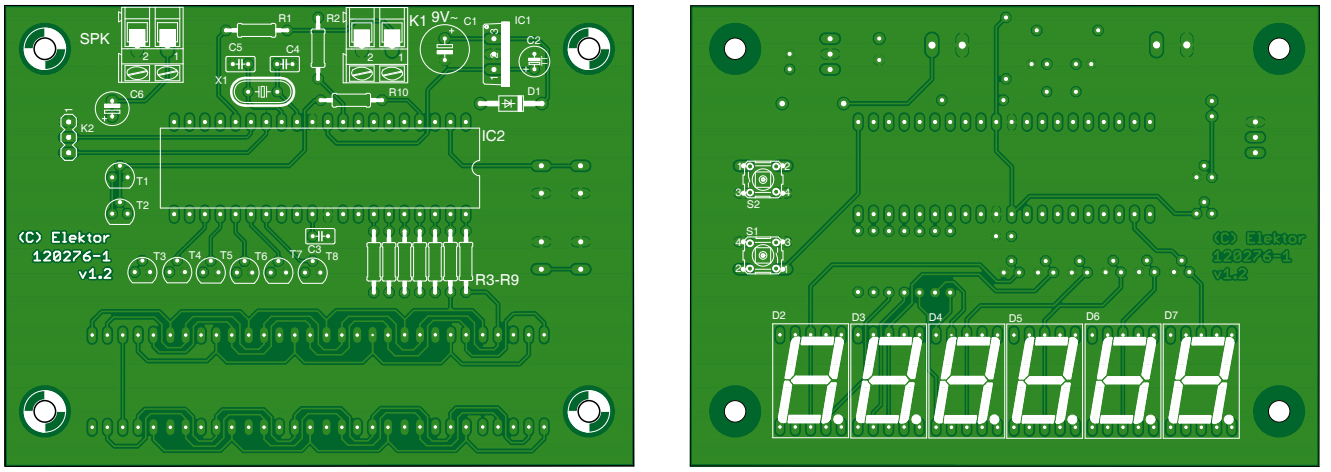


Bild 2.
Bestückung der doppelseitigen Platine (hier 80%). Displays und Taster werden auf der Rückseite angebracht.

Glockenklang wird einfach durch ein schnelleres Abspielen des Sounds erreicht.
Wer eigene Sounds wie den Kuckuck verwenden will, kann das auch tun. Dazu muss man sie in Stereo mit 11025°Samples/s in einer Wave-Datei speichern. Mit dem Java-Programm convert1.jar kann man eine solche Datei dann in eine Include-Datei für die Uhr konvertieren, die dann beim Compilieren des Projektes benutzt wird. Bei der Software sind jeweils Batch-Dateien zu finden, die zeigen, wie man das macht. Im ATmega32 sind circa 24k frei, um einen Sound zu speichern, das entspricht in etwa 24000/11025 = 2,1°s.

Schnittstelle

Die serielle Schnittstelle an K2 führt die beiden USART-Leitungen TxD und RxD (PD0 und PD1) des Controllers nach außen. Hier kann man beispielsweise einen USB/TTL-Konverter anschließen. Möchte man die Schnittstelle direkt mit der RS232-Schnittstelle des PCs koppeln, ist eine Pegelanpassung erforderlich.
Die Schnittstelle ermöglicht zwei Funktionen, nämlich die Ausgabe der Uhrzeit und das Stellen der Uhr. Beides geschieht über ein Terminalprogramm (zum Beispiel HyperTerminal) mit 19200 Baud. Die Uhrzeit wird sekündlich aktualisiert, zur Einstellung gibt man die gewünschte Uhrzeit (beginnend mit den Stunden) ein und tippt den Buchstaben „s“. Es erscheint beispielsweise

Clock 09:01:52 Type ‚s‘ to set clock to: 00:00
Clock 09:01:53 Type ‚s‘ to set clock to: 00:00
...

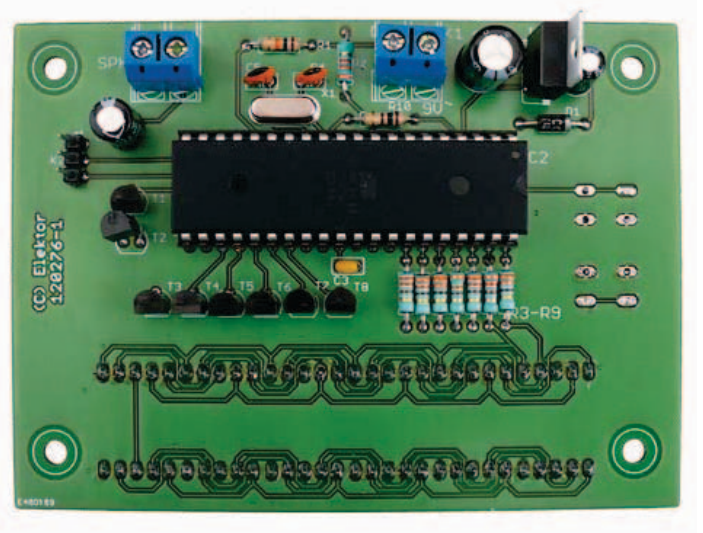
Stückliste

Widerstände:
R1 = 10 k
R2 = 33 k
R3...R9 = 330 W (oder 7-fach-DIL-Array)

Kondensatoren:
C1 = 470 µ/16 V
C2 = 10 µ/16 V
C3 = 100 n
C4, C5 = 22 p
C6 = 100 µ/10 V

Halbleiter:
D1 = 1N4004
LD1...LD6 = 7-Segment-Anzeige SC56-11SRWA (rot) (Kingbright)
IC1 = 7805
IC2 = ATmega32P
T1,T3...T8 = BS170
T2 = BS250

Außerdem:
X1 = Quarz 16 MHz
S1,S2 = Taster
K1,K3 = 2-polige Platinenanschlussklemme
K2 = 3-poliger Pfostenverbinder
40-polige IC-Fassung



Gibt man nun ‚0845' über das Terminal ein, ändert sich etwas:

Clock 09:01:56 Type ‚s' to set clock to: 08:45
Clock 09:01:57 Type ‚s' to set clock to: 08:45
...

Nun ein Druck auf ‚s' und die neue Uhrzeit ist gestellt:

Clock 08:45:00 Type ‚s' to set clock to: 08:45
Clock 08:45:01 Type ‚s' to set clock to: 08:45
...

Der ATmega32 ist zwar prinzipiell in-system-programmierbar, eine entsprechend herausgeführte Schnittstelle ist aber auf der Platine nicht vorgesehen. Um den Controller zu programmieren (zum Beispiel für neue Sounds) muss man ihn aus der Fassung holen und in einen Programmieradapter einsetzen.

Aufbau: ein Kinderspiel

Die digitale Kuckucksuhr ist dank der übersichtlichen doppelseitigen Platine (**Bild 2**), die im Elektor-Service [2] erhältlich ist, in Nullkommanix aufgebaut. Alle Bauteile sind bedrahtet, so dass keinerlei Schwierigkeiten bei den Bestückungsarbeiten zu erwarten sind.
Die Displays und die beiden Taster sind auf der Platinenrückseite angebracht.
Der Controller sollte (damit neue Sounds eingefügt werden können) in einer Fassung untergebracht werden. Der ATmega32 ist fertig programmiert erhältlich (im Elektor-Service), man kann aber auch selbst ans Programmierwerk gehen. Der C-Sourcecode ist ebenfalls auf der Elektor-Site [2] kostenlos im Netz zu finden.

(120276)

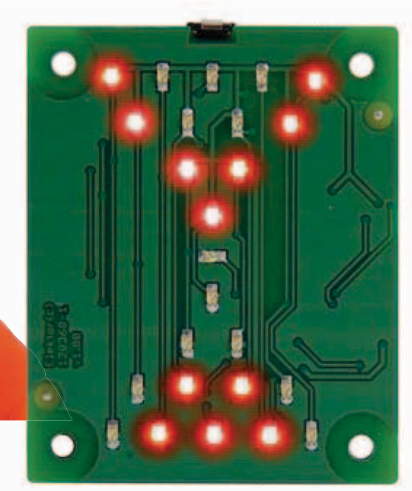
Weblinks

[1] www.fuglar.no/galleri/lyder/Cuculus.canorus.mp3
[2] www.elektor-magazine.de/120276

Sound-Interruptroutine

```
prog_int8_t sound []={
137, 138, 140, 135, 138, 124,
........
........ 129, 128, 124, 127,
} ;

#define Nsamples 18000

ISR(TIMER1_OVF_vect) {
if ( samplePtr<Nsamples){
OCR0=pgm_read_byte( sound+samplePtr);
samplePtr++ ;
}
```

Zahnputztimer für Kinder



Von **Jere Manner**
(Finnland)

Eine klassische Sanduhr verbreitet eine Sphäre von Authentizität und Nostalgie. Obwohl sie keine Elektronik enthält und auch keine Klingel- und Piepstöne produziert, fasziniert es viele kleine Kinder, wenn der Sand sanft nach unten rieselt und sich durch nichts dabei aufhalten lässt. Time is rolling away…
Im Gegensatz dazu lassen sich Kinder selten von ihren Eltern vom rechten „Timing“ überzeugen, zum Beispiel, wenn es sich um das ungeliebte Ritual des Zähneputzens vor dem Zubettgehen handelt. Hier hilft eine spielerische, kinderfreundliche Herangehensweise, die gleichzeitig auch den Erzieher befriedigt, wenn er denn C-Programmierer und/oder AVR-Nerd ist. Der hier gezeigte Timer ähnelt in Funktion und Aussehen einer Sanduhr und bedarf keinerlei Erläuterung: Die Kinder verstehen ihn intuitiv. Wenn sie Fernseher und WLAN-Router einschalten können, dann können sie auch mit diesem Timer umgehen.
Um den Timer zu starten, drückt man einfach auf die Taste. Die LEDs leuchten sequentiell, um die fallenden Sandkörnchen zu imitieren, und ein maßvoller Piepser ertönt alle 30 Sekunden, wenn die Putzzeit abgelaufen ist. Eine LED entspricht dabei zehn Sekunden.
Es handelt sich um ein Musterbeispiel für eine Schaltung, bei der ein kleiner Mikrocontroller eine Unzahl von klassischen TTL-, CMOS- oder NE555-Timer-ICs ersetzen kann. Die Firmware

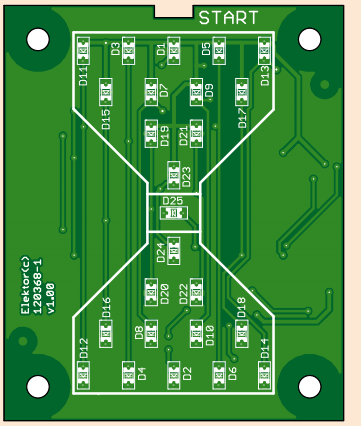
Stückliste

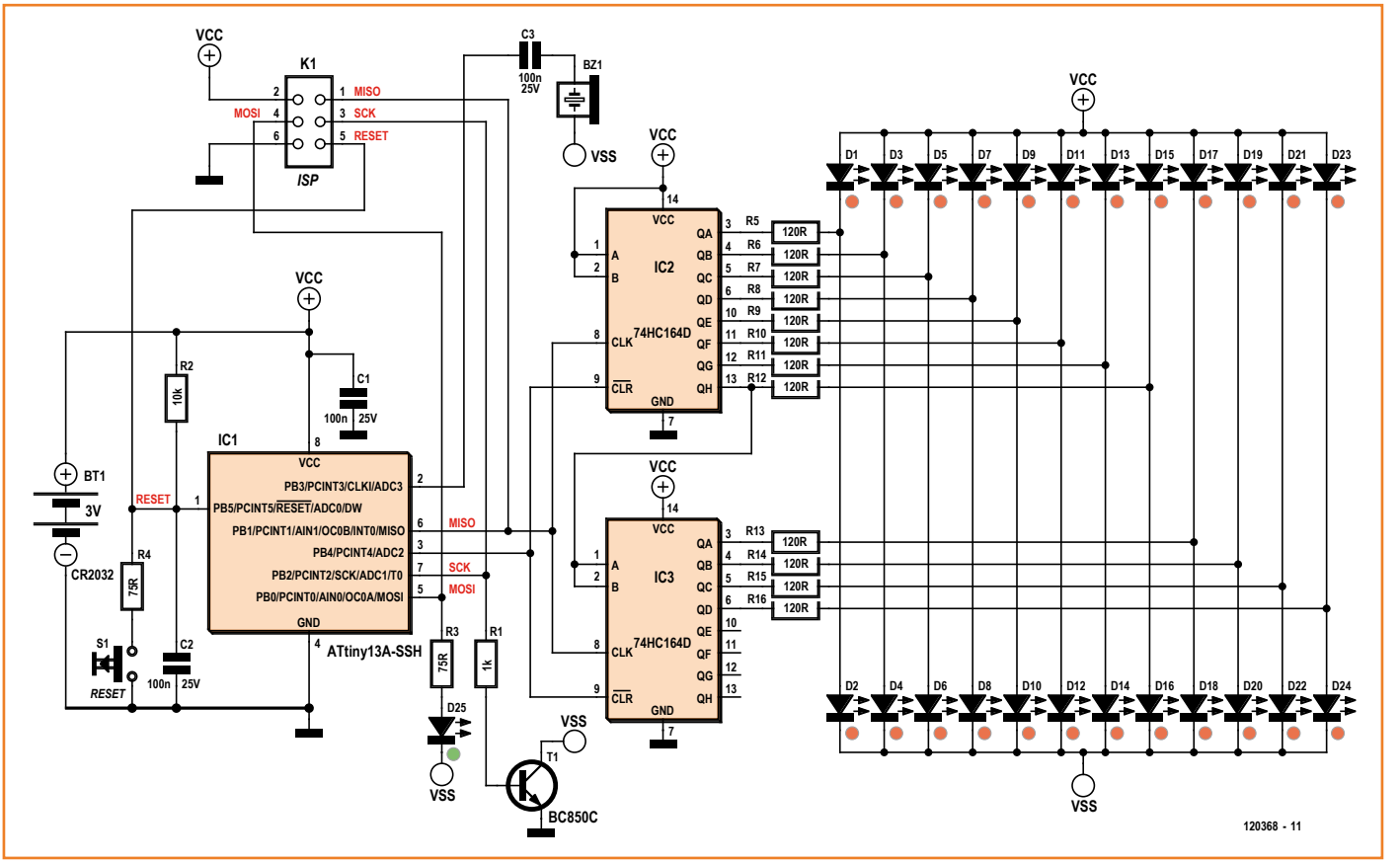
Widerstände
(SMD 0805)
R1 = 1 k 1%
R2 = 10 k 1%
R3,R4 = 75 Ω 1%
R5…R16 = 120 Ω 1%

Kondensatoren
C1,C2,C3 = 100 n 25V, SMD 0805

Halbleiter
D1…D24 = LED, rot, SMD 0805, z.B. Kingbright KPHCM-2012SURCK; Farnell 1686067
D25 = LED, grün, SMD 0805, z.B. Kingbright KPHCM-2012CGCK; Farnell 1686075
IC1 = ATtiny13A-SSH, SO-8-Gehäuse, programmiert: Elektor 120368-41
IC2,IC3 = 74HC164D, SOIC-14-Gehäuse
T1 = BC850C, SOT-23-Gehäuse

Außerdem
BT1 = Knopfzellenhalter und CR2032-Batterie; Farnell 2064725
BZ1 = Piezo-Summer, Farnell 1192551
K1 = 2x3-poliger Pfostenverbinder RM°2,5mm
S1 = Taster, gewinkelt, Farnell 1761633
Platine 120368-1, siehe www.elektor-magazine.de/120368





eines ATtiny13-Mikrocontrollers steuert den Summer Bz1 sowie zwei Schieberegister vom Typ 74HC164 (IC2 und IC3), die ihrerseits ein Array von SMD-LEDs über die Strombegrenzungswiderstände R5…R16 treiben.

Der Clou des Projekts ist kaum seine Elektronik oder Software, sondern liegt in seinem Erscheinungsbild, das heißt, dem Platinenlayout und dem Gebrauch kleiner Bauteile für ein kompaktes, leichtgewichtiges Gerät. Das Original des Autors wurde im Elektor-Labor leicht modifiziert. So wurde R4 hinzugefügt und ein Header für ISP (in system programming), damit die Software auch in Eigenregie geflasht und modifiziert (zum Beispiel auf die empfohlene Zahnputzdauer von 3...5°min) werden kann, außerdem wurden ein paar Bauteile durch leichter erhältliche Äquivalente ersetzt.

Eine Anmerkung noch zur Batterie: Die angegebene Knopfzelle CR2032 hält nicht besonders lange, obwohl der Strom im ausgeschalteten Zustand recht klein ist. Wer mag, ersetzt die Knopfzelle durch zwei leistungsfähigere Micro- oder Mignonzellen. Der in C geschrieben Sourcecode ist frei unter [1] verfügbar.

(120368)

Elektor-Labor Tipps & Tricks

Kleben Sie den Batteriehalter auf die Platine. Wir haben den Halter zuerst von der Platine gerissen, als wir die Batterie wechseln wollten ☹.
Programmieren Sie den Controller, bevor Sie ihn festlöten. Oder benutzen Sie ein externes Netzteil. Beim Programmieren sackte die Batteriespannung von 3,0°V auf 2,3°V → Controller-Reset, keine Funktion mehr → ☹.
Ein Bolzen oder eine Schraube an einer Platinenecke stellt einen industriell anmutenden Low-cost-Ständer für den Timer dar.

[1] www.elektor-magazine.de/120368

Herausforderungen

Entwerfen Sie ein Gehäuse mit einem 3D-Drucker.
Peppen Sie die Stromquelle auf, um den LED-Strom zu reduzieren.
Fügen Sie einen Tilt-Schalter hinzu, mit dem der Timer wie eine Sanduhr durch Umdrehen gestartet werden kann.
Tipp: „Motorrad-Alarm“, www.elektor.de/120106.
Berichten Sie auf www.elektor-labs.com und teilen Sie Ihre Ideen anderen mit.

Arduino auf Kurs (4)

Pflanzenbewässerung für kommunale Zwecke

Von **David Cuartielles** (Schweden/Spanien)

Auf dem letzten Elektor-Expertentreffen ging es einen ganzen Nachmittag darum, wie man Projekte entwickelt und was die heißen Themen sind, die in Zukunft in Elektor zu lesen sein sollten. Während einer dieser Sessions packte der Elektor-Redakteur Jan Buiting plötzlich einige alte Bauelemente aus und forderte die Teilnehmer dazu auf, nach diesem Treffen einmal in die eigenen Bastelkisten zu schauen… Denn sicher schlummern hier überall noch wahre Schätze, die man einer weiteren Verwendung zuführen könnte!

Wieder zuhause angekommen, schaute ich tatsächlich nach meinen vergessenen Bauteilen. Es hatten sich viele Dinge angesammelt, die ich für experimentelle Projekte gekauft hatte, welche niemals fertig gestellt wurden. In einer dieser Schachteln fand ich ein Punkt-Matrix-Display und einige Taster. Da kam mir die Idee, daraus eine Art Uhr zu bauen. Also ergänzte ich die alten Bauteile durch einen neuen RTC-Chip (**R**eal **T**ime **C**lock) und machte mich an den Aufbau. Das Ergebnis sehen Sie oben abgebildet. Die Beschreibung können Sie nachfolgend lesen.

Material

Die Zutaten zu diesem Spontan-Experiment sind:

- Arduino Uno Board
- Prototyping-Shield
- Pinheader (Stecker)
- USB-Kabel
- Zwei Taster
- Punkt-Matrix-Display mit HT1632-Controller (Modell 32x16 von Sure)
- RTC DS1302
- 32-kHz-Quarz
- Batteriehalter für CR2032
- Batterie-Clip für 9-V-Batterien
- 9-V-Batterie (für Arduino) + CR2032-Batterie (für den RTC)
- Ein Gehäuse zum Einbau der Teile

Hinweis: Die Elektronik wurde in eine Pappschachtel eingebaut. Es sind aber viele kreative Verpackungslösungen denkbar, vom Recycling einer Butterbrotdose bis zum professionellen Kunststoffgehäuse. Was passt ist okay!

Mein Motiv: Etwas Nützliches bauen

Ich habe dann einige Gedanken an die Verwertbarkeit dieses Elektronikschrotts verschwendet. Ich wohne in einem Apartment-Haus und da gibt es einige Pflanzen, die allen und niemand gehören. Eigentlich sollte sich jeder Bewohner gelegentlich um diese kümmern. Da aber nicht so klar ist, wann welche Pflanze zum letzten Mal gegossen wurde, enden diese Pflanzen im Treppenhaus in schöner Regelmäßigkeit als traurige Trockensubstanz in Töpfen. Wüste im Kleinformat...
Meiner Ansicht nach kümmert sich deshalb kaum jemand um diese Pflanzen, weil jeder annimmt, dass es ein anderer täte - nicht aus Desinteresse. So reifte in mir der Plan, einen Timer zu bauen,

der anzeigt, wann „irgendjemand“ den Pflanzen zum letzten Mal etwas Wasser spendiert hat.
Das Projekt sollte schön einfach bleiben, weshalb seine Aufgaben auf die reine Zeitanzeige seit der letzten Wasserspende beschränkt wurden. Eine Messung der Erdfeuchtigkeit und anderer Umweltbedingungen sollte außen vor bleiben. Auch sollte das Projekt nicht via Internet an das Gießen erinnern. Es gibt nämlich schon ein anderes Projekt namens Botanicalls [1], das genau dies kann.
Es sollte ein Gerät werden, das meine Nachbarn (und mich) schlicht darüber informiert, wann jemand zum letzten Mal Erbarmen mit den Pflanzen hatte. Für diesen Zweck passt das Paradigma einer Schachuhr: Die Spieler drücken nach jedem Zug eine Taste und ab da läuft die Uhr (für den anderen Spieler). Mein Timer sollte ganz ähnlich funktionieren. Wenn jemand Wasser in die Blumenkübel gekippt hat, drückt er auf einen Knopf und übergibt damit die Verantwortung an einen anderen Hausbewohner. Ein Druck auf einen zweiten Knopf zeigt an, wie lange die letzte Wässerungsaktion schon her ist.
Nach diesen Überlegungen begann ich, die notwendigen Bauteile zusammenzusuchen (**Bild 1**).

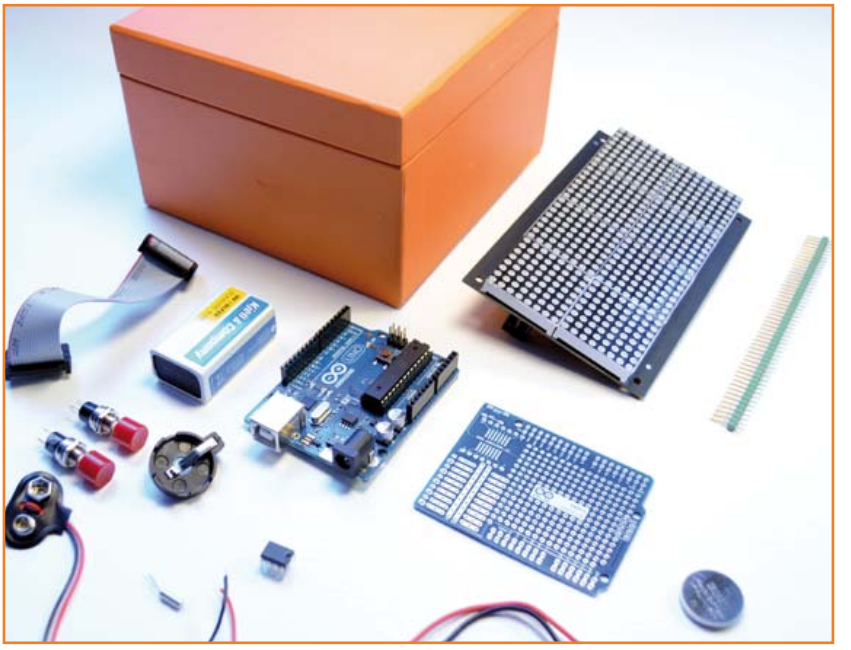


Bild 1.
Die Teile, die zu einem Pflanzenbewässerungstimer auf Arduino-Basis werden.

Steuerung des Punkt-Matrix-Displays

Blinkende LEDs müssen ein Geheimnis haben, das Leute jeden Alters fasziniert. Und wenn man schon einmal eine gewöhnliche 5-mm-LED bei einem kleinen Projekt angeschlossen hat, dann ist man infiziert. Ich würde wetten, dass auch Sie schon einmal mit dem Gedanken gespielt haben, zig oder gar hunderte an LEDs zu einem großen Display zu kombinieren, oder?
Eine Grenze ist durch die Anzahl an Pins eines Mikrocontrollers gegeben und die andere liegt im maximalen Strom, den diese Pins vertragen. Ein Punkt-Matrix-Display ist ja eine Anordnung, wo viele LEDs mit gemeinsamer Anode oder Kathode zusammengefasst betrieben werden, was das mechanische Handling vereinfacht. Manchmal ist in solchen fertigen LED-Modulen schon der passende Treiber-Chip integriert. Für dieses Projekt werden die Teile wie in der „Schaltung“ von **Bild 2** verkabelt.

