www.elektor-magazine.de

.magazine

Juni 2013 | Nr. 510

elektor

Fernsteuern per WLAN

Für LEDs, Motoren, Relais – und vieles mehr!

Netzteil-Recycling | **Elektronik einfach steuern**

Peilsender für Flugmodelle | Akkulader ohne Mikrocontroller

Resonanzmeter | **FPGAs programmieren**

Neues aus dem Labor Soziale Roboter

G3078 E