Bei mir liegen einige Displays des Herstellers Sure herum. Ich habe sie mir letztes Jahr im Arduino-Store für einige Experimente im Unterricht besorgt. Das gleiche Display müsste auch von anderen Lieferanten zu bekommen sein. Im Internet wird das Display gut bewertet und von daher verwende ich es auch.
Das Display beherbergt 16x32 R/G-LEDs. Es ist also mit 512 Zweifarb-LEDs bestückt. Die Punkte können also rot, grün oder (als Mischfarbe) orange leuchten. Man kann sie leicht kaskadieren und ich habe von Projekten gehört, wo insgesamt vier Displays eingesetzt wurden, was beeindruckende 2048 LEDs ergibt!
Für dieses Projekt genügt aber ein Display vollauf, da es nicht viele Informationen anzeigen

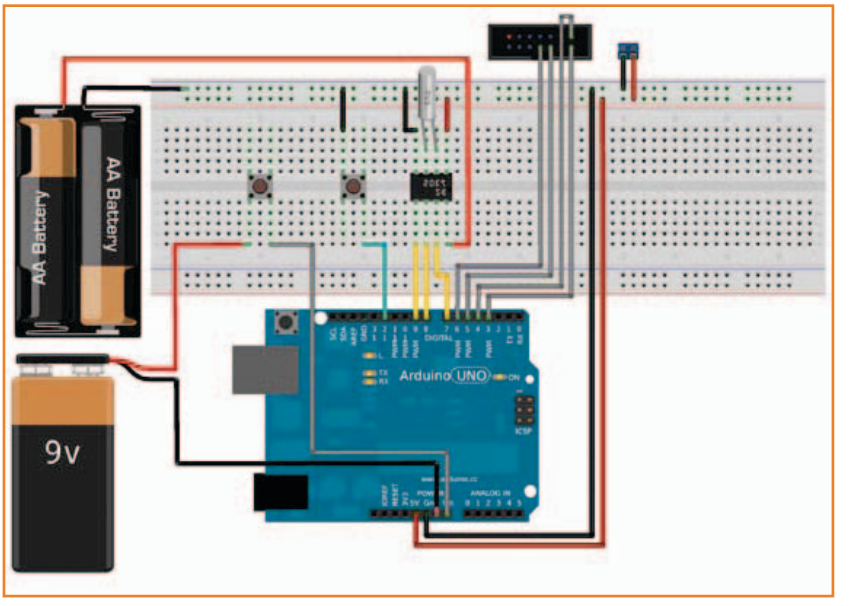


Bild 2.
„Schaltung“ des Gesamtprojekts.

muss. Das Display kann aber mehr, wie man noch sehen wird.

Die Display-Library

Für diese Library habe ich Code von mehreren Autoren recycelt und so kombiniert, dass er den Konventionen einer Arduino-Library entspricht. Außerdem habe ich ein paar Beispiele hinzugefügt. Die Library ist mit den Versionen 1.0 und neuer der IDE kompatibel. Zusätzlich wurde auch noch eine RTC beim Arduino-Prototyping-Shield implementiert (siehe **Bild 3**).
Die Library kann man von der Elektor-Webseite zu diesem Artikel [4] herunterladen und dann installieren. Die Installation einer neuen Library zur IDE erfolgt, in dem im Sketchbook ein Verzeichnis „libraries“ erstellt wird, in das hinein die neue Library entpackt wird. Anschließend startet

Listing 1.

```
#include <fonts.h>
#include <HT1632c.h>
#include <images.h>

HT1632c display(6, 5, 3, 4);

void setup() {
  display.setup();
}

void loop() {
  display.text(“Hej”, 5, 5);
}
```

Listing 2.

```
#include <fonts.h>
#include <HT1632c.h>
#include <images.h>

HT1632c display(6, 5, 3, 4);

void setup() {
  display.setup();
}

void loop() {
  display.image(Arduino_logo, 0, 0, 32, 16);
}
```

Listing 3.

```
#include <fonts.h>
#include <HT1632c.h>
#include <images.h>

#define heart_icon_width 17
#define heart_icon_height 16

unsigned char PROGMEM heart_icon[] = {
   0x00, 0x1e, 0x00, 0x3f, 0x80, 0x7f, 0xc0, 0x7f, 0xe0, 0x7f, 0xf0, 0x7f,
   0xf8, 0x3f, 0xfc, 0x1f, 0xfe, 0x0f, 0xfc, 0x1f, 0xf8, 0x3f, 0xf0, 0x7f,
   0xe0, 0x7f, 0xc0, 0x7f, 0x80, 0x7f, 0x00, 0x3f, 0x00, 0x1e };

HT1632c display(3, 4, 5, 6);

void setup() {
  display.setup();
}

void loop() {
  display.image(heart_icon, 0, 0, heart_icon_width, heart_icon_height);
}
```

man die Arduino-IDE neu und die Library zeigt sich im entsprechenden Menü.
Hinweis: Die Arduino-IDE wird bald Libraries automatisch installieren können, die aus dem Internet geladen wurden. Im Moment aber ist dieses Feature noch nicht eingebaut.

Mit der Library installiert man einige Beispiele zum:

- Testen ob das Display gut funktioniert,
- Anzeigen einfacher Text-Meldungen,
- Scrollen von Text auf dem Display und
- Laden von Bildern.

Zu den Beispielen findet man durch den Pfad: *File / Examples / HT1632c*.
Wenn man die Library benutzen will, muss man die drei Header-Dateien `fonts.h, HT1632c.h` und `images.h`. im Code angeben. Beim Aufruf von `constructor` muss man die Pins benennen, an die das Display angeschlossen ist. Dabei gilt die Pin-Reihenfolge: Data, Write Clock, Chip Select und Clock.

Display-Konfiguration

In diesem Projekt ist das Display an die digitalen Pins 3, 4, 5 und 6 angeschlossen. Die Library lässt die Konfiguration beliebiger Pins für das Display zu. Wie schon erwähnt kann man Displays kaskadieren. Ich selbst hatte nur zwei Exemplare zur Verfügung, doch bin ich sicher, dass es auch mit mehr als zweien funktioniert.
Das Display wird direkt vom Arduino-Spannungsregler versorgt. Bei meinen Tests schien das System nie mehr Strom zu ziehen, als die Batterie hätte liefern können. Doch wenn mehr Displays bei einem Projekt eingesetzt werden, sollte man deren Strombedarf im Auge behalten.
In **Listing 1** wird das Display nach dem Beispiel „simple_text“ angesteuert, das bei der Library dabei ist.
Als Standard nutzt die Text-Methode orange leuchtende LEDs zur Anzeige. Der vierte Parameter bestimmt die Farbe. Hierzu können die Konstanten `BLACK`, `GREEN`, `RED` und `ORANGE` verwendet werden.

Bilder anzeigen

Das Display ist dazu in der Lage, niedrig aufgelöste Bilder darzustellen. Das Bild muss dabei im Programmspeicher als Array abgelegt sein. In **Listing 2** ist das Beispiel „simple_image“ implementiert, es wird ein Standard-Icon geladen. **Bild 4** zeigt wie es aussieht, wenn man so eine einfache Grafik auf dem Display ausgibt.
Wenn man ein eigenes Bild anzeigen möchte, kann man mit der freien Software Gimp code-freundliche Bilder generieren. Die Vorgehensweise:

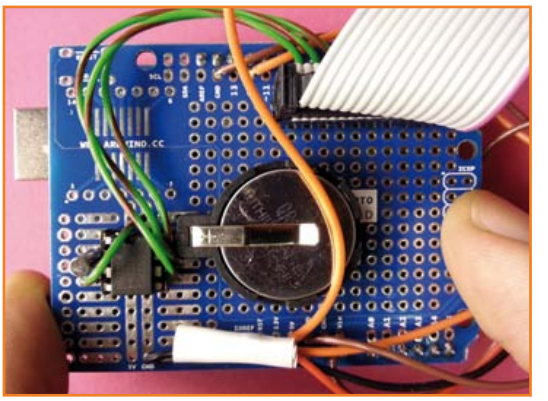


Bild 3.
Ein Arduino-Prototyping-Shield mit aufgelöteten Bauteilen.

Bild 4.
Wow! Unser „Bildschirm“ zeigt das Arduino-Logo.

- Man öffne das Bild und skaliere es, damit es auf das Display mit seinen 16 Reihen und 32 Spalten passt.
- Nun erfolgt eine Rechtsdrehung des Bildes um 90°.
- Anschließend wird es als XbitMap exportiert.
- Nachdem die xbm-Datei geöffnet ist, wird das resultierende Array in die Zwischenablage kopiert.
- Schließlich wird das Array in das Programm als `unsigned char PROGMEM` eingesetzt. Auf diese Weise landet das Bild im Programmspeicher und das RAM bleibt unbehelligt.

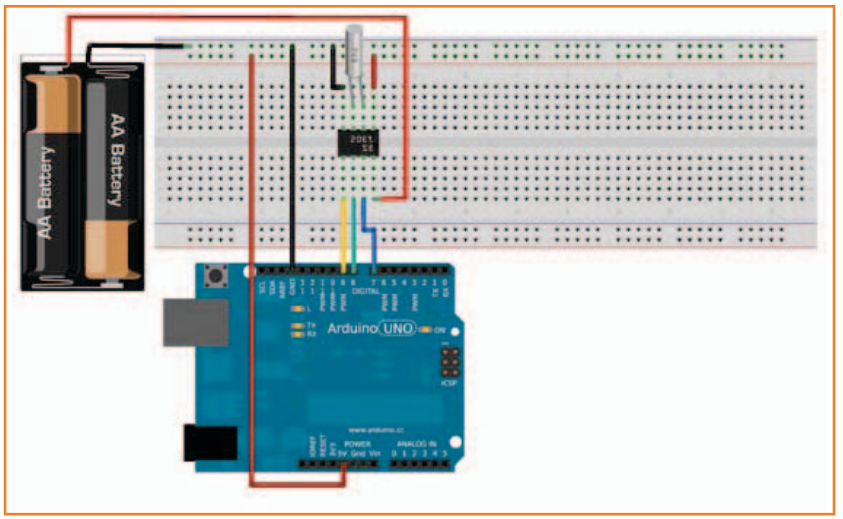
Bild 5 zeigt, wie die Darstellung letztlich aussieht. Es handelt sich um eine herzförmige Clipart-Grafik. **Listing 3** zeigt den Code.

Bild 5.
Herz-Clipart, bevor es für die Integration in den Code bearbeitet wurde.

Hinweis: Das Bild wird auf dem Display etwas anders angezeigt als im Original. Wenn man ein eigens Icon verwenden will, sollte man Breite und Höhe vertauschen. In meinem Fall hatte das Icon ein Format von 16x17, doch dies muss als 17x16 (Breite x Höhe) angegeben werden.
Tipp: Man kann mehrere Icons oder Bilder in den Programm-Code einfügen. Man muss ihnen lediglich unterschiedliche Bezeichnungen geben, damit man sie von einer anderen Stelle des Codes aufrufen kann.

Die Library eignet sich für die unterschiedlichsten Darstellungen auf dem Display. Für dieses Projekt genügt es allerdings, einfachen Text und vielleicht noch einfache Bilder auszugeben. Es lohnt sich aber, mit den Möglichkeiten der Library zu experimentieren. Übrigens: Sie ist freie Software!

Bild 6.
Schaltung der an Arduino angeschlossenen RTC.



RTC-Chips

Dank des 32-kHz-Uhrenquarzes kann eine RTC die Zeit sehr genau messen. Eine RTC muss die Zeit immer erfassen und kann daher nicht zwischendurch ausgeschaltet werden. Deshalb braucht eine RTC eine Versorgung per Batterie. Da so ein Chip aber sehr wenig Strom benötigt, läuft er mit einer 3-V-Knopfzelle oft jahrelang.
Für dieses Projekt habe ich das gebräuchliche IC DS1302 von Maxim ausgewählt, weil es gerne in der Arduino-Community verwendet wird. Dazu gibt es gute Infos von User Krodal im Arduino Playground [2], wo sich auch Code findet, mit dem man die Fähigkeiten des Chips ausloten kann. Wegen der Einfachheit habe ich mich für die Library von Matt Sparks [3] entschieden.
Bis jetzt sind ja erst vier Pins von Arduino Uno zum Anschluss des Displays belegt, das RTC-IC benötigt drei weitere: die Pins 7, 8 und 9. **Bild 6** zeigt die „Schaltung" mit dem RTC - genauer den Anschluss von Versorgung und Quarz und die Verbindung mit Arduino.

Installation der RTC-Library

Die RTC-Library findet sich ebenfalls auf der Elektor-Webseite [4] oder beim Github-Projekt des Autors [3]. Sie wird wie schon beschrieben durch Entpacken in das Verzeichnis „libraries" installiert. Die Beispiele befinden sich dann ebenfalls dort. Zur Zeiteinstellung muss einmal `set_clock.pde` ausgeführt werden. Solange die RTC mit der (vollen) Batterie verbunden ist, muss dieser Code kein zweites Mal ausgeführt werden. Vor der Ausführung sollte natürlich innerhalb des Codes die Zeile mit der einzustellenden Zeit angepasst werden. **Listing 4** zeigt den Teil von `set_clock.pde`, den man anpassen muss.

Aktuelle Zeit ins EEPROM

Der ATmega328 von Arduino Uno hat 512 Byte EEPROM. Dieser nichtflüchtige Speicher behält seine Information auch dann, wenn der Strom ausgeschaltet ist. Der Speicher wird bei Embedded-Projekten gerne zur Ablage der grundlegenden Konfiguration verwendet.

In unserem Fall wird lediglich das Datum im EEPROM abgelegt, an dem zum letzten Mal gegossen wurde (und nicht vergessen wurde, den Knopf zu drücken). Dazu sind dann drei Bytes notwendig: Jahr, Monat und Tag. Unser Programm muss nun noch die Fähigkeit haben, die Differenz zwischen der gespeicherten und der aktuellen Zeit

Listing 4.

```
[…]

void setup() {
  Serial.begin(9600);

  /* Initialize a new chip by turning off write protection and clearing the
     clock halt flag. These methods needn't always be called. See the DS1302
     datasheet for details. */
  rtc.write_protect(false);
  rtc.halt(false);

  /* Make a new time object to set the date and time */
  /*   Tuesday, May 19, 2009 at 21:16:37.            */
  Time t(2009, 5, 19, 21, 16, 37, 3);

  /* Set the time and date on the chip */
  rtc.time(t);
}

[…]
```

Ein Projekt aus Bauteilen, die keiner mehr beachtet hatte…

zu berechnen, die dann angezeigt wird. **Listing 5** zeigt, wie man Daten im EEPROM speichert.

Knopf speichert Zeit

Der schon erwähnte Taster liefert ein Triggersignal, das anzeigt, dass die Zeit aufgezeichnet werden soll. Bei einem Tastendruck muss dann die aktuelle Zeit aus der RTC gelesen und im EEPROM gesichert werden. Der Taster ist im Code an Pin 12 angeschlossen (siehe **Listing 6**). Die Konfiguration des Pins als INPUT_PULLUP spart immerhin einen Widerstand ein.

Nach dem Speichern der Zeit wird auf dem Display die Meldung „nice!“ erscheinen und das war's auch schon. Die weitere Code-Ausführung ist dann blockiert, bis das Programm neu gestartet wird. Diese Maßnahme verhindert, dass das EEPROM durch Fehlbedienung zu oft beschrieben wird.
Schauen Sie sich ruhig den ganzen Code in der Download-Datei von Elektor an. Dort ist noch eine besondere Animation enthalten, mit der ich meinen Nachbarn fürs Gießen danken will!

Ein Knopf für alles

Ein wichtiger Vorteil des Designs ist der Stromver-

Listing 5.

```
#include <EEPROM.h>

int val;

void setup() {
}

void loop() {
  val = analogRead(A0);
  EEPROM.write(0, val);
}
```

Bild 7.
„Hallo, es ist schon drei Tage her, seit mir jemand etwas Wasser spendierte!“

brauch. Meiner Ansicht nach ist die beste Methode zur Schonung der Batterie, wenn das System die meiste Zeit schlicht aus ist. Dass da noch ein zweiter Taster in der Stückliste aufgeführt wird, das ist Ihnen bestimmt nicht entgangen. Bislang wurde nur einer davon benutzt.

Listing 6.

```
[…]

int saveTimePin = 12;
int saveTimeButton = LOW, saveTimeButtonOld = LOW;

void setup() {
  […]
  pinMode(saveTimePin, INPUT_PULLUP);
}

void loop() {
  […]
  saveTimeButton = digitalRead(saveTimePin);
  if(saveTimeButton == LOW && saveTimeButtonOld == HIGH) {
    // we will use the Time stored in t to save the time
    EEPROM.write(0, t.yr);   // position 0 – year
    EEPROM.write(1, t.mon);  // position 1 – month
    EEPROM.write(2, t.date); // position 2 – day
    // clear the display
    display.cls();
    // show a thank you text
    display.text(“nice!”, 2, 5);
    // stay here
    while(true) {};
  }
  saveTimeButtonOld = saveTimeButton;
}
```

Der zweite Taster ist in die Stromversorgung eingeschleift. Ohne Tastendruck kein Strom. Ein Mikrocontroller kann sehr schnell aufwachen. Wenn er aufwacht, dann sieht man zuerst das Arduino-Logo und nach wenigen Sekunden wird die Anzahl an Tagen angezeigt, die seit der letzten Gießaktion vergangen sind.
Ein so geschalteter Taster erfordert dann von meinen Nachbarn, dass sie zum Abspeichern der aktuellen Zeit (nachdem die Pflanzen getränkt wurden), beide Taster zugleich betätigen müssen. Ein Knopf schaltet das System ein und der andere speichert das aktuelle Datum der RTC im EEPROM ab (siehe **Bild 7**).

Schlusswort

Wenn man „bloß mal etwas“ mit „ein paar Bauteilen“ auf die Beine stellen will, führt einem solch ein Vorsatz leicht zu Libraries, mit denen man gleich hunderte LEDs auf einmal ansteuern kann. Der nächste Schritt wäre dann aber zu überprüfen, ob der Prototyp seinen Zweck auch wirklich erfüllt. Ich habe meinen Prototypen ins Treppenhaus gehängt. Die Zeit bzw. die Farbe der Blätter wird zeigen, ob „unsere“ Pflanzen nun genug gegossen werden...

Ich bin mir sicher, dass Ihnen viele andere Anwendungen für eine Kiste mit großem Display und zwei Tastern einfallen. Man müsste ja nur einen kleinen Buzzer und noch ein paar Taster hinzufügen und schon hätte man eine eigene Spielkonsole…

(120714)

Weblinks

[1] Das Botanicalls-Projekt:
http://botanicalls.com

[2] Erläuterungen zum DS1302 von Krodal:
http://arduino.cc/playground/Main/DS1302

[3] RTC-Library von Mark Sparks:
http://github.com/msparks/arduino-ds1302

[4] Download von Programm und Libraries:
www.elektor-magazine.de/120714

Dank

Beim Arduino-Store möchte ich mich ganz herzlich dafür bedanken, dass mir das Display für dieses Projekt überlassen wurde.

elektor ACADEMY
the school of electronics

Eagle PCB und Design (1-tägiges Seminar)

In diesem Kurs werden Sie lernen, wie man mit dem Programm Eagle der Firma Cadsoft GmbH Leiterplatten entflechten kann. Begonnen wird mit dem Zeichnen von Schaltplänen unter Verwendung von Standard-Eagle-Bibliotheken. Sie lernen, wie man Schaltpläne über mehrere Seiten hinweg zeichnet und wie man eigene Bibliotheken und Bauteile erstellt. Nach erfolgreichem Layout werden Produktionsdaten erzeugt, die man benötigt, wenn man die Platine fertigen lassen möchte. Dabei wird auch auf die verschiedenen Produktionsarten wie fräsen und ätzen eingegangen. Zum Abschluss gibt es Tipps und Tricks zum Umgang mit EAGLE. Zur Vertiefung des Stoffes werden Sie mit der Light-Version von Eagle Übungen an Ihrem eigenen Notebook durchführen

Referent: Prof. Dr.-Ing. Francesco P. Volpe - Teilnahmegebühr: € 449,00

PIC-Mikrocontroller-Programmierung in C (2-tägiges Seminar)

In diesem 3-tägigen Kurs werden Sie die Programmierung von „Eingebetteten Systemen" in der Programmiersprache C kennenlernen. Dazu wird auf die PIC-Mikrocontroller-Familie der Firma Microchip anhand des C18-/C30-Compilers eingegangen. Das Hauptaugenmerk liegt auf der C-Sprache, um sich besser auf die verschiedenen Elemente von C selbst zu konzentrieren. Die Darstellung wird von einer Reihe praktische Übungen, die Sie selbst an Ihrem Notebook durchführen werden, begleitet. Dabei werden Übungen innerhalb des MPLAB-Simulators ausgeführt und ermöglichen Ihnen, die Vertiefung des Stoffes. Ferner ist es möglich, das Erlernte mit jedem ANSI C-Compiler anzuwenden. Abgeschlossen werden die Betrachtungen mit praktischen Übungen, die Sie selber mit einem PICkit 3 Debug Express und zugehöriger Platine durchführen werden.
Das Hardware-Tool im Wert von ca. 69,00 € nehmen Sie im Anschluss mit nach Hause.

Referent: Prof. Dr.-Ing. Francesco P. Volpe – Teilnahmegebühr: 899,00 € (inkl. MwSt.)

NEU

Arduino - Programmierung und Projektentwicklung

Hier setzt das Seminar "Arduino – Programmierung und Projektentwicklung" an. Nach einer kurzen Einführung und der Inbetriebnahme des Arduino-Boards erfolgt eine systematische Einführung in verschiedene Themengebiete.
Dabei wird neben den erforderlichen theoretischen Grundlagen stets größter Wert auf eine praxisorientierte Ausrichtung gelegt. So werden wichtige Techniken wie AD-Wandlung, Timer oder Interrupts anhand von Praxisprojekten ausführlich erläutert.
Den Abschluss des Seminars bildet eine Einführung in die eigenständige Entwicklung von Projekten und Systemen. Der Seminarteilnehmer wird damit in die Lage versetzt, auch komplexe eigene Ideen in praxistaugliche Geräte umzusetzen. Ganz nebenbei hat der Kursteilnehmer dann auch die „Basics" der zugehörigen Controllertechnik verstanden und im wahrsten Sinne des Wortes begriffen

Referent: Dr. Günter Spanner - Teilnahmegebühr: € 349,00

elektor ACADEMY
the school of electronics

Workshops * Seminare * Kurse * Weiterbildungen

Top-Fachleute aus der Branche referieren über ein faszinierendes Thema!

SEHR GUT
9 von 10 Seminaren werden von unseren Teilnehmern mit sehr gut bewertet.

Embedded Linux In Theorie und Praxis

15. bis 17.01.2013	Zürich (CH)
15. bis 17.04.2013	München
16. bis 18.09.2013	Hanau

Multi Core / Parallel Programming

10. bis 12.09.2013	München

Eagle PCB und Design

21.02.2013	Dortmund

Linux Debugging

18. + 19.04.2013	München
06. + 07.06.2013	Hanau
19. + 20.09.2013	Dortmund

"Arduino – Programmierung und Projektentwicklung"

09.03.2013	München
16.04.2013	Hanau
05.09.2013	Dortmund
07.11.2013	Zürich (CH)

PIC Microcontroller Programmierung

05. + 06.03.2013	Hanau
16. + 17.04.2013	München

Echtzeitbetriebssysteme in Theorie und Praxis

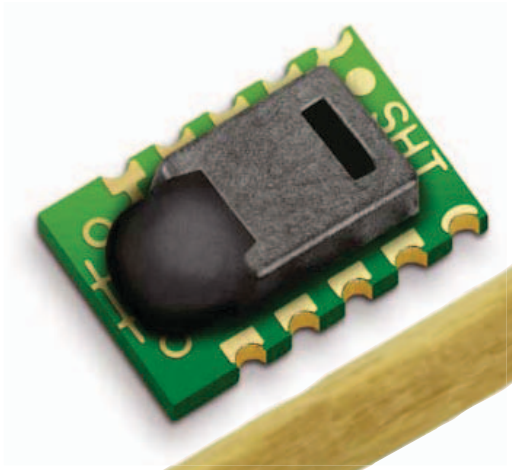
03. bis 05.06.2013	Hanau

INKLUSIVE elektor ZERTIFIKAT

Weitere Infos & Anmeldung: www.elektor.de/events

SHT11:
Luftfeuchtesensor am PC
Luftfeuchtigkeit langfristig überwachen

Von Pavel Setnicar
(Slowenien)



Der Sensor SHT11 des Schweizer Herstellers Sensirion misst neben der Temperatur auch die Luftfeuchtigkeit und ist voll digital. In diesem Beitrag geht es darum, wie man damit einen Daten-Logger aufbaut, der Luftfeuchtigkeit und Temperatur längerfristig mit einem PC erfasst.

Auch wenn die Lektüre des Datenblatts zum SHT11 [1] beileibe keine Zeitverschwendung ist, kann man doch mit wenig Wissen loslegen: Der Ausgang ist digital, die Genauigkeit liegt bei ±3 % RL (Relative Luftfeuchtigkeit) und der Messbereich umfasst 0...100 % RL. Temperaturmessungen können im Bereich –40...+124 °C (–40...+255 °F) mit einer Genauigkeit von ±0,4 % getätigt werden.

Der SHT11-Ausgang ist vom I²C-Typ – aber bei näherer Betrachtung gibt es doch Besonderheiten beim verwendeten Protokoll. Die Kommunikation läuft über die beiden Pins SCK und DATA. Der Takt an SCK dient zur Synchronisierung mit anderen Geräten und der tri-state-fähige DATA-Pin transferiert Daten in beide Richtungen.

Um Daten vom Sensor zu erhalten, benötigt man bestimmte Kommandos, die das positive Schweizer Klischee bestätigend sehr ordentlich im Datenblatt beschrieben sind.

Mein Ziel war es, diesen Sensor an meinen PC anzuschließen und (entsprechend programmiert) Messungen in bestimmten Intervallen durchzuführen, die Daten anzuzeigen und bei Bedarf als Text-Datei für die spätere Bearbeitung abzuspeichern. Aufgrund der Einfachheit wollte ich den Sensor an eine serielle Schnittstelle des PCs anschließen. Wenn man einen modernen „legacy free“ PC hat, kann man für diesen Zweck einen preiswerten USB/Seriell-Wandler einsetzen. Das Minimal-Interface von **Bild 1** zeigt, welche Leitungen der seriellen Schnittstelle wie belegt sind:

- DTR (Pin 4) schickt Daten über einen BC557 an den DATA-Pin des Sensors;
- RTS (Pin 7) liefert den Takt an den SHT11-Pin SCK;
- CTS (Pin 8) liest die Daten vom Data-Pin des Sensors.

Die Versorgung für den Sensor wird von den Pins der seriellen Schnittstelle abgezweigt. Das ist zwar nicht im Sinne des Erfinders, es wird aber dennoch häufig gemacht und ist hier kein Problem, da ein SHT11 im aktiven Zustand kaum 0,5 mA benötigt. Da der Sensor die meiste Zeit sowieso im Idle-Mode verharrt, ist der mittlere Stromverbrauch zu vernachlässigen. Die beiden Dioden D1 und D2 sind als Stromdiebe geschaltet und laden Kondensator C1. Je nach Fabrikat liefert eine serielle Schnittstelle hier gut 10 V, sodass mit der Z-Diode D4 die Spannung passend begrenzt und stabilisiert wird.

Die hohen RS232-Pegel von typisch ±10 V machen es zudem notwendig, das RTS-Signal mit D3 ebenfalls 5-V-kompatibel zu machen.

Mit Hilfe von Bit-Shifting (beide Richtungen, gut im Datenblatt beschrieben) erhält man die beiden

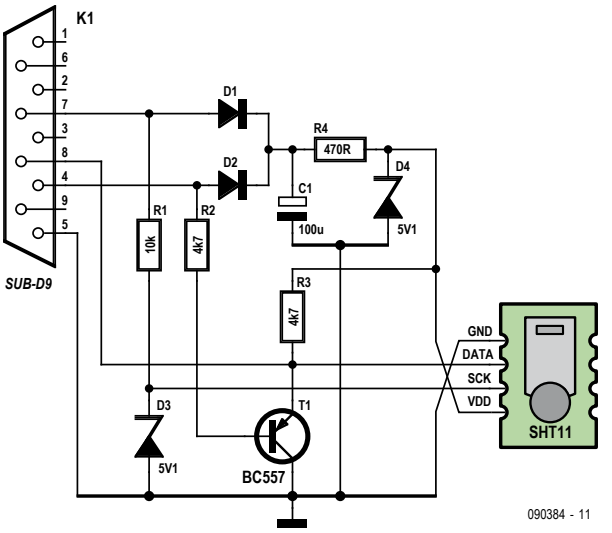


Bild 1.
Das ist schon alles an Elektronik, was man für den Anschluss des SHT11 via serielle Schnittstelle an einen PC benötigt.

Rohdatenblöcke für Temperatur und Feuchtigkeit. Bei meinen Experimenten stellte ich einen leichten Offset bei der Temperatur fest, weshalb ich die Software mit einer Option zur Kalibrierung der Temperatur erweiterte. Wenn die vom Sensor gelieferten Werte z.B. um 1,2 °C zu hoch sein sollten, gibt man im Eingabefeld Temp.offset einfach den Wert „-1.2“ ein.

Die Feuchtigkeitswerte des Sensors müssen zunächst noch mathematisch linearisiert werden, was die Software ebenfalls übernimmt. Für Pedanten gibt es je eine Anzeige mit den rohen und eine mit linearisierten Werten. Die Unterschiede sind klein, da auch die Nichtlinearität des Sensors klein ist. Im unteren Teil kann man das Sample-Intervall in Sekunden eingeben.

In **Bild 2** sieht man die eintreffenden Daten und kann im Fenster durch die Zeilen scrollen. Man kann die gesammelten Daten jederzeit als Datei sichern und später z.B. mit Excel weiter verarbeiten.

Die Software für den PC wurde in C# mit Microsoft Visual Studio 2008 geschrieben. Für die Installation muss zuvor das .NET-Framework ab Version 3.5 installiert sein. Die unter [2] erhältliche Archiv-Datei enthält sowohl den Installer des Programms als auch den Source-Code. Das PC-Programm wurde mit verschiedenen Rechnern unter Windows XP getestet. Meines Wissens läuft es auch gut mit verschiedenen USB/Seriell-Konvertern - bei mir jedenfalls funktionierte das Modell STLab-4 prima. Die Schaltung arbeitete bei mir sogar noch mit einem 5 m langen seriellen Kabel. Wie lang das Kabel bei Ihrem PC sein darf, können Sie nur durch Ausprobieren herausfinden.

(090384)

Weblinks

[1] SHT1x-Datenblatt: www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf

[2] www.elektor-magazine.de/090384

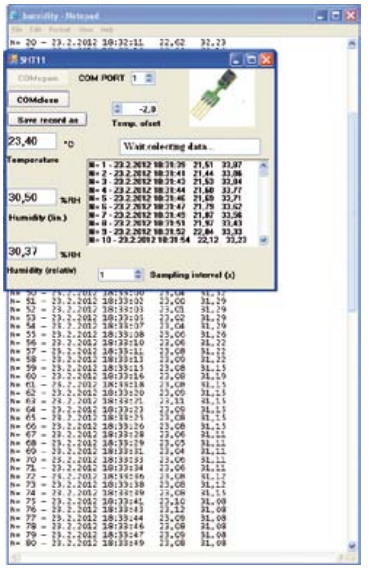
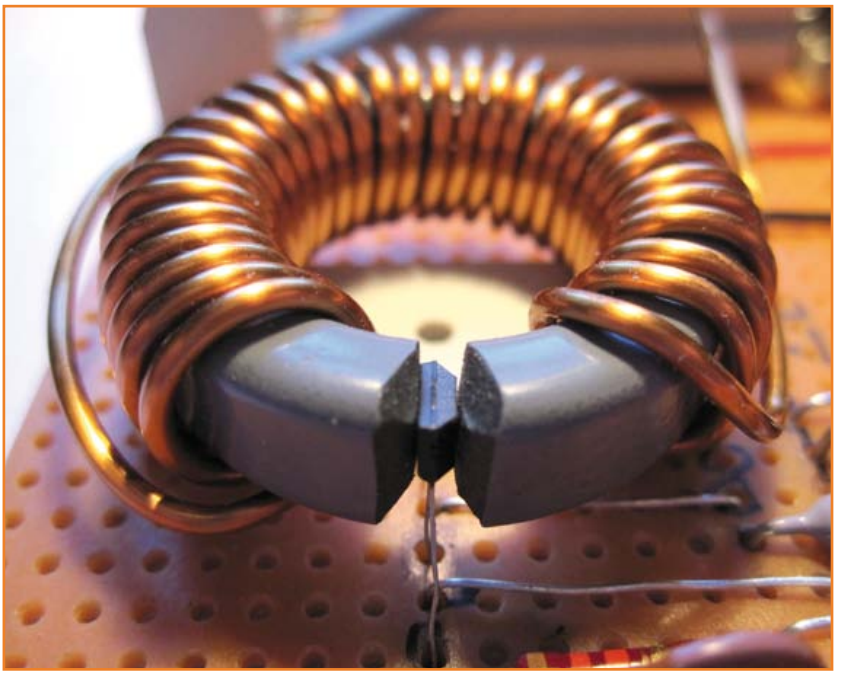


Bild 2.
Das Programm in Aktion: Die Daten laufen ein und können jederzeit für eine spätere Bearbeitung in einer Datei gesichert werden.

Einschaltstrom-Begrenzer

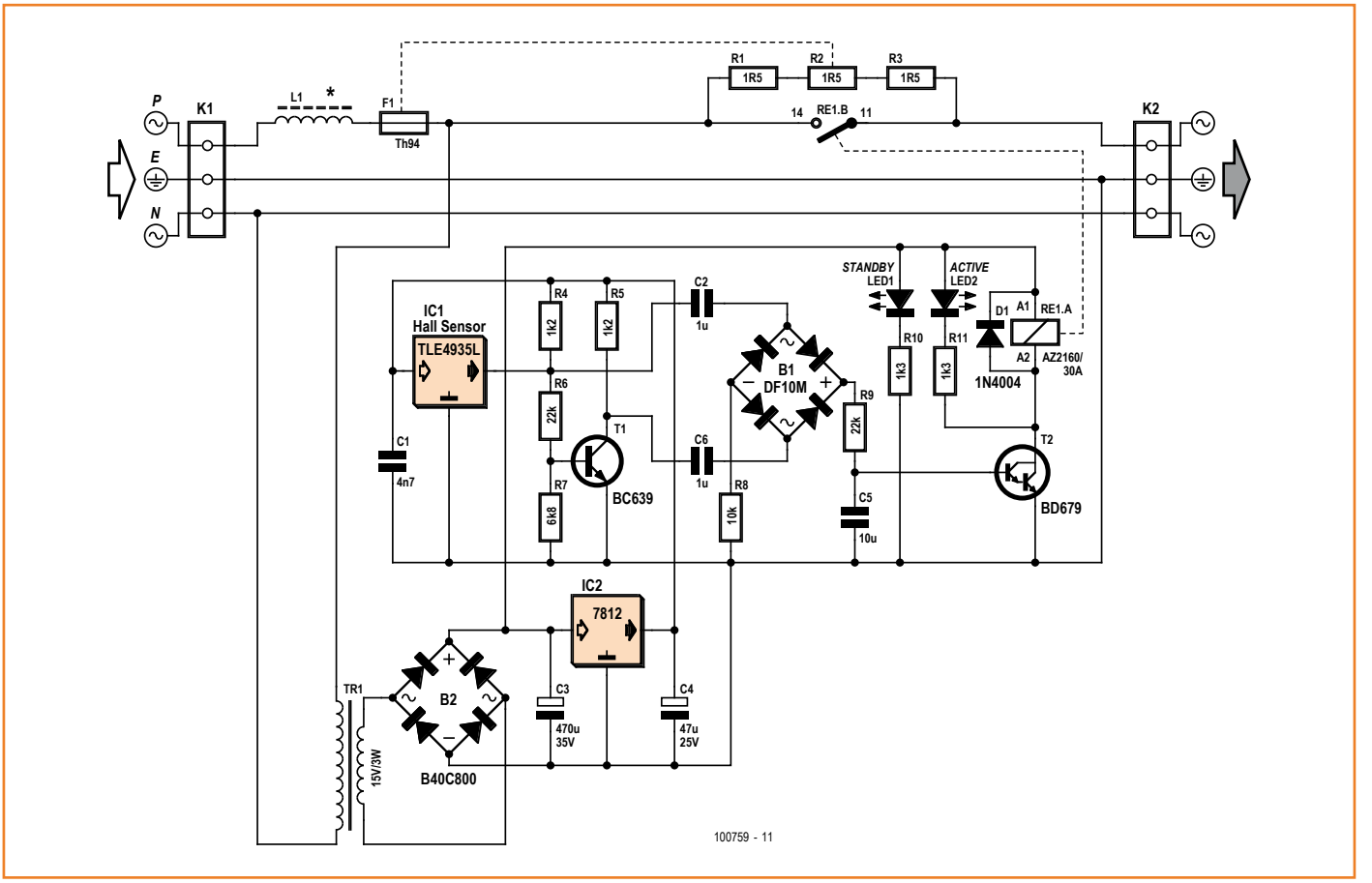
Von
Wilfried Wätzig (D)



Beim Einschalten bestimmter Verbraucher (beispielsweise mit großen Transformatoren oder Motoren) kann es physikalisch bedingt zu Stromspitzen kommen. Diese können dann eine vorgeschaltete Sicherung oder den Motorschutzschalter auslösen und so verhindern, dass das Gerät ordnungsgemäß arbeitet. Ein Einschaltstrom-Begrenzer löst dieses Problem, indem ein Hochlastwiderstand die entstehenden Stromspitzen aufnimmt. Nachdem der Einschaltvorgang beendet ist, wird dieser Widerstand überbrückt und dem Verbraucher die volle Netzspannung zur Verfügung gestellt.

Die hier vorgestellte Schaltung realisiert einen stromgesteuerten Einschaltstrom-Begrenzer, der direkt an der Netzspannung angeschlossen ist und mit Hilfe einer Spule und einem Hall-Sensor den zum Verbraucher fließenden Strom überwacht. Die Herstellung der Spule erfordert ein wenig Fingerspitzengefühl: Grundbaustein ist ein Ferritring, z.B. ein FT 114-74 mit 30 mm Außendurchmesser, 20 mm Innendurchmesser und einer Höhe von 6 mm. Zunächst wird ein Luftspalt von ca. 1,5 mm Breite in den Ferritring geschnitten, was etwas knifflig ist; deshalb zeigen wir in unserer •Labs-Rubrik in dieser Ausgabe eine kurze Anleitung. Der Spalt muss so breit sein, dass ein Hall-Sensor vom Typ TLE4935L genau dort hineinpasst (siehe Foto). Auf den Ring wird dann mit einem 1,2 mm starken Kupferlackdraht eine Spule aus 30 Windungen gewickelt. Der durch die Spule fließende Strom erzeugt im Luftspalt des Ferritrings ein Magnetfeld, das von dem dort positionierten Hall-Sensor erfasst wird. Es handelt sich dabei um eine digitale Variante, mit einem Schmitt-Trigger und einem Open-Collector-Ausgang. Wenn das Magnetfeld eine bestimmte Stärke überschreitet, schaltet der Sensor den Ausgang auf Masse „durch". In der hier beschriebenen Dimensionierung passiert dies, wenn der Strom durch die Spule größer als 1 A wird. Da die aktive Fläche des Hall-Sensors mit ca. 1 mm² viel kleiner als der ca. 30 mm² große Querschnitt des Ferritrings ist, wird bei den hier vorgeschlagenen Bauteilen nur ein Bruchteil des erzeugten Magnetfeldes genutzt. Ein Ferritring mit geringerem Querschnitt wäre also noch besser geeignet.

Die Schaltung ist übersichtlich: Über K1 ist der Einschaltstrom-Begrenzer mit dem Niederspannungsnetz (230 V) verbunden. Der Verbraucher, dessen Einschaltstrom begrenzt werden soll, ist am Ausgang der Schaltung (K2) angeschlossen. Der Transformator TR1 bildet zusammen mit dem Brückengleichrichter B2 und dem Spannungsregler IC2 das Netzteil für die Steuerschaltung und versorgt diese mit stabilen 12 V.



L1 misst zusammen mit dem Hall-Sensor IC1 den Strom, der zum Verbraucher am Ausgang der Schaltung fließt. Wenn das Leistungsrelais RE1 den Leistungswiderstand aus R1, R2 und R3 nicht überbrückt, nimmt dieser die Stromspitzen auf. Solange der Strom durch die Spule einen gewissen Wert nicht überschreitet, wird die Spannung am Ausgang des Hall-Sensors durch den Spannungsteiler aus R4, R6 und R7 auf einen Wert nahe bei 12 V gelegt. Die Spannung an der Basis von T1 ist dann hoch genug, dass der Transistor leitet. Beim Einschalten des Verbrauchers (und einem hohen Strom durch die Spule) schaltet der Hall-Sensor durch, so dass T1 nun sperrt. Wegen der Netzwechselspannung entsteht am Ausgang des Hall-Sensor ein 50-Hz-Rechtecksignal, das durch T1 invertiert wird. Über den nachgeschalteten Gleichrichter B1 wird C5 nun geladen. In der hier gezeigten Dimensionierung von R9 und C5 beträgt die Verzögerung rund 60 ms, bis der Darlington-Transistor T2 leitet und das Leistungsrelais K1 schließt. In dieser Zeit nimmt der Leistungswiderstand (R1, R2 und R3) die anfallenden Stromspitzen auf.

Der Schutzwiderstand ist in drei Lastwiderstände zu je 1,5 Ω aufgeteilt. Da bei einem Stromfluss von beispielsweise 20 A in jedem der Widerstände kurzzeitig 600 W Leistung umgesetzt werden, ist zusätzlich die Thermosicherung F1 vorgesehen, die in gutem thermischen Kontakt mit mindestens einem der Lastwiderstände stehen sollte. Falls die Steuerschaltung einmal ausfällt, ist auf diese Weise sichergestellt, dass das gesamte Gerät stromlos wird. Dazu noch ein Tipp: Thermosicherungen sollten niemals verlötet, sondern immer festgeklemmt werden.

Achtung, Lebensgefahr: Die Schaltung muss berührungsgeschützt und entsprechend den VDE-Bestimmungen (VDE 0100, VDE 0701) in ein Gehäuse eingebaut werden! Die Leitungen, die Netzspannung führen, müssen ausreichenden Abstand zur Steuerschaltung aufweisen!

(100759)

CAN mit Bascom-AVR

Von **Mark Alberts** (NL)

Bascom-AVR [1], der bekannte Basic-Compiler für AVR-Mikrocontroller unter Windows, unterstützt von Haus aus zahlreiche periphere Komponenten. Eine Ausnahme machte bisher der CAN-Bus (Controller Area Network). Ursprünglich war dieser Bus für die System-Kommunikation in Fahrzeugen gedacht, doch inzwischen ist CAN in fast alle Bereiche der Technik vorgedrungen. Was das Protokoll betrifft, existieren zwar mehrere Standards, die grundlegenden Eigenschaften sind jedoch gleich.

Elektor veröffentlichte bereits im April 2009 ein CAN-Controller-Projekt [2], auf dem zugehörigen Board befindet sich ein AT90CAN32 von Atmel. Die CAN-Schnittstellen, die in Mikrocontrollern integriert sind, stellen nur geringe Anforderungen an die Software, außerdem wird der Mikrocontroller für andere Aufgaben entlastet.

Hardware

Das Board vom April 2009 ist auch heute noch topaktuell. Weil beim Testen auf eine serielle Schnittstelle kaum verzichtet werden kann, wird der Einsatz des USB/TTL-Konverter-Kabels empfohlen, das im Elektor-Shop unter 080213-71 bestellbar ist [3]. Das Kabel wird mit TX1 und RX1 auf dem Board verbunden. Gleichzeitig hat das Kabel den Vorteil, dass der USB-Port die Betriebsspannung bereitstellt. Wenn ein MAX232 (als Pegelwandler für eine serielle Schnittstelle) eingesetzt wird, muss ein Netzteil die Betriebsspannung liefern.

Für den Datenaustausch über den CAN-Bus müssen mindestens zwei Boards vorhanden sein. Der CAN-Bus wird realisiert, indem die Boards wie nebenstehend dargestellt zweiadrig über K2 verbunden werden. Zum Programmieren dient ein ISP-Programmer mit 6-poligem Stecker.

Normalerweise ist das Fuse-Byte des Mikrocontrollers für den internen Taktoszillator und den Teilfaktor 8 konfiguriert. Der Betrieb des CAN-Busses setzt aber einen stabilen Takt voraus, auf dem Board ist bereits ein 12-MHz-Quarz vorhanden.

Quellcode CAN-Demo-Programm

```
On Canit Can_int                                    ' define the CAN interrupt
Canreset                                            ' reset can controller
Canclearallmobs                                     ' clear all message objects
Canbaud = 125000                                    ' use 125 KB

Config Canbusmode = Enabled                         ' enabled,standby,listening
Config Canmob = 0 , Bitlen = 11 , Idtag = &H0120 , Idmask = &H0120 , Msgobject = Receive , Msglen = 1 ,
    Autoreply = Disabled                            'first mob  is used for receiving data
Config Canmob = 1 , Bitlen = 11 , Idtag = &H0120 , Msgobject = Disabled , Msglen = 1  ' this mob is used for sending data

Cangie = &B10110000                                 ' CAN GENERAL INTERRUPT and TX and RX
Print #2 , "Start"

Do
  If Pinc <> Bdil Then                              ' if the switch changed
     Bdil = Pinc                                    ' save the value
     Bok = Cansend(1 , Pinc)                        ' send one byte using MOB 1
     Print #2 , Bok                                 ' should be 0 if it was send OK
  End If
Loop

Can_int:
_can_pageok = Canpage                               ' save can page because the main program can access the page too
  Cangetints                                        ' read all the interrupts into variable _can_mobints
```

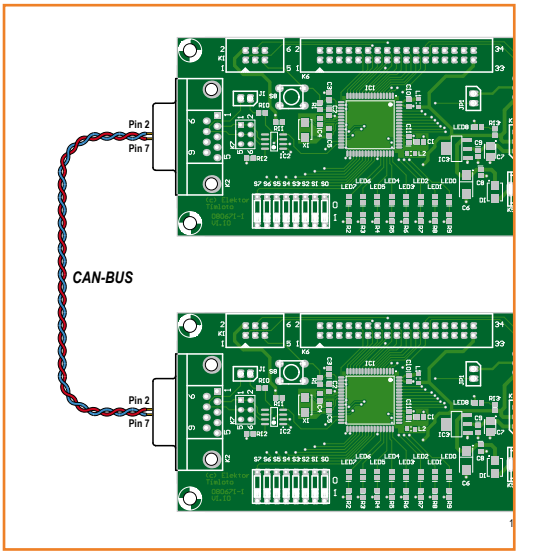
Die Anweisung $PROG in BASCOM muss wie folgt lauten: `$prog &HFF , &HCF , &HD9 , &HFF.`

Software

Für die Programmierung einer CAN-Schnittstelle wurde BASCOM um spezielle CAN-Statements erweitert. Der Programmcode wird mithilfe von Interrupts abgearbeitet, was den Controller entlastet. An dieser Stelle können die CAN-Statements nur in Kurzform beschrieben werden.
Für die CAN-Schnittstelle ist die mit *CANBAUD* einstellbare Busgeschwindigkeit ein wichtiger Parameter. Alle am CAN-Bus angeschlossenen Subsysteme müssen mit gleicher Busgeschwindigkeit arbeiten. In der Praxis verlaufen häufig mehrere CAN-Busse parallel, die für unterschiedliche Geschwindigkeiten ausgelegt sind.

Das CAN-Protokoll arbeitet mit so genannten *Message Objects* (MObs), wobei dem CAN-Controller 15 MObs zugeordnet sind. Die an einen MOb vergebenen Definitionen legen unter anderem fest, ob Daten gesendet oder empfangen werden. Die übertragenen Blöcke können 11 bit oder 29 bit lang sein. Die Definitionen eines MOb sind von den übrigen MObs unabhängig. Ein Beispiel für die Definition eines MObs:

```
Config Canmob = 0 ,
Bitlen = 29 ,
Msgobject = Receive ,
Msglen = 8 ,
Idtag = &H0000 ,
Idmask = &H0000 ,
Autoreply = Disabled
```



Hier wird MOb 0 auf 29 bit für einen Datenblock eingestellt, es werden Datenblöcke empfangen. Die empfangenen Informationen können bis zu acht Datenblöcke lang sein. Wenn die Länge nicht ausreicht, werden die Daten auf mehrere MObs verteilt. Mit IDTAG ist das Filtern der IDs möglich, die empfangen werden sollen, und mit IDMASK sind auch Bereiche definierbar. Die Filter ignorieren alle IDs, die für andere Übertragungsziele bestimmt sind. Für die Ökonomie der Systemressourcen ist wichtig, die Filter sorgfältig einzustellen. Das Unterbrechen des Hauptprozesses durch fremde Informationen ist zu vermeiden. Auch dazu ein Beispiel:

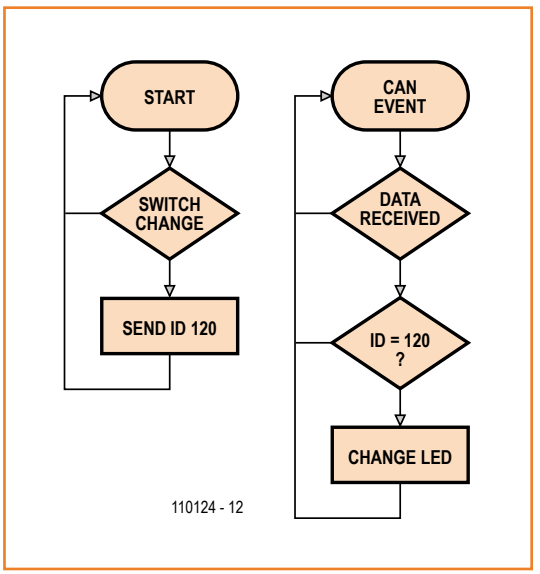
```
For _can_int_idx = 0 To 14                              ' for all message objects
   If _can_mobints._can_int_idx = 1 Then                ' if this message caused an interrupt

      Canselpage _can_int_idx                           ' select message object

      If Canstmob.5 = 1 Then                            ' we received a frame
         _canid = Canid()                               ' read the identifier
         Print #2 , Hex(_canid)

         Breceived = Canreceive(porta)                  ' read the data and store in PORTA
         Print #2 , "Got : " ; Breceived ; " bytes"     ' show what we received
         Print #2 , Hex(porta)
         Config Canmob = -1 , Bitlen = 11 , Msgobject = Receive , Msglen = 1 , Autoreply =
            Disabled , Clearmob = No                    ' reconfig with value -1 for the current MOB
      Elseif Canstmob.6 = 1 Then                        'transmission ready
         Config Canmob = -1 , Bitlen = 11 , Msgobject = Disabled , Msglen = 1 , Clearmob =
            No                                ' reconfig with value -1 for the current MOB and do not set ID and MASK
      End If
   End If
 Next
 Cansit1 = 0 : Cansit2 = 0 : Cangit = Cangit            ' clear interrupt flags
 Canpage = _can_pageok
Return
```

Kommando-Übersicht	
CANRESET	Reset CAN-Controller, wird auch bei Hardware-Reset ausgelöst
CANCLEARALLMOBS	Löscht alle Message Objects
CANCLEARMOB	Löscht ein bestimmtes Message Object
CANBAUD	Einstellen CAN-Übertragungsgeschwindigkeit
CONFIG CANBUSMODE	Einstellen des Bus-Modus
CONFIG CANMOB	Einstellen der Eigenschaften eines Message Objects
CANGETINTS	Lesen der Message Interrupts
CANSELPAGE	Auswählen eines Message Objects
CANID	Lesen einer CAN-ID
CANRECEIVE	Lesen von Daten
CANSEND	Senden von Daten

`IDTAG=&H0123 , IDMASK=&H0123` : Reagiere nur auf ID &H0123

`IDTAG=&H0123 , IDMASK=&H0120` : Reagiere auf ID &H0120-&H0123

Beim Senden einer Information legt IDTAG die ID der Information fest. IDMASK ist beim Senden nicht relevant.
Weil mehrere MObs aktiv sein können, besteht die Gefahr, dass sie gleichzeitig Interrupts auslösen. Die CAN-Interrupt-Routine prüft deshalb jedes MOb, ob es einen Interrupt erzeugt hat. Wenn das MOb die Interruptquelle war, werden die Daten mit *CANRECEIVE* gelesen, und das MOb wird neu aktiviert. Das ist wichtig, weil das MOb nun wieder freigegeben wird und ohne erneute Aktivierung keine weiteren Informationen empfangen werden können. Ein wiederholtes Einstellen der Definitionen ist nicht notwendig. Die Funktion *CANRECEIVE* speichert die empfangenen Daten in der zugeordneten Variablen und gibt die Anzahl der empfangenen Bytes zurück. Ein Beispiel: *received=CANRECEIVE(portA)*.
Der Programmcode der Interrupt-Routine soll möglichst kurz sein. Dort können Daten gelesen und gespeichert werden. Das Hauptprogramm kann dann an einer Flag erkennen, dass Daten zu verarbeiten sind. Die Interruptroutine des Demo-Programms enthält *Print*-Kommandos, obwohl solche verzögernden Elemente dort eigentlich nicht vorkommen sollten.



Flussdiagramm CAN, Message Object 0 und 1
Mit MOb 0 werden Daten empfangen,
mit MOb 1 wird der Status des DIP-Schalters nach dem Ändern übertragen.

Mit *CANSEND* werden Daten über den CAN-Bus gesendet. Als Parameter müssen das MOb, eine Variable und optional die Anzahl der Byte übergegeben werden. Ein Beispiel:

```
ok= CANSEND(0 , ar(1) , 4)   ' Sende 4 Bytes von Array AR nach MOb 0
```

Diese Funktion gibt den Wert 0 zurück, nachdem die Daten korrekt gesendet wurden. In der Interruptroutine wird das MOb neu konfiguriert.
Auf dem Board des CAN-Controller-Projekts [2] befinden sich ein DIP-Schalter und acht LEDs. Es soll ein Programm geschrieben werden (siehe Flussdiagramm), das in beide Mikrocontroller geladen werden kann. Das Programm (Download von [4]) wartet auf eine Nachricht mit der ID &H0120 und empfängt dabei nur ein Daten-Byte. Das Byte wird bitweise von den LEDs angezeigt. Außerdem wird der DIP-Schalter abgefragt, der Status wird ebenfalls mit der ID &H0120 übertragen. Dass die IDs hier übereinstimmen, ist problemlos möglich.

Nach Ändern der DIP-Schalter-Einstellung auf einem Bord ändert sich die LED-Anzeige auf dem anderen Board. Das ist der Beweis dafür, dass der CAN-Bus Daten transportiert.

(110124)gd

Weblinks

[1] www.mcselec.com
[2] www.elektor-magazine.de/080671
[3] www.elektor-magazine.de/080213
[4] www.elektor-magazine.de/110124

Zweiadriges Interface 2.0
Mit noch geringerem Ruhestrom

Von
Klaus Jürgen Thiesler
(D)

Weniger Adern bedeuten eine erhöhte Zuverlässigkeit. In der Aprilausgabe 2012 haben wir eine auf den Ideen des Autors beruhende Schaltung veröffentlicht, die statt drei Adern für den Anschluss eines Tasters mit LED-Rückmeldung nur noch zwei erfordert [1]. Klaus Jürgen Thiesler hatte außerdem aber noch eine stromsparendere Variante dieses Prinzips in petto, die auch bei Controllern funktioniert, die mit nur 2,1 V versorgt werden. Außerdem fließt bei offenem Taster auch noch weniger Strom. Grund genug also, Ihnen diese Schaltung nicht vorzuenthalten. Noch einmal kurz, worum es geht: Will man einen Taster und eine LED als optische Rückmeldung an einen Mikrocontroller anschließen, dann sind auch bei gemeinsamer Masseleitung normalerweise mindestens drei Adern notwendig. Diese Schaltung hingegen kommt mit einem Drittel weniger aus.

Prinzip

Das Trickreiche der Schaltung ist, dass auch dann Strom durch die LED fließt, wenn sie nicht leuchtet. Dieser Strom ist mit einigen µA aber so gering, dass an der LED zwar eine Flussspannung abfällt, sie aber dunkel bleibt. Leuchtet die LED wirklich, dann müssen einige mA fließen, und die Flussspannung ist etwas höher. Folglich steht an der LED immer eine Spannung zwischen etwa 1,1 V und 1,7 V an, die bei Betätigung des Tasters kurzgeschlossen wird. Das kann man leicht auswerten.

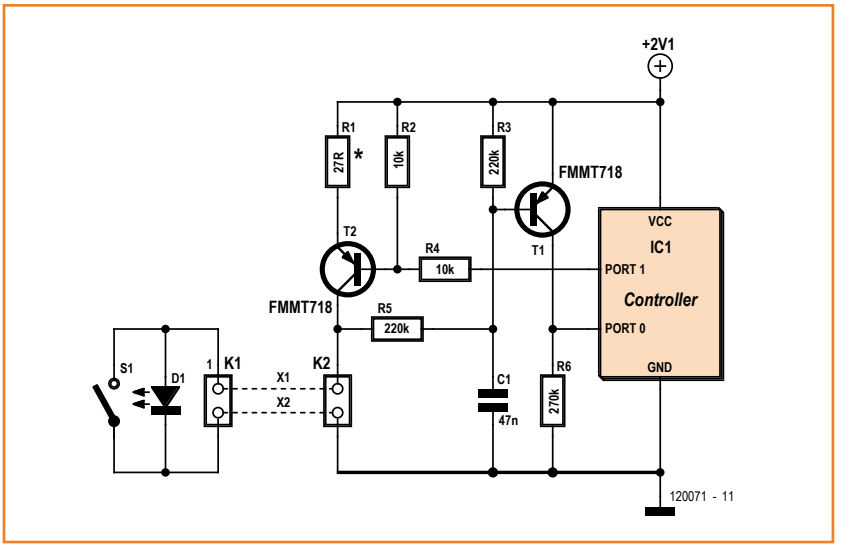
Schaltung

Gegenüber der Schaltung vom April kommen hier zwei ungewöhnlichere PNP-Transistoren mit besonders niedriger U_{CEsat} zum Einsatz. Doch keine Angst: die sind preiswert z.B. bei Reichelt erhältlich. R3 und R5 sorgen für den Ruhestrom durch die LED (etwa 2,5 µA). Soll die LED leuchten, so muss der Mikrocontroller-Ausgang Port.1 „low“ sein. In diesem Fall operiert T1 als Konstantstromquelle: an R1 fällt etwa 0,3 V ab. Durch die LED fließt dann bei R1 = 27 Ω ein kurzschlussfester Strom von rund 10 mA.

Nun zum Taster: R3 und R5 sind so dimensioniert, dass T1 bei einem Kurzschluss der Spannung an X1 durch S1 durchschaltet. Dann wird der Pegel an Port.0 auf „high“ gelegt. Der Tiefpass aus R5 und C1 unterdrückt eventuelle Störpegel an X1. Als Ruhestrom wirkt sich nur der Strom durch die LED aus. Die Interface-Schaltung benötigt bei offenem S1 daher nur etwa 2,5 µA.

Anpassungen

Manche Applikation benötigt eine andere Versorgungsspannung als die angegebenen 2,1 V.



Mikrocontroller arbeiten bei 3,3 V und weniger. Bei V_{CC} = 1,8 V sollte man für R4 einen Wert von 8,2 kΩ und für R5 einen Wert von 160 kΩ vorsehen. Bei 3,3 V erhöhen sich die Werte: R4 = 22 kΩ und R5 = 470 kΩ.

Die Schaltung funktioniert bei VCC ≤ 2,1 V am besten mit roten LEDs. Grüne und gelbe LEDs eignen sich durch ihre höhere Flussspannung erst bei höheren Spannungen. Von blauen und weißen LEDs ist abzuraten. Manche Low-Current-LED-Typen kommen mit deutlich geringeren Strömen aus. Dies kann man durch höhere Werte bei R1 erreichen. Mit 180 Ω landet man bei etwa 2 mA.

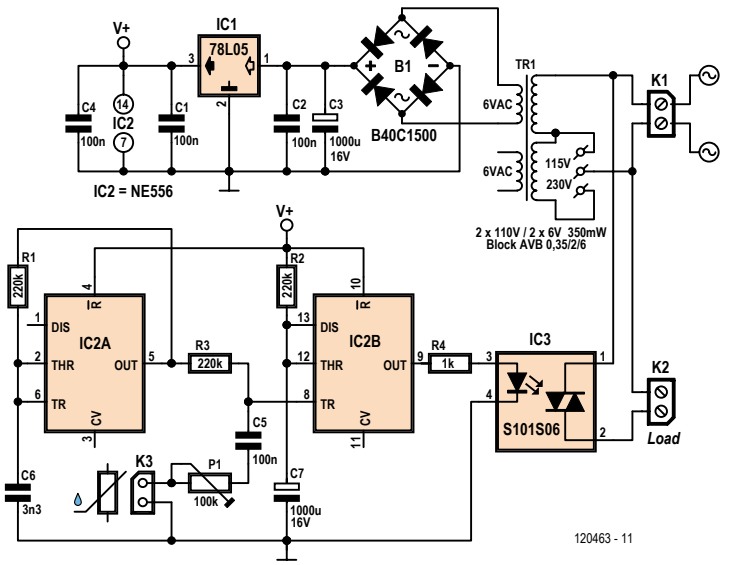
(120071)

[1] www.elektor.de/110572

Feuchte-Schalter

Nützlich für Bad und Waschküche

Von **Leo van der Linde**
(Neuseeland)



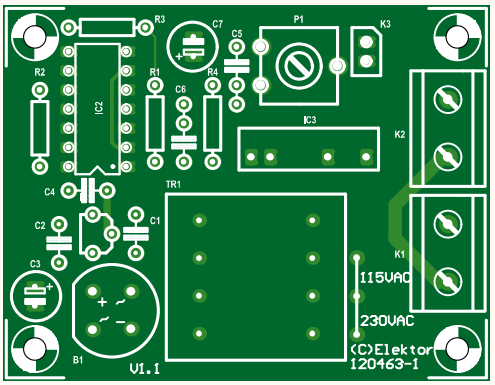
Stückliste

Widerstände:
R1,R2,R3 = 220 k
R4 = 1 k
P1 = 100 k Trimmpoti liegend

Kondensatoren:
C1,C2,C4,C5 = 100 n
C3,C7 = 1000 µ/16 V stehend
C6 = 3n3

Halbleiter:
B1 = Brückengleichrichter
B40C1500
IC1 = 78L05
IC2 = NE556
IC3 = Solid-State-Relais S101S06V (Sharp)

Außerdem:
Tr1 = Netztrafo 2 · 6 V, 0,35 VA (z. B. Block AVB 0,35/2/6)
K1, K2 = Schraubklemmverbinder 2-polig, Raster 7,5 mm
K3 = Stiftkontaktleiste 2-polig, für den Sensor-Anschluss
Betauungssensor SHS A2 (Hygrosens, z. B. Conrad 187606)
Platine 120463-1 (siehe www.elektor-magazine.de/120463)



Der Feuchte-Schalter setzt den elektrischen Lüfter eines Feuchtraums in Betrieb, sobald die Luftfeuchte einen einstellbaren Wert übersteigt. Ein so genannter resistiver Betauungssensor, ein doppelter Timer des Typs 556 und ein Halbleiter-Relais sind die wesentlichen Bauteile.

In der Schaltung ist Timer IC2A als Rechteckoszillator geschaltet, die von R1 und C6 abhängige Frequenz liegt bei 1 kHz. Das Rechtecksignal gelangt über Widerstand R3 zum Trigger-Eingang des Timers IC2B und gleichzeitig über C5 und P1 zum Sensor, der an Klemme K3 angeschlossen ist. Kondensator C5 verhindert die Polarisierung des Sensors durch den Gleichspannungsanteil des Rechtecksignals, mit Poti P1 lässt sich die Empfindlichkeit einstellen. Wegen der exponentiellen Sensor-Charakteristik (< 20 kΩ bei 75 %, < 100 kΩ bei 93 % und < 150 kΩ bei 95 % Relative Luftfeuchte) steigt der Widerstand bei hoher Luftfeuchte sprunghaft an. Da der Sensor-Widerstand einen hohen Wert annimmt, kann das Rechtecksignal den als nachtriggerbares Monoflop geschalteten Timer IC2B triggern. Das Ausgangssignal von IC2B schaltet über Halbleiter-Relais IC3 den Lüfter ein. Der Schaltzustand bleibt erhalten, solange die Luftfeuchte im hohen Bereich liegt. Wenn die Luftfeuchte unter die Ausschaltschwelle sinkt, reicht das Rechtecksignal an Monoflop IC2B nicht mehr aus, um das Monoflop zu triggern. Weil das Monoflop erst nach etwa vier Minuten in den Ausgangszustand zurückkehrt, läuft der Lüfter für vier Minuten nach. Auch nach Anlegen der Betriebsspannung an die Schaltung ist das Monoflop getriggert. In diesem Fall vergehen ebenfalls etwa vier Minuten, bis der Lüfter bei niedriger Luftfeuchte abschaltet.

Für die Schaltung wurde eine Platine entworfen, auf der nur bedrahtete Bauelemente vorkommen. Das Anschließen der Platine bedarf eigentlich keiner Erklärung: An Klemme K1 liegt die Netzspannungsleitung, die vorher zum Lüfter führte, der Lüfter wird mit Klemme K2 verbunden. Wegen des Betriebs in Feuchträumen muss die Platine in ein vor Tropfwasser geschütztes Gehäuse eingebaut werden. Vor dem Einbau in das Gehäuse kann ein Blick auf die Drahtbrücke nicht schaden,

die für die Wahl der Netzspannung verantwortlich ist. Die Drahtbrücke muss korrekt positioniert sein. Die Leitungen zum Stromnetz, zum Lüfter und zum Sensor werden mit Silikonmasse abgedichtet. Mit etwas Glück bietet das Lüftergehäuse genügend freien Raum, um den Feuchte-Schalter dort einzubauen. Der Sensor muss natürlich seinen Platz an einem Ort haben, wo er mit der feuchten Luft in Kontakt kommt. Mit P1 wird die Einschaltschwelle so eingestellt, dass der Lüfter bei etwa 90 % Relative Luftfeuchte anläuft. Auf die genaue Zahl kommt es nicht an, es genügt ein „gefühlter“ Wert.

Beim Aufbau des Feuchte-Schalters sind die einschlägigen, für Feuchträume geltenden Schutzregeln unbedingt zu beachten. **Netzspannung an dem im Feuchtraum zu installierenden Sensor kann lebensgefährlich sein!**

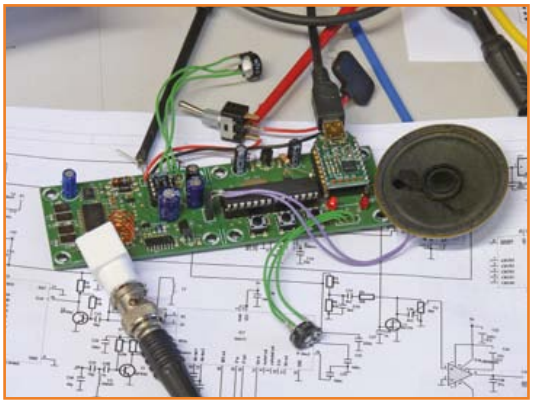
(120463)gd

Antenne für Flugfunk-Band

Passend zum Flugfunk-Scanner

Von **Gert Baars** (NL)

Diese Antenne, ausgelegt für das zivile Flugfunk-Band 108...137 MHz, überrascht durch ihre Leistung in Kombination mit dem an anderer Stelle in dieser Elektor-Ausgabe beschriebenen Flugfunk-



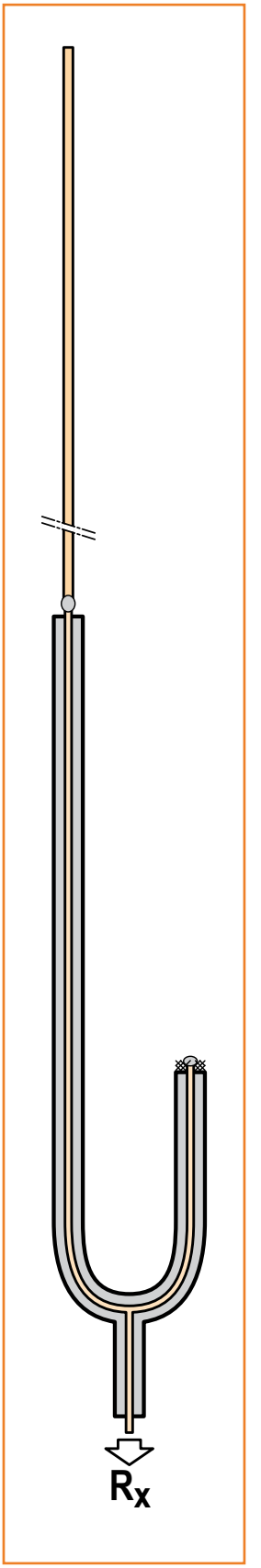
Scanner. Sie kann schnell aus einem 117 cm langen Volldraht und Koaxialkabel des Typs RG58 oder RG174 zusammengebaut werden.
Die Antenne besteht aus einer Rute der Länge λ/2 sowie zwei Koaxialkabel-Abschnitten mit den Längen 12 cm und 46 cm. Die Koaxialkabel werden zu einem Impedanz-Transformator zusammengeschaltet, die Ausgangsimpedanz beträgt 50 Ω. Zu den genannten Längen muss jeweils ein Zentimeter für die Lötverbindungen hinzugerechnet werden.

Das 12 cm lange Koaxialkabel ist auf einer Seite kurzzuschließen, Innenleiter und Mantel werden miteinander verlötet. Die andere Seite wird mit dem 46 cm langen Koaxialkabel parallel geschaltet. Von diesem Punkt führt ein 50-Ω-Kabel, gegebenenfalls über einen Stecker, zum Antenneneingang des Scanners. Die obere Seite des langen Koaxialkabels ist der Fußpunkt der 117 cm langen Antennenrute.

Die Rute kann aus massivem Kupferdraht, Durchmesser 1 mm oder Querschnitt 1,5 mm² bestehen.

Die Antenne darf unter Dach ungeschützt bleiben, für Aufstellungsorte im Freien ist das Umhüllen mit einem PVC-Schutzrohr empfehlenswert.

(120678)gd



Einfaches Fahrradnetzteil

5 V durch Muskelkraft

Von **Philip Jaschewski**
(Elektor Labor)

Bei einer Radtour hat man im Jahre 13 nach der letzten Jahrtausendwende keine gedruckten Karten mehr dabei. Ein Smartphone genügt! Da die nächste Steckdose aber meist weit weg ist, wäre ein 5-V-Anschluss am Fahrrad nicht schlecht. In diesem Beitrag steht, wie das ganz einfach geht.

Es liegt sehr nahe, die Energiequelle eines Fahrrads auch für all die kleinen elektronischen Begleiter anzuzapfen. Bei diesem Vorsatz kommt einem entgegen, dass heute fast jede portable Elektronik mit 5 V zufrieden ist. Man benötigt also in der Regel nur ein einziges Fahrrad-Festspannungsnetzteil für die unterschiedlichsten Geräte. So ein Netzteil muss die Dynamospannung in 5-V-Gleichspannung umwandeln. Das Ziel ist geklärt – jetzt folgt die praktische Umsetzung.

Der übliche Fahrrad-Dynamo liefert „normgemäß“ 6 V mit einer Leistung von 3 W.

Dabei dominieren zwei Ausführungen: klassische außen am Rad angebrachte Dynamos mit Reibrad-Antrieb und Nabendynamos. Beide liefern Wechselspannungen im Bereich einiger zig bis einiger hundert Hz. Die Wechselspannung muss also zunächst in Gleichspannung verwandelt werden und diese dann in stabile 5 V. Zur Erreichung dieser Anforderungen braucht es keine Zauberkräfte und noch nicht einmal einen Mikrocontroller.

Wie die Schaltung in **Bild 1** zeigt, besteht das Fahrradnetzteil aus einem klassisch beschalteten Spannungsregler plus Brückengleichrichter. Die Besonderheiten sind Folgen der Eigenschaften von Dynamos: Mit keiner oder wenig Last steigt dessen Spannung recht linear mit der Drehzahl an. Schon bei moderatem Tempo hat man dann mehr als 6 V≈ am Brückengleichrichter. Bei Rennfahrertempo kommt man auf deutlich mehr als 12 V≈. Da es sich um eine halbwegs sinusförmige Wechselspannung handelt, wird der Ladeelko C1 ohne große Last auf die Spitzenspannung aufgeladen. IC1 verträgt zwar 25 V am Eingang. Sicherheitshalber ist aber noch mit D1 eine Z-Diode vorhanden, die einen Spannungsanstieg auf allzu große Werte verhindert.

Am Eingang der Schaltung ist der doppelpolige Umschalter S1 zu finden. Mit ihm kann man zwischen der Versorgung der Beleuchtung und dem Fahrradnetzteil umschalten. Beides gleichzeitig

Technische Daten

- Umschaltbar zwischen Licht und Netzteil
- Ausgangsspannung: 5 V
- Ausgangsstrom: min. 0,5 A
- Geeignet zur Versorgung/Aufladung portabler Geräte

geht nicht gut, denn auch wenn man sicher etwas mehr als die nominalen 3 W aus dem Dynamo quetschen kann – das Doppelte klappt nicht immer. Also entweder Licht oder Netzteil.

Damit die Sache halbwegs effektiv ist, haben wir für IC1 statt eines 7805 einen so genannten Low-Drop-Out-Regler (LDO) verwendet, so dass der Spannungsregler schon bei kleiner Differenz zwischen Ein- und Ausgangsspannung gut arbeitet. „Normale“ 7805-Versionen benötigen mindestens 8 V am Eingang. Etwas bessere Varianten kommen mit 7 V aus. Andere ICs wie der Pin-kompatible LM2931T benötigen hierfür weniger als 5,6 V. Auch am Spannungsabfall des Brückengleichrichters kann man noch etwas drehen, wenn man von gewöhnlichen Silizium- auf Schottky-Dioden wie den Typ SB140 oder 1N5817 umsattelt.

Die Layout-Dateien kann man über die Elektor-Webseite [1] zu diesem Artikel herunterladen. Man kann dort auch eine professionell gefertigte Platine bestellen.

Dank einfacher bedrahteter Bauteile ist die Bestückung sicher kein Problem. **Bild 2** zeigt das 3D-Modell der bestückten Platine. Die fertige Platine kann man prima mit einem kleinen 6-V-Trafo testen. Am Fahrrad sollte sie in ein isolierendes Kunststoffgehäuse, denn Elektronik und Wasser sind bekanntlich keine engen Freunde.

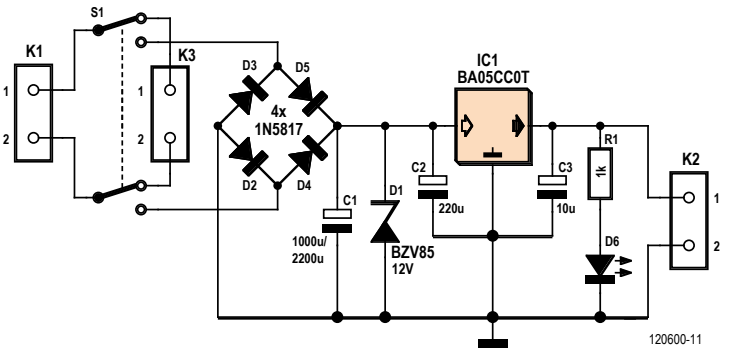


Bild 1.
Die Schaltung des Fahrradnetzteils ist ausgesprochen einfach.

Ein nicht zu vergessender Punkt ist, dass manche Fahrräder und etliche Dynamos das Gehäuse bzw. den Rahmen des Fahrrads als zweiten Leiter nutzen. Der Dynamo hat dann unter Umständen keine zwei Anschlüsse, sondern eben nur einen. In diesem Fall kommt der zweite Anschluss der Platine (z.B. K1-2) eben an den Rahmen. Die Masse des Ausgangs K2-2 sollte dann aber keinesfalls mit metallischen Teilen des Fahrrads in Berührung kommen – auch nicht über das angeschlossene Gerät.

(120600)

Weblink

[1] www.elektor-magazine.de/120600

[2] http://eagleup.wordpress.com

Stückliste

Widerstände:
R1 = 1 k

Kondensatoren:
C1 = 1000 µ/35 V, Elko, RM 5 mm, ø 10 mm
C2 = 100 µ/35 V, Elko, RM 2,5 mm, ø 6 mm
C2 = 10 µ/16 V, Elko, RM 2,5 mm, ø 5 mm

Halbleiter:
D1 = Z-Diode, 1 W, 24 V
D2..D5 = 1N5817 oder SB140*
IC1 = BA05CC0T, LDO, 1 A, TO220-Gehäuse*
LED1 = LED, grün, 5 mm

Außerdem:
K1..K3 = zweipolige Schraubklemme für Platinenmontage, RM 5 mm
S1 = doppelpoliger Umschalter, 24 V/1 A
Platine # 120600-1

* siehe Text

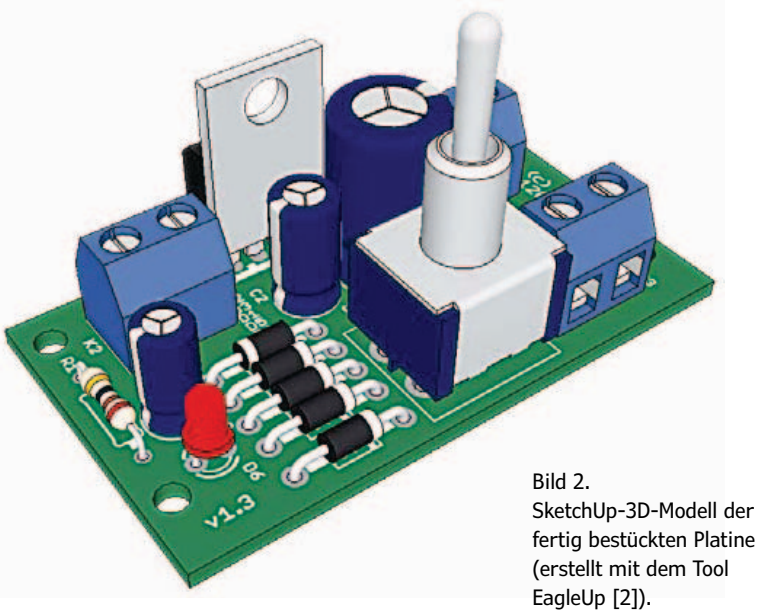


Bild 2.
SketchUp-3D-Modell der fertig bestückten Platine (erstellt mit dem Tool EagleUp [2]).

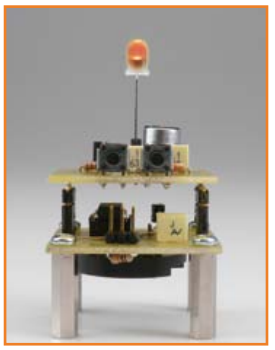


Bild 1.
Die beiden Seiten der Schaltung.

Duo-LED-Kerze

Mit einstellbarer Farbe und Luftzugsensor

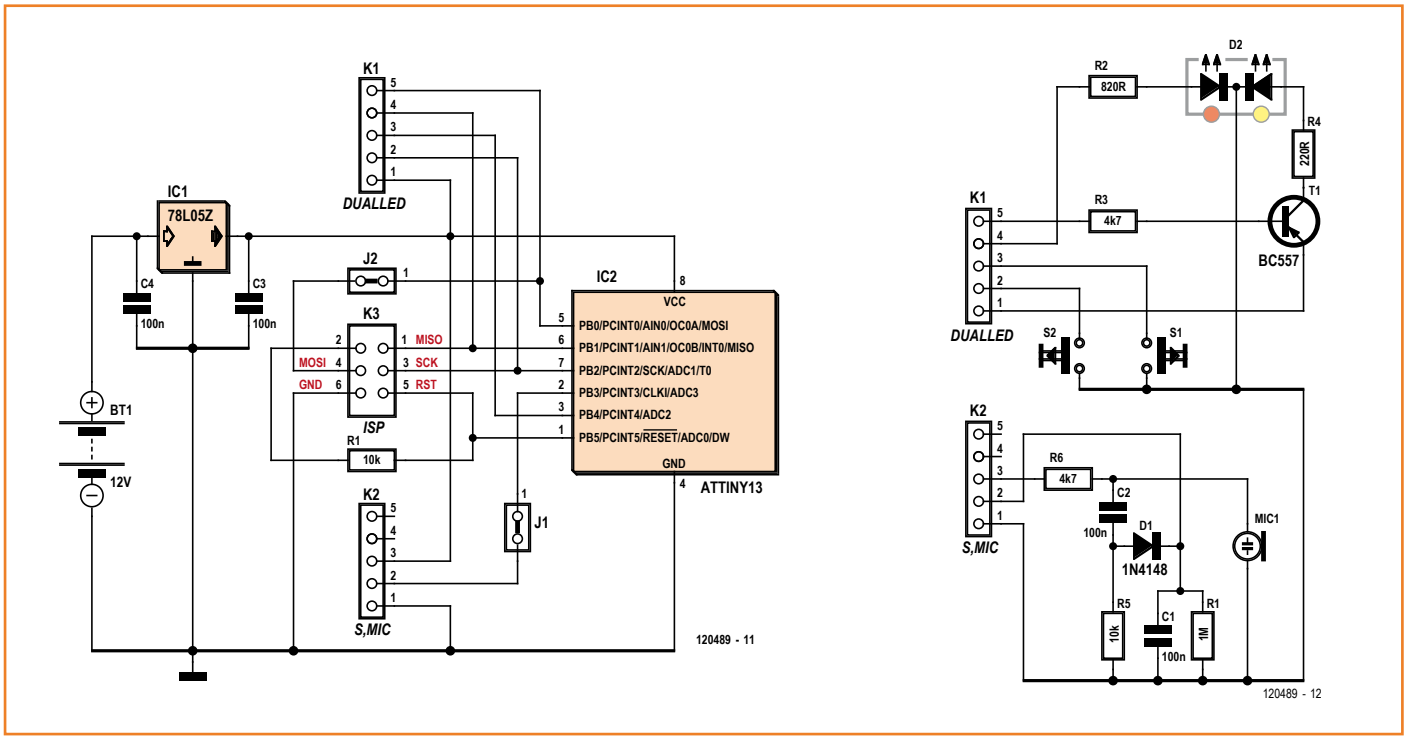
Um eine komfortable LED-Kerze aufzubauen, werden hier ein Tiny-Controller und eine Duo-LED eingesetzt.

Von **Jörg Trautmann**

LED-Kerzen gibt es in den verschiedensten Variationen. Meist simpel aufgebaut, befindet sich im Inneren ein Chip, wie er auch in Melodie-Glückwunschkarten verwendet wird. Die abgespielte Melodie soll die LED so modulieren, dass sich eine einigermaßen realistisch flackernde Kerzenflamme ergibt. Bei teureren Modellen lässt sich die LED-Kerze durch einen Zugluftsensor realistisch „auspusten“. Doch eine naturgetreue Flammenfarbe bekommt kaum ein Modell hin: Entweder ist die LED kalt gelb leuchtend oder die Flamme ist unrealistisch stark orange. Auch die durch die Musik wild zuckende LED erzeugt nicht wirklich ein echtes Kerzenfeeling.
Um eine LED-Kerze mit Zugluftsensor und variabler Flammenfarbe zu entwickeln, benötigen wir zunächst einmal Informationen über die fraglichen Farben. Eine normale Kerzenflamme leuchtet in etwa mit einer Wellenlänge von 600°nm. Eine gelbe LED überstreicht einen Bereich von 565…590 nm, eine rote von 625…740°nm. Um die Lücke zwischen 590°nm und 625°nm zu schließen, müssen die beiden LED-Farben lediglich in der richtigen Intensität gemischt werden. Perfekt geeignet für diese Aufgabe ist eine Duo-LED mit den Farben rot und gelb.

Ein winziger Controller

Die hier vorgestellte Schaltung basiert auf einem ATtiny13 [1]. Dieser Winzling von Controller verfügt über zwei PWM-Ausgänge (OC0A und OC0B), mit denen die Helligkeit der Duo-LED verändert werden kann. Die gelbe LED wird über einen PNP-Transistor T1 angesteuert, die rote ist direkt am Mikrocontroller angeschlossen. Dies ist mög-



lich, da der Rotanteil nur gering ist; durch den Vorwiderstand R2 von 820 Ω bleibt der Strom unterhalb des zulässigen Wertes von 5 mA. Bei der gelben LED, die die Hauptarbeit übernimmt, ist der Strombedarf wesentlich größer. Legt man für diese LED eine zulässige Stromaufnahme von 15 mA zugrunde, errechnet sich für R4 ein Wert von 220 Ω. Da der Mikrocontroller lediglich 5 mA treiben kann, wird zur Ansteuerung T1 benötigt. Um die Farbe der Kerzenflamme einzustellen, gibt es zwei Taster. Solange S1 geschlossen ist, nimmt kontinuierlich der Rotanteil zu und der Gelbanteil ab. Durch Drücken von S2 lässt sich der Vorgang umkehren. Das Puls-Pausenverhältnis kann in 256 Stufen, entsprechend 8 bit, verändert werden.
Um nun realistisch eine Kerzenflamme nachzuahmen, wird das eingestellte Puls-Pausenverhältnis mittels Zufallsgenerator für die gelbe LED um maximal 35 % und für die rote LED um maximal 10 % verändert. Damit ergibt sich ein recht naturgetreuer, schwacher Flackereffekt. Wer die optimale Flammenfarbe eingestellt hat, möchte diesen Zustand beim nächsten Einschalten sicher wieder aktivieren. Deshalb werden die zuletzt eingestellten Werte im EEPROM des Mikrocontrollers (dazu reichen die 64 Byte aus!) abgelegt und nach einer Stromunterbrechung wieder hervorgeholt.

Um den Zugluftsensor zu realisieren, wird die Elektret-Mikrofonkapsel Mic1 verwendet. Über die Diode D1 und Kondensator C1 wird die Mikrofonspannung leicht geglättet einem A/D-Wandler zugeführt. R1 fungiert als Entladewiderstand. Der Triggerlevel ist so gewählt, dass ein leichter Luftzug noch keinen Einfluss auf das Flammenbild hat. Wird nun etwas stärker gepustet, flackern die LEDs und werden dunkler. Wird länger als zwei Sekunden gepustet, verlöschen die LEDs. Ein erneutes kurzes Anpusten erweckt die LED-Kerze wieder zum Leben. Je nach Mikrofon-Empfindlichkeit muss die Konstante `Trigger_value` angepasst werden. Ein höherer Wert verringert die Luftzug-Empfindlichkeit. Damit man diesen Wert auch bequem ändern oder andere Modifikationen in der Software vornehmen kann, ist die ISP-Programmier-Schnittstelle auf K3 herausgeführt.

Zur Spannungsversorgung können zwei Knopfzellen CR2032 verwendet werden. Da der Mikrocontroller ATtiny13 mit maximal 5,5 Volt betrieben werden darf, kommt ein Low-Drop-Spannungsregler 78L05 zum Einsatz. Da die Schaltung einen Ruhestrom von etwa 1°mA zieht, sollten bei längerem Nichtgebrauch die Batterien entfernt werden.

Aufbau auf zwei Platinen

Um die ganze Angelegenheit so kompakt wie möglich zu halten, haben wir zwei Platinchen entworfen, die über Steckverbinder miteinander verbunden werden. Auf der Controllerplatine sind zwei Drahtbrücken einzusetzen, die LED-Platine kommt ohne aus. Ansonsten gibt es wenig Aufregendes über die Bestückung zu berichten. Der Controller ist fertig programmiert erhältlich, kann aber auch in Eigenregie programmiert werden. Die Software steht kostenlos auf der Elektor-Website zur Verfügung [2].

(120489)

Weblinks

[1] www.atmel.com/Images/doc2535.pdf

[2] www.elektor-magazine.de/120489

Stückliste

Controllerplatine

Widerstand:
R1 = 10 k

Kondensatoren:
C3 = 100 n
C4 = 100 µ

Halbleiter:
IC1 = 78L05
IC2 =Atmel ATtiny13-20PU, programmiert 120489-41

Außerdem:
K1,K2 = Stiftleiste, 5-pol.
K3 = DIL-Stiftleiste, 2x3-polig
Platine 120489-1

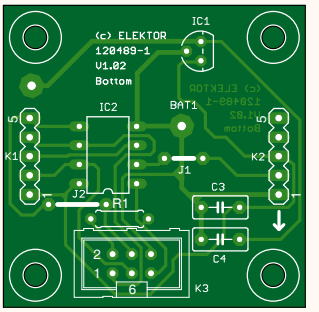
LED-Platine

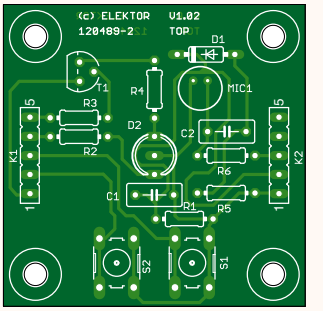
Widerstände:
R1 = 1 M
R2 =820 Ω
R3,R6 = 4k7
R4 = 220 Ω
R5 = 10 k

Kondensatoren:
C1,C2 = 100 n

Halbleiter:
D1 = 1N4148
LD1 = Duo-LED gelb/rot
T1 = BC557

Außerdem:
Mic1 = Elektret-Mikrofonkapsel
S1,S2 = Taster
K1,K2 = Pinsockel, 5-pol.
Platine 120489-2



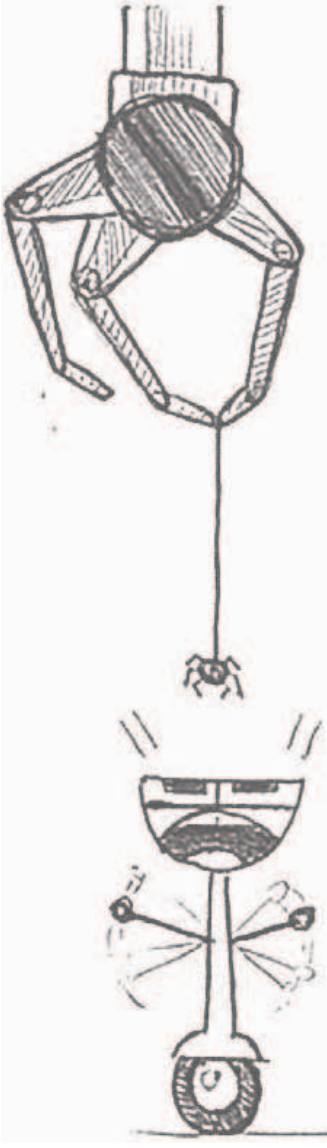


Gadgeteer

Rapid Prototyping à la Microsoft

Von **Clemens Valens**
(Elektor.LABS)

Anfang 2011 startete Microsoft eine Rapid-Prototyping-Plattform unter der Bezeichnung .NET Gadgeteer, die auf dem .NET Micro Framework (NETMF) basiert. Wie viele andere solcher Systeme soll es die Erstellung von elektronischen Systemen durch Nichtspezialisten erleichtern, indem diese von der Komplexität der Hardware verschont bleiben. Anders als bei ähnlichen Systemen bietet Microsoft aber keine passende Hardware, sondern lediglich detaillierte Spezifikationen samt Software.



Es brauchte dann etwas Zeit, bis das erste kompatible Mainboard erschien, da die Portierung des Frameworks auf Mikrocontroller-Hardware nicht trivial ist. Sogar heute nach zwei Jahren gibt es nur wenige Mainboards und Erweiterungs-Boards von anderen Herstellern.

Was ist ein Gadgeteer?

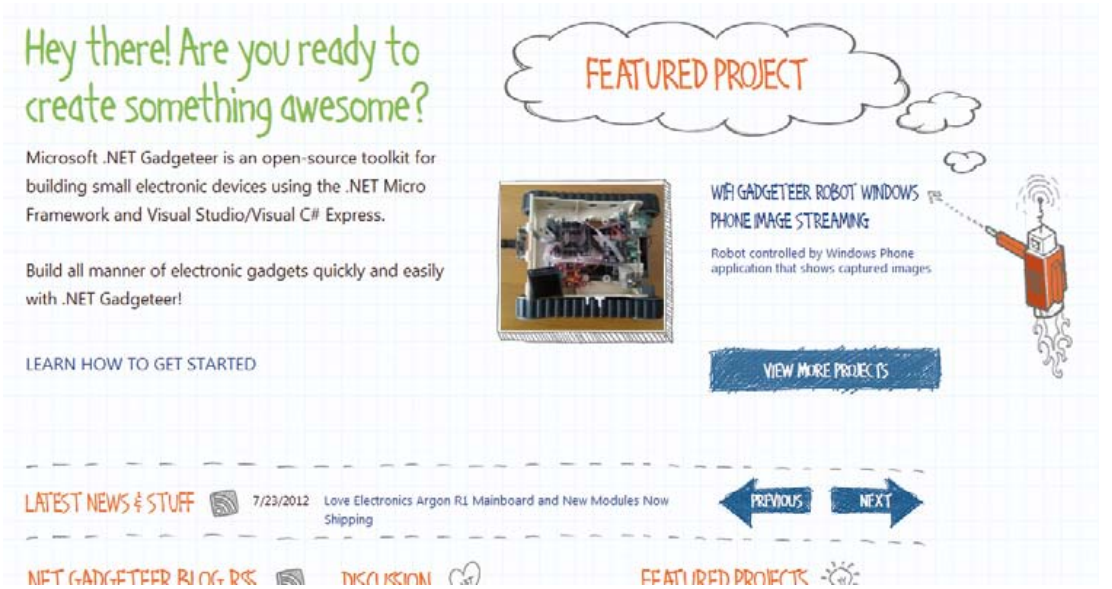
Ein Gadgeteer-System besteht aus einem Mainboard, an das man Erweiterungs-Boards anschließen kann, die Module genannt werden. Die Verbindungen werden dabei per Flachbandkabel über zehnpolige Wannenstecker (2x5) – die so genannten „sockets" – vorgenommen. Auch wenn alle Sockel gleich aussehen: Labels bezeichnen, welchen Typ von Modulen man damit verbinden darf (siehe **Tabelle 1**). Jeder Typ hat seinen eigenen Buchstaben. Ein Sockel vom Typ A bietet beispielsweise drei analoge Eingänge, einen GPIO (**G**eneral-**P**urpose **I**nput/**O**utput) sowie drei nicht angeschlossene Pins. Der P-Typ hat drei PWM-Ausgänge, zwei GPIO-Pins plus zwei nicht angeschlossene Pins. Bei allen Sockeln ist Pin 1 mit 3,3 V, Pin 2 mit 5 V und Pin 10 mit GND belegt. Da die Typen A und P komplementär belegt sind, ist auch ein AP-Sockel möglich. Es gibt noch zwei reine GPIO-Sockel-Typen: der Typ X hat drei und der Typ Y gleich sieben GPIOs. Auch damit sind Kombinationen möglich. Die Sockel sind daher z.B. mit X oder Y oder einer Buchstabenkombination beschriftet. Z steht für einen MS-Sockel (**M**anufacturer **S**pecific), dessen sieben nutzbare Pins unspezifiziert bleiben.

Die Module haben also einen oder mehrere gelabelte Sockel, die nur mit anderen Sockeln des passenden Typs verbunden werden dürfen. Ein Modul-Sockel mit einem Stern (*) indiziert, dass hier ein Daisy-Chaining via einem DaisyLink-Protokoll möglich ist. Dabei handelt es sich um ein I²C-basiertes-Protokoll, das mit einem 1-Draht-Bus für Initialisierung und Interrupts kombiniert ist.

Die Gadgeteer-Hardware-Spezifikation erlaubt auch so genannte Shields = Tochterplatinen, die auf ein anderes Board gesteckt sind. Dies dient zur Erweiterung bereits existierender Hardware mit Gadgeteer-Fähigkeiten.

Microsoft hat keinen speziellen Controller für Gadgeteer vorgeschrieben, da die Plattform Hardware-unabhängig sein soll. Die bisher erhältlichen Mainboards sind ARM-basiert, doch sind auch andere Controller denkbar. Die Unterschiede auf Hardware-Ebene werden von Treiber-Libraries (DLLs) behandelt, die vom Hardware-Hersteller geliefert werden. Das Gadgeteer-System scheint deutlich von der PC-Architektur beeinflusst: Der unter Windows laufende PC ist durch das Mikrocontroller-Board und der USB-Bus durch DaisyLink ersetzt.

Man kann sich ein individuelles System zusammenstellen, indem man die benötigten Module mit einem Mainboard kombiniert. Das System kann man dann mit einem grafischen Tool konfigurieren, mit dem man die Module und ihre



Verbindungen zeichnet. Man kann damit sogar die Module „verdrahten“ (siehe **Bild 1**). Futuristisch wäre es, wenn man das reale System per Webcam erfassen könnte. In der Gegenwart genügt ein Klick auf einen Knopf, damit ein Software-Framework für das System erstellt wird, das Software-Objekte für alle Funktionen des Moduls bereit stellt (**Bild 2**). Selbstverständlich müssen zuvor die Treiber installiert werden. Wenn das Framework steht, kann man mit der Programmierung anfangen, was gleichzeitig der Punkt ist, wo die Einfachheit ein Ende hat.

Common Language Runtime?

Ein Gadgeteer-Mainboard ist ein recht leistungsfähiges 32-bit-Mikrocontroller-System mit mindestens 390 KB Programmspeicher und 64 KB RAM. Diese Angaben stammen aus der Dokumentation des NETMF-Porting-Kit (.**NET M**icro **F**ramework). Der viele Speicher ist unter anderem für die CLR (**C**ommon **L**anguage **R**untime) notwendig, die zur Ausführung von Anwender-Programmen erforderlich ist. Es existiert auch noch eine kleinere Variante unter dem Namen TinyCLR, sodass man mit etwas Glück mit 256 KB auskommt. Da aber auch

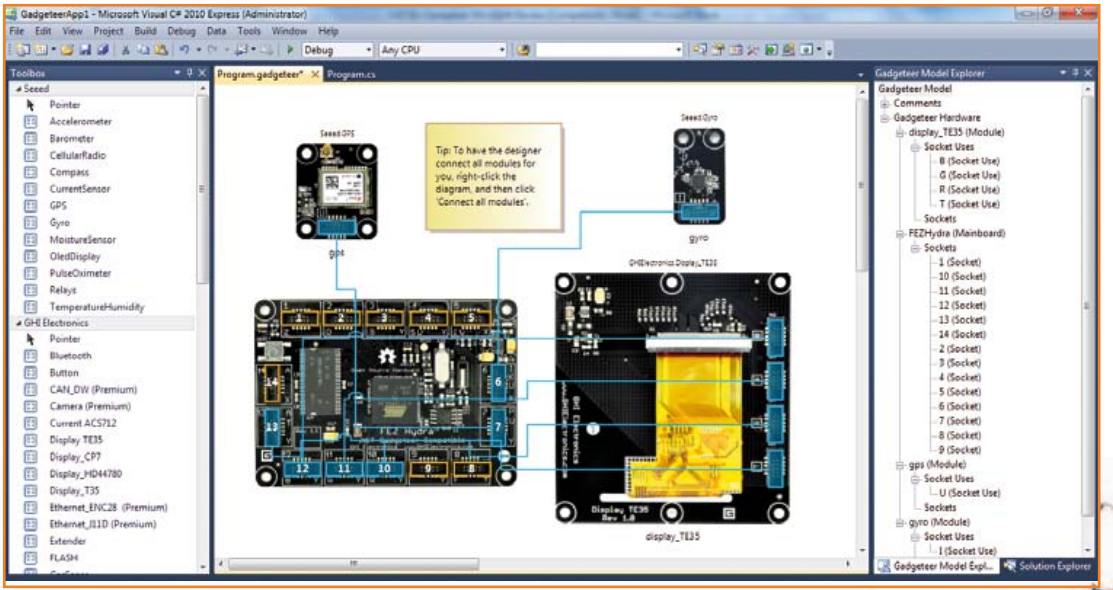


Bild 1.
Nach Installation von Software und den SDKs sowie dem Aufklappen des rechten Fensters konnte ich dieses System erstellen. Dabei verbindet die Software die Module. Die linke Toolbox enthält die Module, die ich hier installiert hatte. Rechts kann man die in der Hardware verfügbaren Sockel sehen.

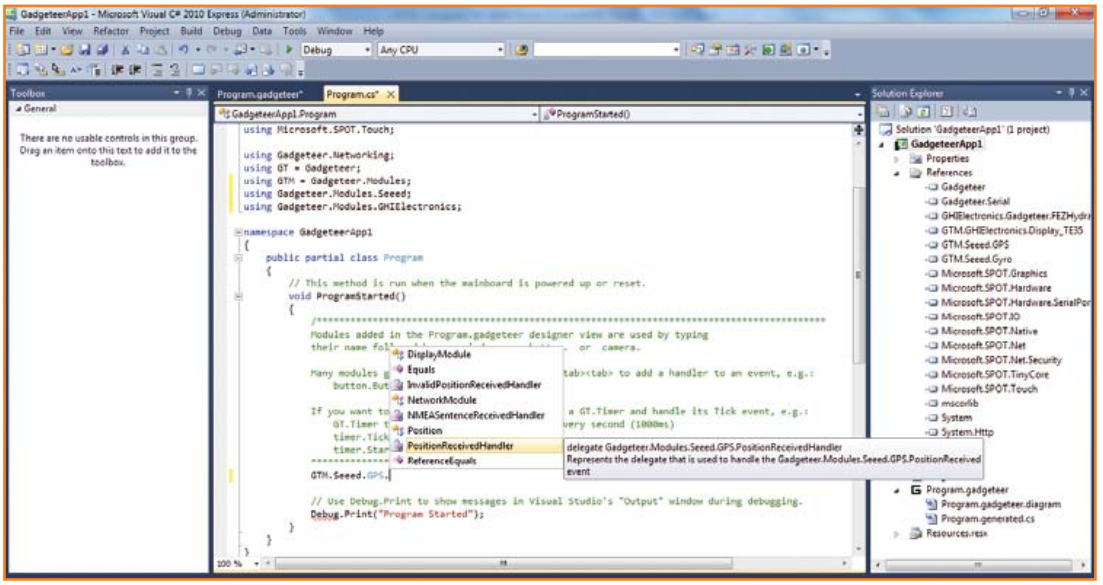


Bild 2.
Das ist das für mein System erstellte Programm. Namespace, partial Cass etc. – man braucht schon etwas Programmiererfahrung, um das alles zu verstehen. Wenn man (wie ich hier für das GPS-Modul) einen Befehl hinzufügt, schlägt die IDE eine Liste an Optionen zur Vervollständigung des Befehls vor. Und auch hier muss man viel wissen, um diese Optionen zu verstehen.

das Anwenderprogramm Platz benötigt, bieten übliche Mainboards schon oft einen Programmspeicher von 512 KB und mehr als 64 KB RAM. Mit dem erwähnten PK (**P**orting **K**it) kann man bei Bedarf eine an eigene Hardware angepasste CLR erstellen. Auch wenn viele Beispiele zur Verfügung stehen, ist das nichts für jemand mit schwachen Nerven. Man könnte sogar eine CLR für bisher nicht unterstützte Mikrocontroller erstellen – und wäre damit unter den Ersten, die sich so etwas zutrauen. Zur Zeit werden neben ARM die Familien Blackfin von Analog Devices und SH2 von Renesas unterstützt.

Eine CLR hat zwar Ähnlichkeiten zu einem Betriebssystem, ist aber nur ein Befehls-Interpreter, der die Befehle der Anwendung zur Laufzeit in den Code der Zielmaschine umsetzt. Dabei werden Funktionen wie Multi-Threading, Timer, Speicher- und Exception-Management unterstützt, was sehr kompakte Programme ermöglicht. Wenn ein Programm die CLR-Möglichkeiten nutzt, wird es als „managed“ bezeichnet. Programme, die alles selbst erledigen, sind „unmanaged“. Microsoft verspricht eine Ausführungsgeschwindigkeit von 550 managed Methoden-Aufrufen pro MHz Controller-Takt. Diese Angabe ist leider nicht so plastisch wie etwa eine maximale Pin-Toggle-Frequenz oder ähnliche Werte. Aktuell versteht eine NETMF-CLR nur in C# (gesprochen wie das Englische „see sharp“) erstellten Code. Die Code-Dateien haben die Extension „cs“. C# ist eine OO- (**O**bjekt-**O**rientierte) und Event-gesteuerte Sprache. Sie unterscheidet sich nicht viel von anderen OO-Varianten wie C++ oder Java. Die Tools zur Entwicklung von Code in C# stehen kostenlos zur Verfügung und werden mit Visual C# 2010 Express installiert. Es ist noch nicht klar, wann die Version 2012 unterstützt wird. Man muss neben dem .NET-SDK und dem Gadgeteer-Core noch sogenannte Builder-Templates installieren. Außerdem muss für jede Hardware natürlich auch der passende Treiber installiert sein.

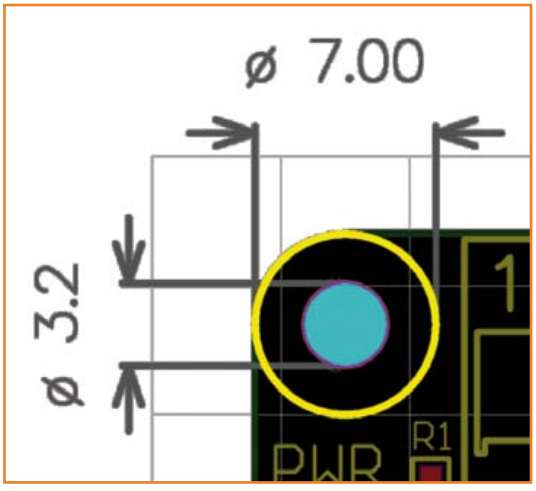


Bild 3.
Microsoft definiert detaillierte Spezifikationen für die Gadgeteer-Hardware. Wie man sieht, müssen sogar die Ecken der Platinen abgerundet und bestimmte Befestigungslöcher vorgesehen werden.

Das Schreiben der Applikation für das Anwender-System – das „device“ — ist noch einfach. Dennoch befindet man sich damit auf einer anderen Ebene wie beim simplen Paradigma eines Arduino-Systems (siehe **Kasten**). Die Gadgeteer-Webseite schreibt dazu: *„.NET Gadgeteer verwendet*

Tabelle 1. Gadgeteer-Sockel-Typen. Legende:

GPIO	Ein digitaler Vielzweck-Ein- oder Ausgang mit 3,3-V-Pegel.
(G)	Ein Pin, der zusätzlich zu anderen Funktionen auch als GPIO dienen kann.
(OPT)	Vom Mainboard oder einem Modul optional unterstützter Sockel-Typ.
[UN]	Module bei diesem Sockel-Typ nicht mit diesem Pin verbinden.
[MS]	Ein Hersteller-spezifischer Pin.
!	GPIO mit Software-Pull-up und Interrupt.

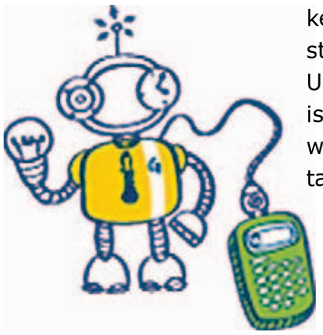
Typ	Buchstabe	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8	Pin 9	Pin 10
3 GPIO	X	+3.3V	+5V	GPIO!	GPIO	GPIO	[UN]	[UN]	[UN]	[UN]	GND
7 GPIO	Y	+3.3V	+5V	GPIO!	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GND
Analog-Eingang	A	+3.3V	+5V	AIN (G!)	AIN (G)	AIN	GPIO	[UN]	[UN]	[UN]	GND
CAN	C	+3.3V	+5V	GPIO!	TD (G)	RD (G)	GPIO	[UN]	[UN]	[UN]	GND
USB-Device	D	+3.3V	+5V	GPIO!	D-	D+	GPIO	GPIO	[UN]	[UN]	GND
Ethernet	E	+3.3V	+5V	[UN]	LED1 (OPT)	LED2 (OPT)	TX D-	TX D+	RX D-	RX D+	GND
SD-Karte	F	+3.3V	+5V	GPIO!	DAT0	DAT1	CMD	DAT2	DAT3	CLK	GND
USB-Host	H	+3.3V	+5V	GPIO!	D-	D+	[UN]	[UN]	[UN]	[UN]	GND
I2C	I	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	[UN]	SDA	SCL	GND
UART+ Handshake	K	+3.3V	+5V	GPIO!	TX (G)	RX (G)	RTS	CTS	[UN]	[UN]	GND
Analog-Ausgang	O	+3.3V	+5V	GPIO!	GPIO	AOUT	[UN]	[UN]	[UN]	[UN]	GND
PWM	P	+3.3V	+5V	GPIO!	[UN]	[UN]	GPIO	PWM (G)	PWM (G)	PWM	GND
SPI	S	+3.3V	+5V	GPIO!	GPIO	GPIO	CS	MOSI	MISO	SCK	GND
Touch	T	+3.3V	+5V	[UN]	YU	XL	YD	XR	[UN]	[UN]	GND
UART	U	+3.3V	+5V	GPIO!	TX (G)	RX (G)	GPIO	[UN]	[UN]	[UN]	GND
LCD 1	R	+3.3V	+5V	LCD R0	LCD R1	LCD R2	LCD R3	LCD R4	LCD VSYNC	LCD HSYNC	GND
LCD 2	G	+3.3V	+5V	LCD G0	LCD G1	LCD G2	LCD G3	LCD G4	LCD G5	BACK-LIGHT	GND
LCD 3	B	+3.3V	+5V	LCD B0	LCD B1	LCD B2	LCD B3	LCD B4	LCD EN	LCD CLK	GND
Hersteller-spezifisch	Z	+3.3V	+5V	[MS]	[MS]	[MS]	[MS]	[MS]	[MS]	[MS]	GND
DaisyLink	*	+3.3V	+5V	GPIO!	GPIO	GPIO	[MS]	[MS]	[MS]	[MS]	GND

das .NET Micro Framework und macht damit das Schreiben von Code für Ihr Device so einfach wie für eine Desktop-, Web- oder Windows-Phone-Applikation." Und wann haben Sie zum letzten Mal eine Desktop-, Web- oder Windows-Phone-Applikation geschrieben?

Vom Start weg ist man mit einem recht komplexen Default-Programm konfrontiert, in das man seinen Initialisierungs-Code und Event-Handler einbauen kann. Kein Problem für einen erfahrenen Software-Ingenieur, doch der durchschnittliche Hardware-Entwickler wird sich zu Anfang etwas verloren vorkommen. Nach etwas Übung sollte man aber recht zügig eine eigene Applikation auf die Beine gestellt bekommen.

Fazit

Microsoft versucht, ein NETMF-basiertes Ökosystem aus der Welt der PCs und Smartphones auf die Ebene von Mikrocontrollern zu übertragen. Die Controller-Boards abstrahieren dabei von der konkreten Hardware, sodass sich ein Software-Ingenieur sofort zuhause fühlt. Dies ist ein akzeptables und logisches Vorhaben. Auch andere Firmen haben solche Anstrengungen unternommen. Gadgeteer passt in die Philosophie einer



NETMF-basierten Rapid-Prototyping-Plattform für Software-Ingenieure. Doch als Software-Ingenieur stehen einem auch schon Linux, Embedded Linux oder Android zur Verfügung. Man hat zwar keine netten kleinen Sockets als Erweiterungsstecker, doch die kann man überall hinzufügen. Und die Palette an Nicht-Gadgeteer-Hardware ist um Größenordnungen vielfältiger. Statt aus wenigen kann man gleich aus hunderten oder gar tausenden Mainboards auswählen, von den vielen netten Smartphones einmal ganz abgesehen. Vielleicht wird Gadgeteer in Zukunft doch noch einige Aufmerksamkeit auf sich ziehen, wenn man z.B. absehen kann, wie das System mit Windows Phone interagiert. Im Moment aber sieht es da etwas mau aus, wenn man die Aktivität in den NETMF- und Gadgeteer-Foren betrachtet. Aktuell tummeln sich da nur wenige Entwickler und beim Nachschauen war da neueste Posting schon einen Monat alt. Die letzte News war sogar vom Juli 2012.

(110738)

Weblinks

[1] Gadgeteer_Webseite: www.netmf.com/gadgeteer

[2] .Net-MF für Elektronik-Ingenieure: www.elektor.de/120033

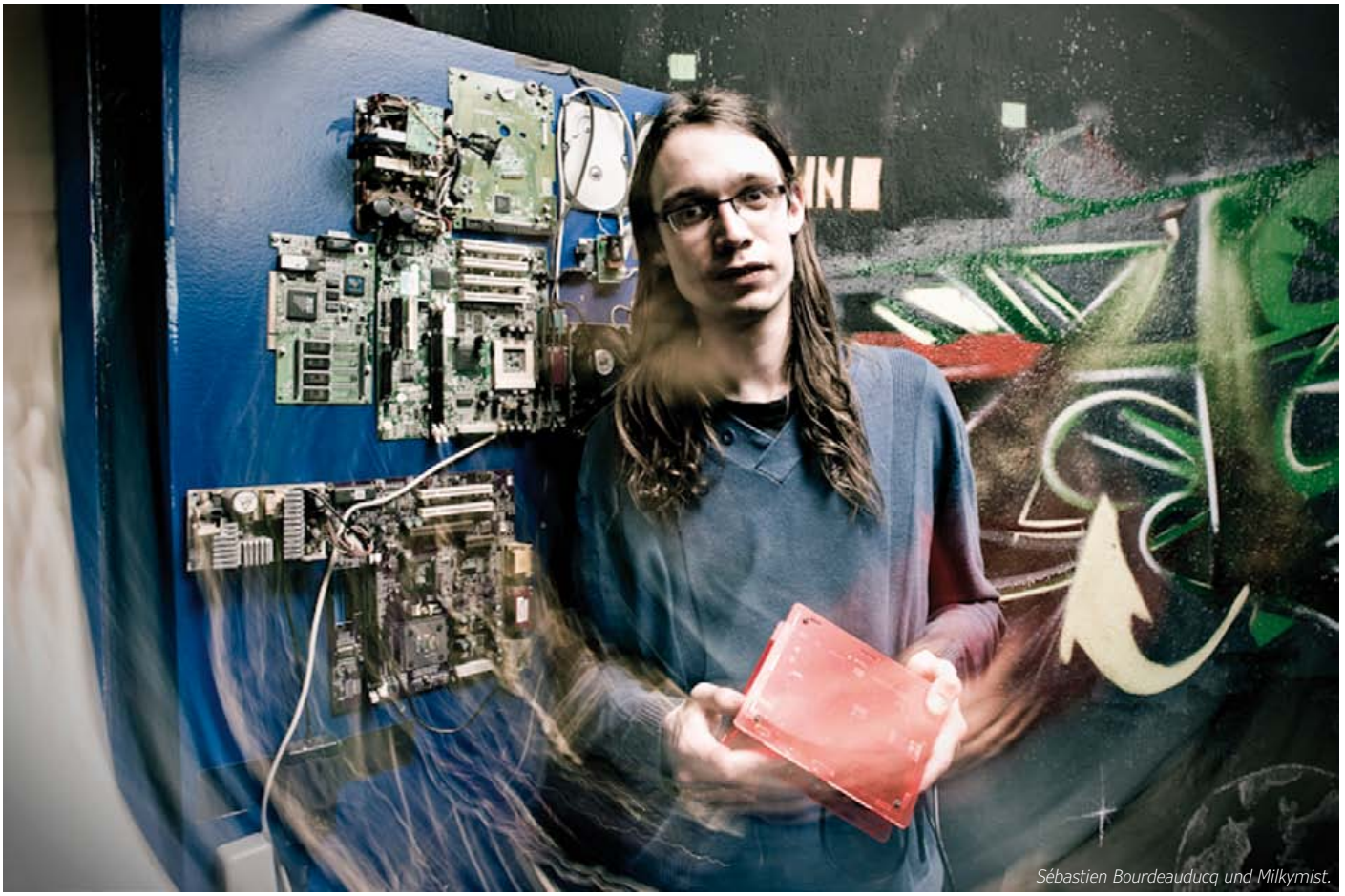
Arduino vs. Gadgeteer

Sowohl Arduino als auch Gadgeteer sind Open-Source-Rapid-Prototyping-Plattformen für Nichtspezialisten. Allerdings gibt es signifikante Unterschiede. Zunächst bei der angepeilten Zielgruppe: Arduino ist für Leute ohne besondere Elektronik- und Programmierkenntnisse gedacht, wogegen sich Gadgeteer an Programmierer ohne Hardware-Erfahrung richtet. Bei Arduino wurde großes Gewicht auf die Vereinfachung der Programmierung gelegt – bei Gadgeteer muss man mit einer typischen IDE zurechtkommen.
Auch bei der Hardware gibt es Unterschiede: Arduino wurde so konzipiert, dass jeder interessierte Amateur damit umgehen oder gar sein eigenes Mainboard bauen kann, da es viele fertige Schaltpläne und Platinen-Layouts dafür gibt. Gadgeteer spezifiziert lediglich, was ein Mainboard können soll – wie es konkret aufgebaut ist, bestimmt der Hersteller. Wichtig ist, dass darauf NETMF läuft.
Ein weiterer interessanter Unterschied betrifft das Paradigma der Programmierung: Auch wenn Arduino nicht damit prahlt, werden die gleichen OO-Techniken wie bei Gadgeteer verwendet. Doch Arduinos Single-Threaded-Polling ist schon etwas anderes als der multi-threaded und event-gesteuerte Ansatz von Gadgeteer. Arduino-Programme, die so genannten *Sketches*, führen eine Endlosschleife aus, in der die Peripherie gesteuert wird. Natürlich kann man trotzdem eine Art Event-Steuerung mit Interrupts realisieren, aber die Anwender nutzen das kaum. Gadgeteer-Programme, die vielsagend *Solutions* genannt werden, sind event-gesteuert. Also keine Schleife. Der Thread des Hauptprogramms kann schlafen, während die Events von Interrupt-Handlern bearbeitet werden. Daher ist Arduino eine gute Wahl für einfache Steuerungsaufgaben ohne komplexe Anwender-Interaktionen. Mit Gadgeteer kann man komplexe, per Menü gesteuerte Applikationen realisieren. Zwischen Arduino und Gadgeteer gibt es also keine Konkurrenz, denn sie ergänzen sich eher.

Tolle Zukunft oder Fehlstart?

Die Wurzeln des NETMF liegen in SPOT (**S**mart **P**ersonal **O**bject **T**echnology) von Microsoft aus dem Jahr 2003. Damit sollten personalisierte Consumer-Elektronik und andere Geräte entwickelt werden. Referenzen zu SPOT zeigen sich auch in NETMF-Programmen. SPOT verwendet „MSN Direct Network Services“, ein UKW-basierter digitaler Dienst, der SPOT-Geräten wie Armband- oder Tischuhren, GPS-Navigationssystemen und sogar Kaffeemaschinen ermöglicht, Informationen aus dem MSN zu beziehen. Bei diesen Informationen handelt es sich um Bezahldienste wie Wetter-Infos, Horoskope, Aktienkurse, Nachrichten, Kalenderereignisse und Sportergebnisse. Auch Kurznachrichten des Windows Live Messenger konnten empfangen werden. Überholt von der technischen Entwicklung wurde dieser MSN-Dienst dann am 1.1.2012 abgeschaltet.
NETMF wird auch von den „Windows SideShow Devices“ genutzt. Windows SideShow, gekoppelt mit der Windows Sidebar (Microsoft Gadgets, eine Verbindung zu Gadgeteer?), ist eine Technologie zur Steuerung verschiedener Geräte. Diese sind an einen Windows-PC angeschlossen, um Informationen und sogar Medien-Daten auch dann anzuzeigen, wenn der PC ausgeschaltet ist. Doch am 10.7.2012 zwang Microsoft die Anwender von Windows Vista und 7 dazu, die Windows Sidebar und die Gadgets zu deaktivieren, da ein paar unsichere Gadgets dazu verwendet werden konnten, Schad-Code auf dem Computer auszuführen...
[*Quelle: Wikipedia*]

Open Source Hardware



Sébastien Bourdeauducq und Milkymist.

Von
Tessel Renzenbrink
(Tech The Future)

Open-Source-Software ist bekanntermaßen ihr Geld wert. Getrieben durch zwei grundsätzliche menschliche Eigenschaften, Neugier und den Willen zur Zusammenarbeit, ist Open-Source der Ausdruck dafür, dass Quellcode allgemein zugänglich sein und weitergereicht, geändert und verbessert werden sollte. Weil Open-Source den Menschen erlaubt, auf den Lösungen anderer aufzubauen, verhindert es Redundanz, senkt Entwicklungskosten, verringert Einstiegsbarrieren und macht generell die Softwareentwicklung zum Vergnügen. Allein die Tatsache, dass mehr als die Hälfte aller Mobiltelefone ein Linux-basiertes Betriebssystem nutzen, zeigt, dass Open-Source-Software einen enormen Stellenwert erreicht hat. Bei all diesem Erfolg ist es keine Überraschung, dass der Open-Source-Gedanke auch die Hardwarewelt erreicht hat, wie das Aufkommen der Open Source Hardware (OSHW) zeigt. OSHW ist Hardware, deren Design allgemein zugänglich gemacht worden ist. Das Design kann verteilt, modifiziert und schließlich beim Bau von Geräten benutzt werden.

Um die Verteilung und Produktion „stromlinienförmiger" zu gestaltet, ist die OSHW-Gemeinschaft an Standardisierung interessiert. Sie entwickelt Kollaborations-Werkzeuge und Designsoftware. Inspiriert von der Open-Source-Software entwirft sie auch Lizenzmodelle, die Rechte und Restriktionen der Benutzer regeln sollen.

Mit den Randbedingungen von OSHW beschäftigt sich Sébastien Bourdeauducq, Begründer eines Open-Source-Projekts namens Milkymist [1]. Er hat auch diverse OSHW-Konferenzen organisiert, zum Beispiel das gerade vergangene Exceptionally Hard & Soft Meeting in Berlin am 28…30. Dezember 2012.

In einem Skype-Gespräch erzählte mir Sébastien

einiges über die Offenlegung von Chip-Designs und wie der Milkymist-Code die Niedrige Erdumlaufbahn erreichte.

Open Source

Tessel: Wie bist du eigentlich zur Open Source Hardware gekommen?

Sébastien: Ich wollte immer wissen, wie die Dinge funktionieren und habe schon als Kind mit Elektronik und Programmieren angefangen. Mein erster echter Kontakt mit „Open-Source" war 1999, als ich ins Linux-Lager wechselte – aber ich habe schon vorher Software anderer Leute in Zeitschriften, Büchern und in BBSs (Bulletin Board Systems) kennengelernt, die nicht ausdrücklich als „Open Source" oder „Free Software" gekennzeichnet war. Ich war von Linux sehr angetan, nachdem ich enttäuscht die Beschränkungen von Windows entdeckt hatte, die mich daran hinderten, am Kern des Betriebssystems herumzubasteln. Mein erstes Projekt, das man als Open-Source bezeichnen könnte, entstand im Jahr 2002 – ein Infrarot-Kommunikationssystem für einen grafischen TI89-Taschenrechner unter der *GNU Free Documentation*-Lizenz, eine ganz simple Sache. Dann habe ich einen Wireless-Treiber für FreeBSD (ein offenes Unix-artiges Betriebssystem) entworfen, der Hard- und Software kombinierte, bis ich verstanden habe, wie die undokumentierte Wireless-Hardware gearbeitet hat. Aber mein größtes aktuelles OSHW-Projekt ist Milkymist, das ich im Jahre 2007 in Angriff nahm (siehe **Kasten**). Ich wollte Chip-Designs zugänglich machen, etwas, das viele Open-Source-Projekte als Ziel haben.

Neuerungen auf der nächsten Ebene

Tessel: Warum möchtest du Chip-Designs offenlegen?

Sébastien: Der eine Grund ist eher philosophischer Natur – fast alle Elektronik, die sich heutzutage Open-Source nennt, ist rund um Black-box-Chips aufgebaut, die man nicht versteht und auch nicht verstehen will. Ich finde dieses Hindernis auf dem Weg der Erkenntnis sehr frustierend.

Ein anderer Grund ist die Sicherheit. Es gibt Hardware-Backdoors, eine solche Backdoor ist eine Manipulation der Firmware eines Chips oder der Hardware, die es nicht authorisierten Personen erlaubt, unter Umgehung der Zugriffssicherung auf den Chip zuzugreifen. Backdoors können auf dem Chip während des Produktionsprozesses eingerichtet werden und stellen - besonders bei militärischer Hardware - eine ernsthafte Bedrohung dar.

Schließlich möchte ich einfach auch auf einer weiteren Ebene innovativ sein. Erfolg hat mein Projekt gerade außerhalb der normalen OSHW-Community. So hat der Milkymist-Code seinen Weg bis in große wissenschaftliche Projekte wie die internationale Raumstation ISS und Teilchenbeschleuniger gefunden

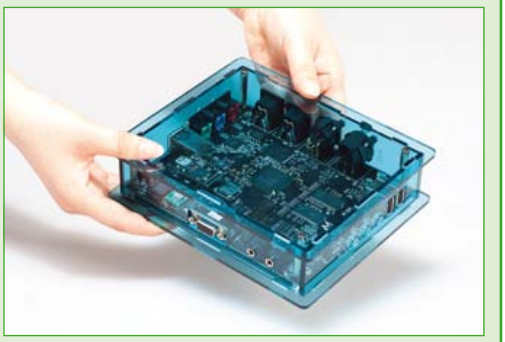
Im Jahre 2009 hat das Jet Propulsion Laboratory (JPL) der NASA den Code des Milkymist SoC Memory Controllers für das Space Communications and Navigation (SCaN) Program [2] verwendet. Am 25.Oktober 2012 hat Sébastien folgenden Brief erhalten:

Dear Sébastien,
Ich möchte die Gelegenheit wahrnehmen, dir für die Open-Source-Arbeit zu danken, die uns bei der Entwicklung unseres JPL Software Defined Radio als Teil des SCaN Testbed half, das nun in der International Space Station installiert wurde. Es wurde in

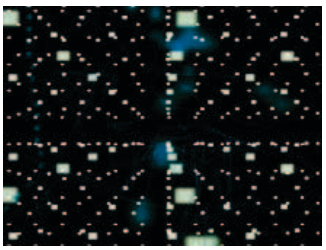
Milkymist

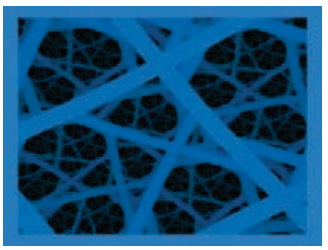
Milkymist One ist ein Stand-alone-Video-Synthesizer, der beim Rendering von Live-Video-Effekten während Musikdarbietungen eingesetzt wurde. Sowohl die Hard- als auch die Software von Milkymist ist – so weit möglich – open-source. Milkymist verwendet ein selbst entwickeltes FPGA-basiertes System-on-Chip (SoC) mit einer Sourcecode-Ergänzung in einer freien Hardware Description Language (HDL). Die meisten Komponenten des SoCs, außer dem LatticeMico32 CPU-Kern, fallen unter die GNU General Public License. Das Programmieren des SoCs und seine Funktion wurde in Elektor 7-8/2011 detailliert beschrieben [6].

Die Video-Rendering-Software Flickernoise ist stark beeinflusst von Milkdrop, einem Open-Source-Plugin für den Mediaplayer Winamp, das „Visualisierungen" erzeugt. Wie bei Open-Source-Projekten üblich haben viele Menschen zu Milkymist beigetragen [1].



Milkymist One Hardware.






der Testphase und für ein paar Monate bei der Inbetriebnahme eingesetzt, inklusive des Dynamic Ram Controllers, den Greg Taylor auf Basis deiner Arbeit gebaut hat. Ich glaube, man kann mit Recht sagen, dass bis heute viele Millionen von Bits deinen Code im All durchlaufen haben.

Thanks again,
James P. Lux, Co-Principal Investigator,
SCaN Testbed

Tessel: Du warst auch an der Open-Hardware-Initiative des CERN, der European Organization of Nuclear Research, beteiligt. Viele Forschungsinstitute gehen mit ihrer Arbeit heimlichtuerisch um, warum begrüßt das CERN die OSHW-Idee?
Sébastien: Natürlich gibt es viele verschiedene Charaktere in den Forschungsinstituten. Es ist richtig, das viele sehr geheimnisvoll tun – manchmal mit Absicht, manchmal, weil sie eine Offenlegung nicht interessiert und manchmal aus kommerziellen Gründen. Ich glaube, Hochenergie-Physiker sind in dieser Beziehung ein wenig offener als die anderen, vielleicht, weil sie Grundlagenforschung betreiben und keine direkt kommerzielle oder militärische Anwendungen entwerfen.
Die Open-Hardware-Initiative des CERN ist auch das Werk einer Handvoll Mitarbeiter, die an OSHW glauben und ihre Institution daran beteiligen wollen. Sie haben das Open Hardware Repository im Netz gegründet, wo Elektronik-Ingenieure an Open-Hardware-Designs, die in der Experimentalphysik gebraucht werden, gemeinsam arbeiten können [3]. Und im Jahre 2011 haben sie die CERN Open Hardware License [4] veröffentlicht.

Exceptionally Hard & Soft Meeting

Tessel: Warum organisierst du das Exceptionally Hard & Soft Meeting (EHSM)?
Sébastien: Vielen Hacker- und OSHW-Konferenzen inklusive der drei, die ich vor dem EHSM [5] mitorganisiert habe, fehlten einige inhaltliche Bereiche. Der Großteil der Hardware dort ist extrem einfach und vermeidet Bereiche, wo es teuer oder schwierig wird. Dies war für mich eine Enttäuschung, die mich veranlasste, das Milkymist-Projekt zu starten. Das EHSM geht da sehr weit voran, zum Beispiel ist unser Hauptredner ein 17-jähriger Junge, der gerne einen Kernfusionsreaktor bauen möchte.
Das EHSM versucht auch, wissenschaftliche Institutionen und Forschungseinrichtungen zu involvieren. Wir haben schon CERN. Wir haben wahrscheinlich Leute aus einem Quantenphysik-Labor und vielleicht auch von einem experimentellen quantenphysikalischen Workshop. Wir sprechen mit einem anderen Institut in Berlin, das sich mit sehr tiefgründiger Materialforschung beschäftigt. Etwas, was wir noch ändern wollen im Vergleich zu anderen Konferenzen: Sie sind zu 99 % männlich.

Tessel: OSHW unterscheidet sich von Open-Source-Software, weil man es nicht nur mit Copyrights, sondern auch mit Patenten zu tun hat. Was denkst du über Patente, beeinträchtigen sie OSHW?
Sébastien: Patente beeinträchtigen alles, einschließlich OSHW. Eines der ursprünglichen Ziele des Patentsystems war es, Erfinder durch einen gesetzlichen Schutz zur Veröffentlichung zu ermutigen, bevor das Patent lizenzfrei wird. Das ist auch eine Art Open-Source-System, denn die Erfindung wird nach Ablauf des Patents zum Allgemeingut. Aber heutzutage ist das Patentsystem nutzlos oder besser gesagt, es ist kläglich gescheitert. Es ist ein Kriegsschauplatz von Konzernen geworden, besonders, wenn Dinge, die keine Innovation darstellen, patentiert werden (siehe Apples berühmtes Patent auf ein Rechteck mit abgerundeten Ecken). Und nur große Konzerne, die gegenseitig ihre Patente verletzen, haben einen Nutzen davon - sie sind sich meist einig, einander nicht zu verklagen. Aber Newcomer sind treffliche Ziele, mit dem Ergebnis, dass das Patentsystem Innovationen unterdrückt statt zu fördern.

Die Zukunft von OSHW

Tessel: Was wären die Vorteile, wenn die Welt ganz auf Open-Source umstiege?

Sébastien: Es wäre ein enormes Bildungsangebot damit verbunden. Aber in ökonomischer Hinsicht weiß ich es nicht so genau. Es ist mir nicht klar, ob offengelegte Innovationen und die Tatsache, dass man viel leichter mit anderen Leuten zusammenarbeiten kann, den finanziellen Nutzen überwiegen, den man aus der Geheimhaltung seiner Designdetails ziehen kann. Aber ich liebe das Experimentieren, und das ist der Grund, warum ich solche Projekte wie Milkymist ins Leben rufe. Positiv ist, dass die OSHW-Herangehensweise talentierte Leute anzieht.

Tessel: Welche Rolle wird OSHW in der Zukunft spielen? Wird sie geschützte Hardware ersetzen?

Sébastien: Ich kann das für die nahe Zukunft nicht erkennen. Die Realität ist, dass die meiste Hardware von Konzernen wie Apple oder Intel produziert wird, die sich nicht einen Deut um OSHW kümmern und auch auf einem Berg von Know-how sitzen, den sie eifersüchtig bewachen. Und die meisten OSHW-Leute, nicht zuletzt die, die nicht am EHSM teilnehmen, sind nicht an der Produktion von Elektronik auf Industrielevel interessiert. Längerfristig wird die Situation vielleicht besser, aber dazu muss sich erst die Mentalität in vielen Köpfen ändern.

(120643)

Weblinks

[1] http://milkymist.org/

[2] http://spaceflightsystems.grc.nasa.gov/SOPO/SCO/SCaNTestbed/

[3] www.ohwr.org/

[4] www.ohwr.org/projects/cernohl/wiki

[5] http://ehsm.eu/

[6] www.elektor-magazine.de/110447

Das Tonbandgerät Nagra IV

Reporterstolz von 1968

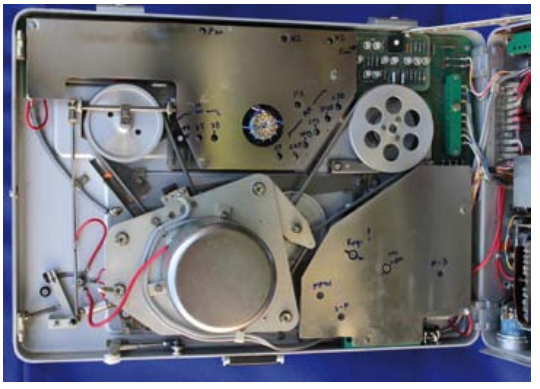
1

Von Peter Beil (D)

Auch wenn es sich um eines der berühmtesten Tonbandgeräte der Welt handeln dürfte, wird es doch nur wenigen Elektor-Lesern ein Begriff sein: Das Nagra in **Bild 1** ist ein Profi-Gerät, das speziell für portable Anwendungen in Funk und Fernsehen entwickelt wurde, wobei es sogar die bild-synchrone Tonaufzeichnung für Film und TV beherrschte.
Der gebürtige Pole Stefan Kudelski als geistiger Vater des Geräts sorgte für die besten damals machbaren technischen Eigenschaften. Im Inneren beherbergte der Apparat Transistoren aus den USA, Potis aus England, Schalter aus der Schweiz, Kondensatoren aus Holland und Steckverbindungen aus Deutschland. Der Name „Nagra“ ist von einem polnischen Wort für Aufzeichnung abgeleitet. Dank seiner Qualität und Zuverlässigkeit wurde das Nagra zu einer Legende. Überall auf der Welt - vom eiskalten Alaska bis zur sengenden Hitze der Sahara - wurde es eingesetzt.
Die Geschichte portabler Bandgeräte begann in der 1950er Jahren mit Modellen, die von Federwerken angetrieben wurden. Das elektrisch angetriebene Nagra III erschien zu Beginn der 1960er Jahre. Am Ende dieser Dekade kam das Modell Nagra IV auf den Markt, um das es in diesem Beitrag geht.
Das Nagra wurde von zwölf Monozellen (engl. „D cell“, IEC R20) versorgt, die schon ein Drittel des Gewichts der Maschine ausmachten (**Bild 2**). Diese unstabilisierte Versorgung war sowohl für den Bandantrieb als auch für den Wiedergabeverstärker zuständig. Letzterer lieferte 1,5 W für einen erstaunlich klaren Klang in ordentlicher Lautstärke. Solange man es mit dem Vor- und Zurückspulen sowie der Lautstärke nicht übertrieb, reichten die

2

3



Retronik ist eine monatliche Rubrik, die antiker Elektronik und legendären ELEKTOR-Schaltungen ihre Referenz erweist. Beiträge, Vorschläge und Anfragen telegraphieren Sie bitte an editor@elektor.com.

4

5

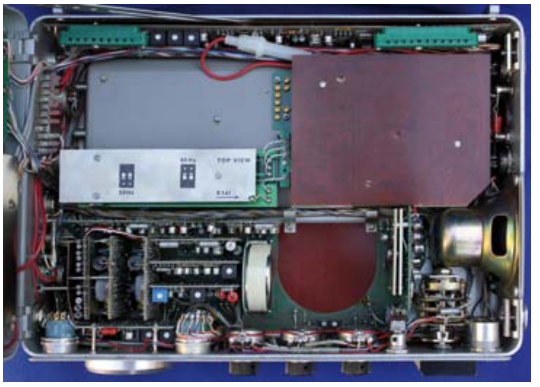
Batterien recht lang. Später gab es dann sogar Akkus in diesem Format, was bedeutete, dass man locker 10 kg Gewicht mit sich herumtragen musste. Intern war eine Referenzspannung von –10 V für alle mit der Modulation zusammenhängenden Schaltungsteile vorgesehen. Wenn Sie sich über das Vorzeichen dieser Spannung wundern sollten, dann sei soviel verraten: Hier war das Massepotential positiv! Vermutlich hing dies teilweise mit der anfänglichen Verwendung von Germanium-Transistoren zusammen. Später wurde dies dann aus Gründen der Rückwärtskompatibilität nicht mehr geändert, was der Zubehörindustrie noch ordentliche Kopfschmerzen bescheren sollte. Eine simple Verbindung mit einem anderen Gerät führte unter Umständen nämlich zu einem Kurzschluss, da die meisten Geräte negatives Massepotential aufwiesen. Schon das Einstecken eines einfachen Audiokabels konnte problematisch werden, wenn die Abschirmung an beiden Enden angeschlossen war.

Elektromechanisch gab es keine Besonderheiten. Der voll gekapselte Capstan-Motor ermöglichte drei elektronisch geregelte Geschwindigkeiten von 9,5, 19 und 38 cm/s. Eine ausgeklügelte Antriebs- und Friktionsmechanik erlaubte den Antrieb beider Spulen mit einem einzigen Motor (**Bild 3**). Bei vollen Batterien konnte man damit richtig schnell zurückspulen. Vorspulen war mit doppelter Geschwindigkeit möglich. Spezielle Öle und Fette, die man über die Servicestellen beziehen konnte, ermöglichten den Betrieb im Temperaturbereich zwischen –20 und +50 °C.

Abgebildet ist übrigens das Modell Nagra 4.2 aus den 1970er Jahren, das optisch immer noch den ersten Maschinen gleicht und den Gipfel der technischen Entwicklung dieser Serie markiert. Auf der Frontplatte sitzen rechts neben den selbsterklärenden Bedienungselementen zwei merkwürdige Anzeigen (**Bild 4**). Diese dienen zur Überwachung von Betriebsparametern und benötigen nur den Bruchteil der Leistung von Indikatorlämpchen.

Die komplette Elektronik besteht der Zeit entsprechend aus diskreten Bauelementen. An guten Audio-ICs wurde damals noch herumentwickelt, und die ersten Exemplare waren nicht gerade für Zuverlässigkeit, Verzerrungs- und Rauscharmut bekannt. Einige Jahre später gab es dann ein Modell mit ICs. Bei Ein- und Ausgängen wurden Symmetrierübertrager eingesetzt. Die internen Verstärkerstufen waren asymmetrisch aufgebaut. Ungewöhnlich war der potentialfreie symmet-

6



7

8

9

10

rische Ausgang mit einer Amplitude von 4,4 V – dem Pegel, der damals von Telekomunternehmen für Übertragungsleitungen gefordert war (**Bild 5**).

Die Elektronik wurde modular konzipiert, was damals einer Pioniertat gleich kam. Auf einem Motherboard waren die Module angeordnet. Alles was gesteckt wurde hatte Goldkontakte (**Bild 6**).

Falls das Auge des Elektronikers angesichts des unpassenden weißen Sicherungshalters oben zusammenzuckt: Das ist kein originales Bauteil, sondern nachgerüstet. Herr Kudelski hatte nämlich bei aller Genialität eines übersehen: die Sicherung...

Dafür wurde sehr große Sorgfalt auf das Anzeigeinstrument gelegt. Hier gab es nämlich sehr viele Normen und Konventionen. Das „Modulometer" konnte diese Anforderungen perfekt erfüllen, da die Zeigerbewegung zur Neutralisierung der Trägheit der Anzeige beschleunigt wurde (**Bild 7**).

Messtechnisch waren alle drei Geschwindigkeiten individuell bezüglich Frequenzgang einstellbar, getrennt für Standard- und rauscharme Verzerrungskorrektur. Sogar die HF-Symmetrie und die Klirrfaktoren H3 und H4 konnten optimiert werden. Abhängig vom Gerät wurde ein Frequenzgang von 20 Hz bis hin zu 18 kHz von –2 dB erreicht. Dies alles bei einem nicht gewichteten Signal/Rausch-Verhältnis von 60 dB, was durchaus im Bereich einer Studio-Bandmaschine lag.

Es sind zwei Mikrofoneingänge und ein zusätzlicher konfigurierbarer Eingang vorhanden (**Bild 8**). Die Mikrofoneingänge wurden mit Eingangstrafos oder elektronischen symmetrischen Eingängen realisiert. In den frühesten Modellen musste der Eingangsverstärker passend zu den Mikrofonen bestückt werden: entweder für dynamische Mikrofone mit niedriger oder hoher Impedanz oder aber für Kondensatormikrofone etc. **(Bild 9**). Später konnte der Eingangsverstärker dann umgeschaltet werden.

Die Eingänge sind in weiten Grenzen einstellbar, so dass der interne Abschwächer kaum je umgeschaltet werden musste. In Verbindung mit einem steckbaren Limiter und einem Spitzenwertbegrenzer mit Knie-Funktion ist es kaum möglich, eine Aufnahme zu übersteuern. Außerdem existiert ja noch eine magnetische Bandsättigung, die ebenfalls Spitzen verdauen kann.

Ein Problem gab es mit der zwar antiquierten aber doch weit verbreiteten Phantomspeisung mit 48 V. Damals war es technisch unmöglich, saubere 48 V aus einer 18-V-Versorgung zu generieren. Natürlich kannten die Nagra-Entwickler die damaligen DC/DC-Konverter, doch ihre Kurvenformen hätten Einstreuungen ins Audiosignal zur Folge gehabt. Die Konverter wären bei der Modulbauweise ja am empfindlichsten Punkt der Elektronik platziert: am Mikrofoneingang.

Hier half die Firma Sennheiser, in dem sie revolutionäre Kondensator-Mikrofone für niedrige Betriebsspannungen auf den Markt brachte. Der Wahlschalter hat rechts von der 12-V-Phantomspeisung noch eine Stellung mit einem „T" für 10 V. Das „T" steht hier für Tonaderspeisung [1] – Polen ist eben ein Nachbarland von Deutschland. Vermutlich liegt hier ein weiterer Grund für die schon erwähnte interne 10-V-Referenz. Aus unbekannten Gründen verschwand die 12-V-Phantomspeisung von der Bildfläche, während die T-Version lange recht verbreitet war. Unglücklicherweise nicht ohne Nebenwirkungen.

Wie der Name schon nahelegt, wird bei diesem Verfahren die Versorgungsspannung über die symmetrischen Audioleitungen angelegt, was tatsächlich keine Interferenz mit dem Nutzsignal ergibt. Allerdings vertragen DC-gekoppelte Eingangsstufen oft keinen Gleichspannungspegel am Eingang. Die Spannungseingänge sind deshalb durch zwei Kondensatoren vor Gleichspannungen geschützt. Allerdings bilden dann Widerstand und Kondensator einen Filter, was als Kompromiss angesehen werden muss, wenn man einen weiten Frequenzbereich erreichen will. Und wenn versehentlich ein dynamisches Mikrofon angeschlossen wird, dann sind die Bässe verschwunden, da der fließende Strom die Spule stark auslenkt. Doch irgendwie spielten diese kleinen Probleme wohl keine große Rolle.

Zurück zur Bandmaschine: Auf der Oberseite sieht man zwischen den beiden Köpfen für Aufnahme und Wiedergabe einen weiteren Tonkopf. Dieser ist das zentrale Element für alle Synchronisationserfordernisse: der Kopf für Pilottöne (**Bild 10**).

Um die Sache auf die Spitze zu treiben: Zur Vermeidung jeglicher Synchronisationsprobleme im Studiobetrieb (wie z.B. Drift) gab es speziel-

les Bandmaterial (SEPMAG) mit Perforationen, die exakt denen von Filmmaterial glichen [2] (**Bild 11**). Spezielle Film- und Sound-Follower-Maschinen waren von so genannten Synchronmotoren angetrieben, deren Drehzahl nicht von der Spannung, sondern der Frequenz abhing. Folglich war deren Drehzahl identisch und fest an die Netzfrequenz gekoppelt.

Die meistens von Batterien versorgten Filmkameras enthielten einen kleinen Frequenzgenerator, der für nominale 24 Bilder/s (später 25 Bilder/s) eine Frequenz von exakt 50 Hz lieferte. Wenn sich die Bildfrequenz der Kamera aufgrund vieler unterschiedlicher mechanischer Einflüsse änderte, dann veränderte sich auch diese Frequenz mit. Diese Frequenz wurde in der Mitte der Tonspur mit dem schon erwähnten Pilottonkopf aufgezeichnet. Um entsprechendes Brummen in der Tonspur zu vermeiden, war der Spalt dieses Zusatzkopfes um 90° gedreht, sodass der Pilotton im Tonsignal nahezu komplett unterdrückt wurde.

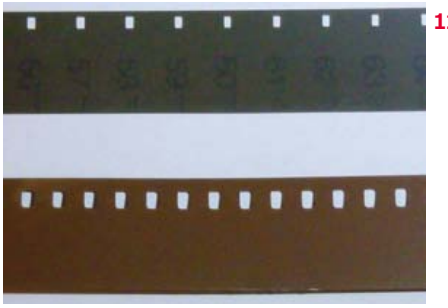
Wenn eine Tonaufnahme zu einer mit Netzfrequenz gesteuerten Sound-Follower-Maschine übertragen wurde, dann wurde der Pilotton mit der Netzfrequenz verglichen und damit entweder die Abspielmaschine nachgesteuert, die in unserem Fall die Bezeichnung Nagra trägt, oder aber das Aufzeichnungsgerät. Zu diesem Zweck enthielt eine Nagra-Maschine einen so genannten Synchronizer, der die Geschwindigkeit um ±4 % korrigieren konnte. Für größere Abweichungen gab es Zubehör, was aber nur in extrem seltenen Fällen nötig war.

Eine weitere zu diesem Thema passende Funktion nannte sich „Playback Procedure“, die bei Gesangs- oder Musik-Playbackaufnahmen zum Einsatz kam. Hierzu wurde vor der eigentlichen Aufnahme die Netzfrequenz als Pilotton aufgezeichnet. Dann wurde der Pilotton der Kamera empfangen und mit dem aufgezeichneten Pilotton verglichen, womit dann der Synchronizer die Differenz ausregeln konnte. Dieses Verfahren wurde Jahrzehnte lang genutzt.

Das Interface in Form diverser Steckverbinder war ein wahrhaft offenes System. Dementsprechend gab es unzähliges Zubehör; von großen 900-m-Spulen über Mischpulte und Quarz-Steuerungen bis hin zu Funklösungen. Der anfängliche Preis war für diese Zeit sicherlich extrem hoch. Eine voll ausgebaute Version (ohne Transportkoffer oder Netzteil) kostete den heutigen Gegenwert von gut 10.000 $.

Drei besondere Gadgets sollen nicht unerwähnt bleiben: Das in eine selbst ausrichtende Rolle eingebaute Bandlängenzählwerk, eine Stroboskoprolle zum Abgleich der Geschwindigkeit und einen mechanischen Bandreiniger in Form eines Metallblechs (**Bilder 12** und **13**). Für eine Maschine dieses Kalibers gab es selbstverständlich auch passende Kopfhörer. Damals lieferte die Firma Beyer Dynamic die Referenz in Form des linearen „Normal-Telefons“ mit 2 x 25 Ω. In modifizierter Form blieb dieser Kopfhörer für mehr als 50 Jahre aktuell!

Das Bandgerät zeichnete „natürlich“ monofon auf (siehe **Kasten**). Später folgte dann auch noch eine Stereo-Version, und in den frühen Tagen der Videotechnik gab es sogar eine Variante mit Time Code. Ende der 1980er Jahre war die Digitaltechnik noch lange nicht ausgereift, doch 1992 erschien eine digitale Version, die vier PCM-Spuren auf Standard-Bandmaterial aufzeichnen konnte – mit der damals unglaublichen Auflösung von 24 bit! Die technische Entwicklung machte aber nicht plötzlich halt und von daher gibt es jetzt auch ein Nagra mit Flash-Speicher und all den Dingen, die für eine moderne digitale Tonaufzeichnung wichtig sind!


11


12


13

(120570)

Weblinks

[1] www.ips.org.uk/faq/index.php?title=Tonader_Power

[2] http://en.wikipedia.org/wiki/Sound_follower

Warum mono?

Falls Sie sich wundern, warum überhaupt jemand monofon aufzeichnen möchte, hier die Erklärung: Wenn man Sprache authentisch mit einem Stereo-Mikrofon oder zwei räumlich separaten Mikrofonen aufzeichnen will, dann muss die Tonquelle an einem bestimmten Punkt im Raum positioniert sein. Wenn sich wie üblich in einem Film die Position des Mikrofons oder des Sprechers ändert, kommt es zu Effekten, für die das menschliche Gehör sehr empfindlich ist. Spätere Bearbeitungen solcher Aufnahmen können recht irritierend wirken, denn die Klangunterschiede lassen sich nicht mehr aus dem Signal herausfiltern. Ein Monosignal enthält dagegen keine solchen räumlichen Anteile, bis es bei der Abmischung virtuell positioniert wird. Im Endprodukt ist die Sprache dann zusammen mit vielkanaligen Soundeffekten räumlich aufgezeichnet.

Hexadoku Sudoku für Elektroniker

Frühling... Zeit für eine kleine Putzaktion und ein bisschen Handwerkerei im Haus. Danach sollten Sie sich mit etwas Entspannendem belohnen. Wie wäre es zum Beispiel mit diesem Rätsel? Bei unserem monatlichen Hexadoku sind wie immer die richtigen Zahlen in den grauen Kästchen herauszufinden. Wer uns diese einsendet, hat die Chance auf einen schönen Elektor-Gutschein!

Die Regeln dieses Rätsels sind ganz einfach zu verstehen: Bei einem Hexadoku werden die Hexadezimalzahlen 0 bis F verwendet, was für Elektroniker und Programmierer ja durchaus passend ist. Füllen Sie das Diagramm mit seinen 16 x 16 Kästchen so aus, dass alle Hexadezimalzahlen von 0 bis F (also 0 bis 9 und A bis F) in jeder Reihe, jeder Spalte und in jedem Fach mit 4 x 4 Kästchen (markiert durch die dickeren schwarzen Linien) **genau einmal** vorkommen. Einige Zahlen sind bereits eingetragen, was die Ausgangssituation des Rätsels bestimmt.
Wer das Rätsel löst - sprich die Zahlen in den grauen Kästchen herausfindet - kann wie jeden Monat einen Hauptpreis oder einen von drei Trostpreisen gewinnen!

Mitmachen und gewinnen!

Unter allen internationalen Einsendern mit der richtigen Lösung verlosen wir

Einen **ELEKTOR-Gutschein im Wert von 100 €**

und drei **ELEKTOR-Gutscheine im Wert von je 50 €**.

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:
Elektor – Redaktion – Süsterfeldstr. 25 – 52072 Aachen
Fax: 0241 / 88 909-77 E-Mail: hexadoku@elektor.de
Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 28. Februar 2013!

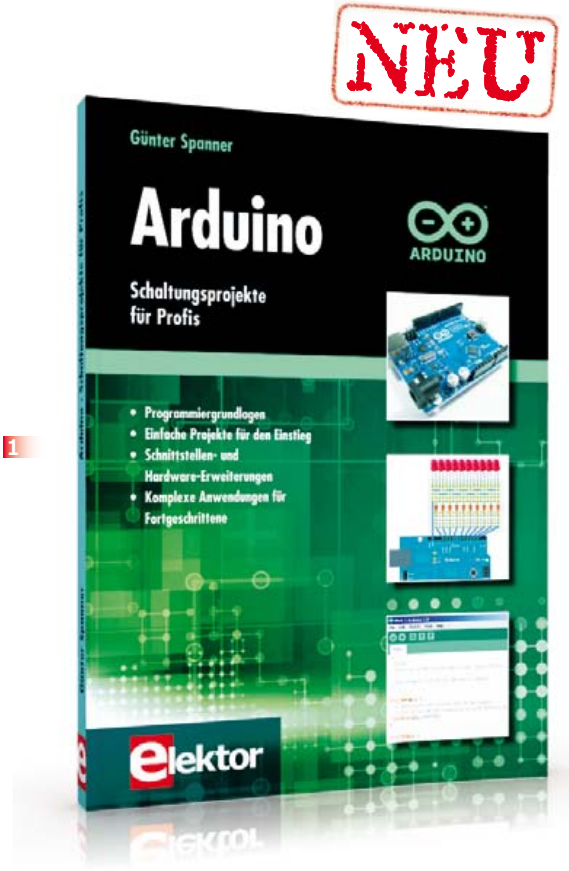
Die Gewinner des Hexadokus aus dem Novemberheft stehen fest!

Die richtige Lösung ist: **BD18A**.
Der Elektor-Gutschein über 100 € geht an: Ron Ware aus Witley (Großbritannien).
Einen Elektor-Gutschein über je 50 € haben gewonnen: Panagiotis S. Krokidis, Didier Pelegri und P. Scheepers.
Herzlichen Glückwunsch!

F	A		4								9		E	6	
9			E	6		1			0		3	F			C
		B		3		4			C		F		7		
0	7				B	F			5	6				2	A
	C	F		2	4					D	8		E	0	
			9	5		D			7		C	2			
	2	5	3		A		E	9		F		C	4	D	
						C	B	A	2						
						9	7	E	F						
	E	6	C		2		3	0		7		D	1	A	
			5	4		B			D		2	7			
	F	8		A	5					1	4		9	3	
5	1				C	3			A	0				B	E
		7		9		A			E		1		6		
D			F	1		E			3		B	4			0
E	3		0									A		7	1

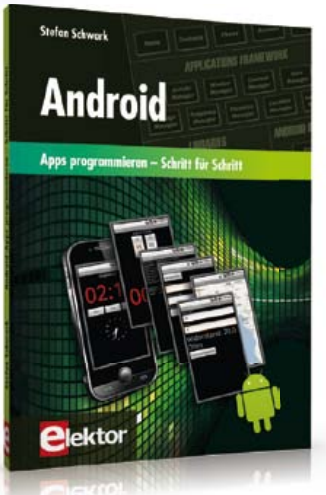
0	E	3	A	5	6	F	9	D	1	4	B	2	7	8	C
1	D	2	9	A	0	B	7	C	F	E	8	5	3	4	6
7	B	F	4	C	1	2	8	9	3	5	6	A	D	E	0
5	8	6	C	D	E	3	4	2	0	7	A	9	B	F	1
D	9	C	3	6	2	E	1	7	8	F	4	B	A	0	5
B	F	E	0	3	4	5	D	6	9	A	2	7	1	C	8
4	5	7	1	F	C	8	A	E	B	D	0	3	9	6	2
2	6	A	8	7	9	0	B	1	5	3	C	E	4	D	F
6	7	1	B	8	A	D	5	0	2	9	E	F	C	3	4
E	A	4	D	9	B	1	F	8	6	C	3	0	5	2	7
3	0	8	2	E	7	4	C	F	A	1	5	D	6	9	B
F	C	9	5	0	3	6	2	4	7	B	D	1	8	A	E
A	2	0	F	B	D	7	6	3	4	8	1	C	E	5	9
9	1	5	E	4	8	C	0	A	D	2	7	6	F	B	3
8	3	B	6	1	F	A	E	5	C	0	9	4	2	7	D
C	4	D	7	2	5	9	3	B	E	6	F	8	0	1	A

Elektor Store



1

2



3

Schaltungsprojekte für Profis

1 Arduino

Für den großen Erfolg der Arduino-Plattform lassen sich zwei Ursachen finden. Zum einen wird durch das fertige Board der Einstieg in die Hardware enorm erleichtert; der zweite Erfolgsfaktor ist die kostenlos verfügbare Programmieroberfläche. Unterstützt wird der Arduino-Anwender durch eine Fülle von Software-Bibliotheken. Die täglich wachsende Flut von Libraries stellt den Einsteiger vor erste Probleme. Nach einfachen Einführungsbeispielen ist der weitere Weg nicht mehr klar erkennbar, weil oft detaillierte Projektbeschreibungen fehlen. Hier setzt dieses Buch an. Systematisch werden Projekte vorgestellt, die in verschiedene Themengebiete einführen. Dabei wird neben den erforderlichen theoretischen Grundlagen stets größter Wert auf eine praxisorientierte Ausrichtung gelegt.

270 Seiten (kart.) • ISBN 978-3-89576-257-4
€ 39,80 • CHF 49,40

Bestückte und getestete Platine inkl. Gehäuse

2 USB-Isolator

Masseschleifen zwischen PCs und Geräten, die über USB miteinander kommunizieren, sind oft Ursache hoher Störpegel auf den Signalen. Der USB-Isolator ist der Retter in der Not, wenn die Systeme in störintensiver Umgebung arbeiten müssen. Er trennt die Systeme galvanisch voneinander und schlägt gleichzeitig eine Brücke, sowohl für die Daten- als auch die Betriebsspannungsleitungen.

Art.-Nr. 120291-91 • € 69,95 • CHF 86,80

Apps programmieren – Schritt für Schritt

3 Android

Smartphones und Tablet-Computer mit dem Betriebssystem Android finden immer weitere Verbreitung. Die Anzahl der Anwendungsprogramme – die sogenannten Applikationen oder kurz Apps – mit denen sich die Geräte individuell an die Vorlieben und Wünsche ihrer Benutzer anpassen lassen, steigt täglich an. Man ist bei der Individualisierung seines Smartphones aber nicht auf fix und fertige Applikationen beschränkt. Es ist einfacher als man denkt, Android-Geräte selber zu programmieren und eigene Apps zu schreiben. Dieses Buch bietet eine Einführung in die Programmierung von Apps auf Android-Geräten. Es erklärt leicht nachvollziehbar die Funktionsweise des Android-Systems und Schritt für Schritt die Programmierung von Applikationen.

256 Seiten (kart.) • ISBN 978-3-89576-252-9
€ 34,80 • CHF 43,20

Bestücke und getestete Platine

4 Elektor-Linux-Board

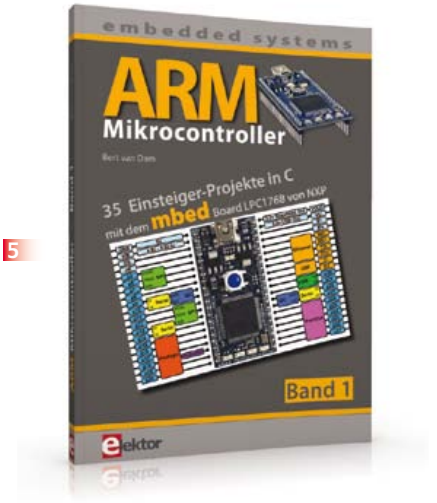
Linux läuft heutzutage auf den unterschiedlichsten Geräten – sogar in Kaffeemaschinen. Es gibt daher viele Elektroniker, die an Linux als Basis für eigene Controller-Projekte interessiert sind. Eine Hürde ist jedoch die scheinbar hohe Komplexität, außerdem sind Entwicklungsboards oft recht teuer. Mit diesem kompakten Modul, das bereits für modernste Embedded-Projekte fertig bestückt ausgestattet ist, gelingt der Linux-Einstieg ideal und preiswert zugleich.

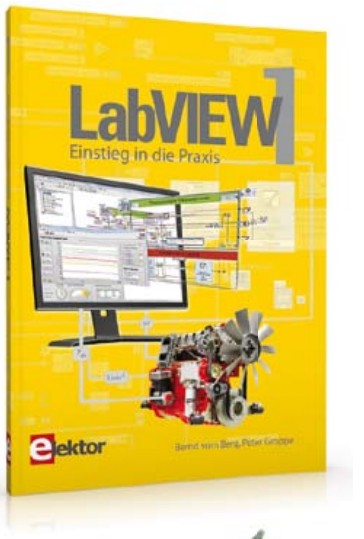
Art.-Nr. 120026-91 • € 64,95 • CHF 80,60

Band 1: 35 Einsteiger-Projekte in C

5 ARM-Mikrocontroller

Die Projekte in diesem Buch sind für Einsteiger in C und ARM-Mikrocontroller ausgelegt. Das heißt nicht, dass diese Projekte einfach sind. Sie sind aber einfach zu verstehen. Es wird beispielsweise die USB-Verbindung zur Kommunikation benutzt, eine Methode, die im mbed-Board so einfach integriert ist, dass sie sich auch für ein Einsteiger-Buch eignet. Der mbed NXP LPC1768 nutzt Cloud-Technologie, ein revolutionäres Konzept in der Software-Entwicklung. Es bedeutet, dass man keinerlei Software auf seinem PC installieren muss, um den mbed zu pro-





grammieren. Das Einzige, was Sie brauchen, ist ein Webbrowser mit Internetzugang und einen freien USB-Anschluss an Ihrem PC.

261 Seiten (kart.) • ISBN 978-3-89576-262-8
€ 39,80 • CHF 49,40

Einstieg in die Praxis
6 LabVIEW 1

Das LabVIEW-Programmpaket ist ein international anerkannter Standard zur Entwicklung und Gestaltung von Messgeräten und Prozesssteueroberflächen. Seine Universalität konfrontiert den LabVIEW-Einsteiger allerdings mit einer unübersichtlichen Vielfalt von Funktionen, die er ohne fundierte Anleitung kaum überblicken kann. Hier setzt diese neue mehrteilige Lehrbuchreihe an: Von Grund auf werden in einfach nachvollziehbaren Schritten der Aufbau, die Struktur und die Verwendung von LabVIEW erklärt, in praktischen Beispielen dargestellt und mit Übungen vertieft. Der erste Band erläutert die Grunddatentypen und die zugehörigen numerischen Grundfunktionen ebenso ausführlich wie die elementaren Programmstrukturen.

240 Seiten (kart.) • ISBN 978-3-89576-253-6
€ 34,80 • CHF 43,20

Bausatz mit allen Bauteilen & Platine(n)
7 TAPIR – E-Smog-Detektor

Der TAPIR (Totally Archaic but Practical Interceptor of Radiation) spürt "strahlendes Missverhalten" elektronischer Geräte in Ihrer Umgebung auf. Trotz der einfachen Schaltung handelt es sich hierbei um einen ultrasensitiven E-Smog-Detektor, der jede Quelle eines elektrischen oder – mit einer entsprechenden Antenne versehen – magnetischen Feldes aufspürt und dies akustisch signalisiert.

Art.-Nr. 120354-71 • € 14,95 • CHF 18,60

Theorie und Praxis
mit WinFACT und Multisim
8 Regelungstechnik

Die heutige Regelungstechnik hat Verknüpfungspunkte mit fast jedem technischen Gebiet. Ihre Anwendungen reichen von der Elektrotechnik über die Antriebstechnik und den Maschinenbau bis hin zur Verfahrenstechnik. Will man nun die Regelungstechnik anhand der fachlichen Regeln dieser einzelnen Gebiete erklären, so müsste man von einem Regelungstechniker verlangen, jedes Fachgebiet, in dem er Regelungen vornehmen will, fundiert zu beherrschen. Dies ist aber bei dem heutigen Stand der Technik nicht möglich. Bei der Regelung einer Antriebsaufgabe, einer Druck- oder einer Temperaturregelung tauchen Gemeinsamkeiten auf, die man mit einer einheitlichen Vorgehensweise beschreiben kann. Die Grundgesetze der Regelungstechnik gelten in gleicher Weise für alle Regelkreise, ganz unabhängig davon, wie verschieden sie im Einzelnen auch apparativ aufgebaut sein mögen. Dieses Buch richtet sich an den Praktiker, der gründlicher in die Regelungstechnik eindringen möchte, auf ausschweifende theoretische Exkursionen in die Mathematik aber gerne verzichten kann.

365 Seiten (kart.) • ISBN 978-3-89576-240-6
€ 49,00 • CHF 60,80

Weitere Informationen zu unseren Produkten sowie das gesamte Verlagssortiment finden Sie auf der Elektor-Website:

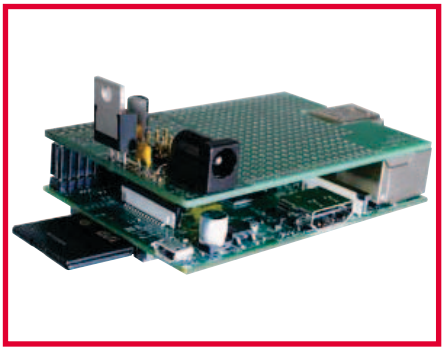
www.elektor.de/shop

Elektor-Verlag GmbH
Süsterfeldstr. 25
52072 Aachen
Tel. +49 (0)241 88 909-0
Fax +49 (0)241 88 909-77
E-Mail: bestellung@elektor.de

Nächsten Monat in Elektor

Intelligentes LCR-Meter

Bereits 1997 veröffentlichte Elektor ein High-Tech-LCR-Meter, die Eigenschaften standen den professionellen Erzeugnissen der Industrie in jener Zeit nicht nach. In der nächsten Ausgabe präsentieren wir die Neuentwicklung eines LCR-Meters, das den inzwischen rund 16 Jahre alten Vorgänger weit hinter sich lässt. Das neue LCR-Meter ist up-to-date und top-fit für den Einsatz. Es kann nicht nur stand-alone, also unabhängig arbeiten, über USB ist es auch an einen PC anschließbar.



Raspberry Pi Protoboard

Raspberry Pi heißt ein kleines, kostengünstiges Computer-Entwicklungssystem im Format einer Kreditkarte. Anschließbar ist eine ganze Palette von Peripherie, dazu gehören auch ein Bildschirm und eine Tastatur. Die Entwicklung von Prototypen wird durch ein ergänzendes Prototyping-Board erleichtert, auf dem eine stabilisierte Stromversorgung und ein Lötpunktraster-Feld zum Probieren und Experimentieren ihren Platz haben. Das Prototyping-Board kann auf Raspberry Pi aufgesteckt werden.

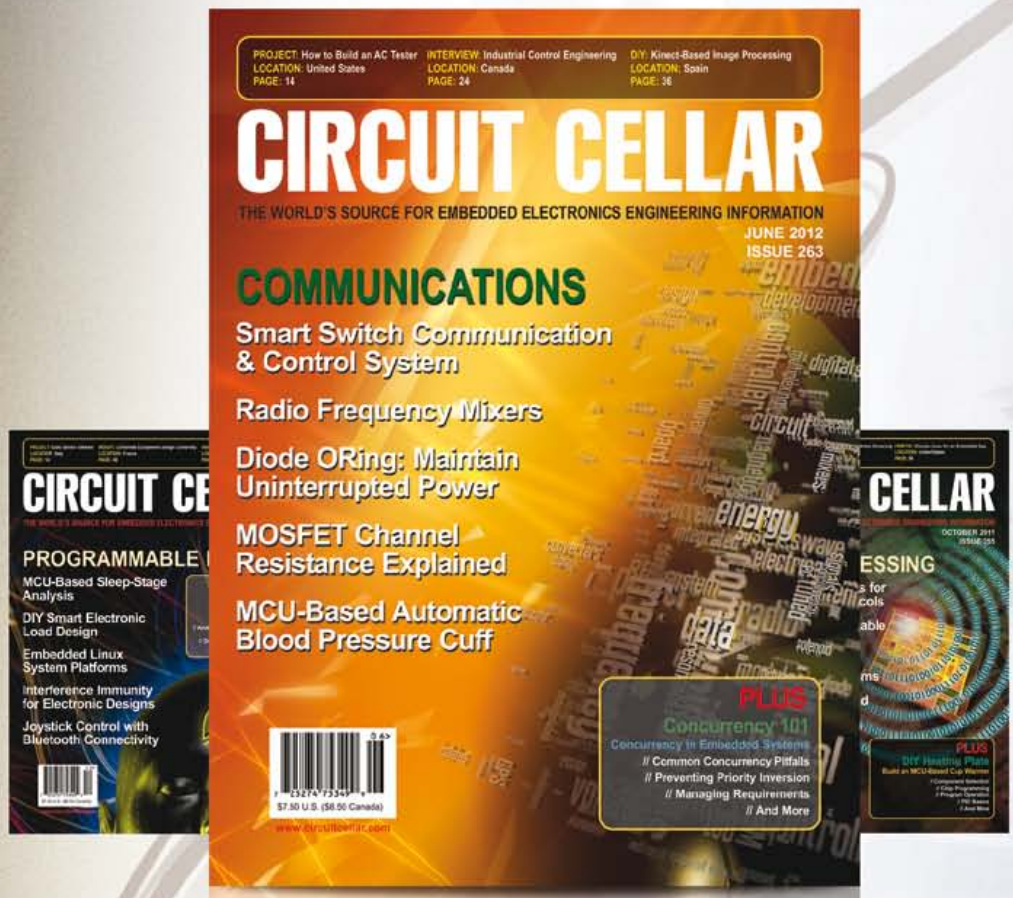
Thermobuch

Noch einmal ein Temperaturmesser? Ja und Nein. Hier ist nicht nur die Funktionalität ausgeweitet, dieses Selbstbauprojekt überrascht auch durch sein nicht alltägliches Gewand. Das Buch, das eher der Kategorie der seltenen Literaturarten zuzurechnen ist, gibt wahlweise Auskunft über die Temperatur oder die Luftfeuchte. Es kann die Anzeige in festem Rhythmus wechseln, im wortwörtlichem „Handbetrieb" reagiert es aber auch auf Händeklatschen.

Elektor März 2013 erscheint am 27. Februar 2013. **Änderungen vorbehalten!**

SPECIAL: SAVE 50% SPECIAL: SAVE 50% SPECIAL

Celebrate Circuit Cellar's 25th Anniversary



$25 Print or Digital :: $50 Combo

Celebrate *Circuit Cellar*'s 25th year of bringing readers insightful analysis of embedded electronics technology.

Visit www.circuitcellar.com/el912 to take advantage of these great deals.

BONUS OFFER! BONUS OFFER! BONUS OFFER! BONUS OFFER!

Sign up today and you'll also receive the **Special 25th Anniversary Edition** with your subscription!

CIRCUIT CELLAR 25 YEARS OF EMBEDDED INSIGHT

reichelt.de
elektronik

schnelle Lieferung
faire Preise
Top-Service
40 Jahre Erfahrung

Kundenbewertungen:
97.63% zufriedene Kunden
4.88 / 5.00
Seit Jahren bin ich sehr zufrieden. Bisher kein Reklamationsgrund.
* Quelle: Shopauskunft.de (16.11.2012)

Über 97 % unserer Kunden sind vom **reichelt**-Service überzeugt*

Hier bestellen die Experten!

VISATON GERMANY

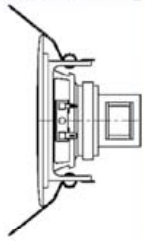
Industrielle Soundtechnik

VISATON

Breitbandlautsprecher

Mit Hochtonkegel
und Euro-Normkorb

Ø 100 mm
4 Ω /8 Ω
PMPO 50 W
75 - 20.000 Hz
Reson.: 88 Hz



VIS DL 10 WS Ø 10 cm **23,30**

VISATON

Elektrodynamischer Exciter

zur Anregung von Platten zu Biegewellenschwingungen. Die Befestigung des Exciters kann durch Kleben oder Schrauben erfolgen.

VIS EX45S-8 Ø 45 mm **25,65**
VIS EX60S-8 Ø 60 mm **41,35**

VISATON GERMANY

Breitbandlautsprecher

Mit Hochtonkegel
und Euro-Normkorb

NB 20 W | MB 30 W | 46 mm Einbautiefe | 94 mm Einbaudurchmesser | +70°C - 25°C

Ø 100 mm
4 Ω /8 Ω
PMPO 30 W
95 - 22.000 Hz
120 Hz

5,60 VIS FR 10HM-4

VIS FR 10HM-8 8 Ω **5,60**

Breitband-Kleinlautsprecher

Besonders geeignet für Anwendungen im Außenbereich.

Ø 50 mm
8 Ω
PMPO 3 W
250 - 10.000 Hz
Reson.: 350 Hz
IP 65

Wasserfest

VIS K 50-8 **1,85**

Breitbandlautsprecher

Mit feuerverzinktem Korb, sehr gutes Tieftonverhalten

Ø 100 mm
4 Ω /8 Ω
PMPO 50 W
80 - 20.000 Hz
Reson.: 88 Hz

VIS FR 10-4 4 Ω **10,25**
VIS FR 10-8 8 Ω **10,25**

Breitband-Kleinlautsprecher

mit transparenter Kunststoffmembran und Gummidichtring.

Ø 80 mm
4 Ω
PMPO 30 W
200–5000 Hz
Reson.: 290 Hz
IP 65

Wasserfest

VIS SL 87WPM-4 4 Ω **7,70**

Breitbandlautsprecher

Mit inverser Sicke, in besonders flacher Bauform

Ø 130 mm
4 Ω /8 Ω
PMPO 40 W
70 - 18.000 Hz
Reson.: 110 Hz

28 mm Einbautiefe

VIS FR 12-4 4 Ω **11,45**
VIS FR 12-8 8 Ω **11,45**

www.reichelt.de
Bestell-Hotline: **+49 (0)4422 955-333**

Katalog kostenlos! Jetzt anfordern!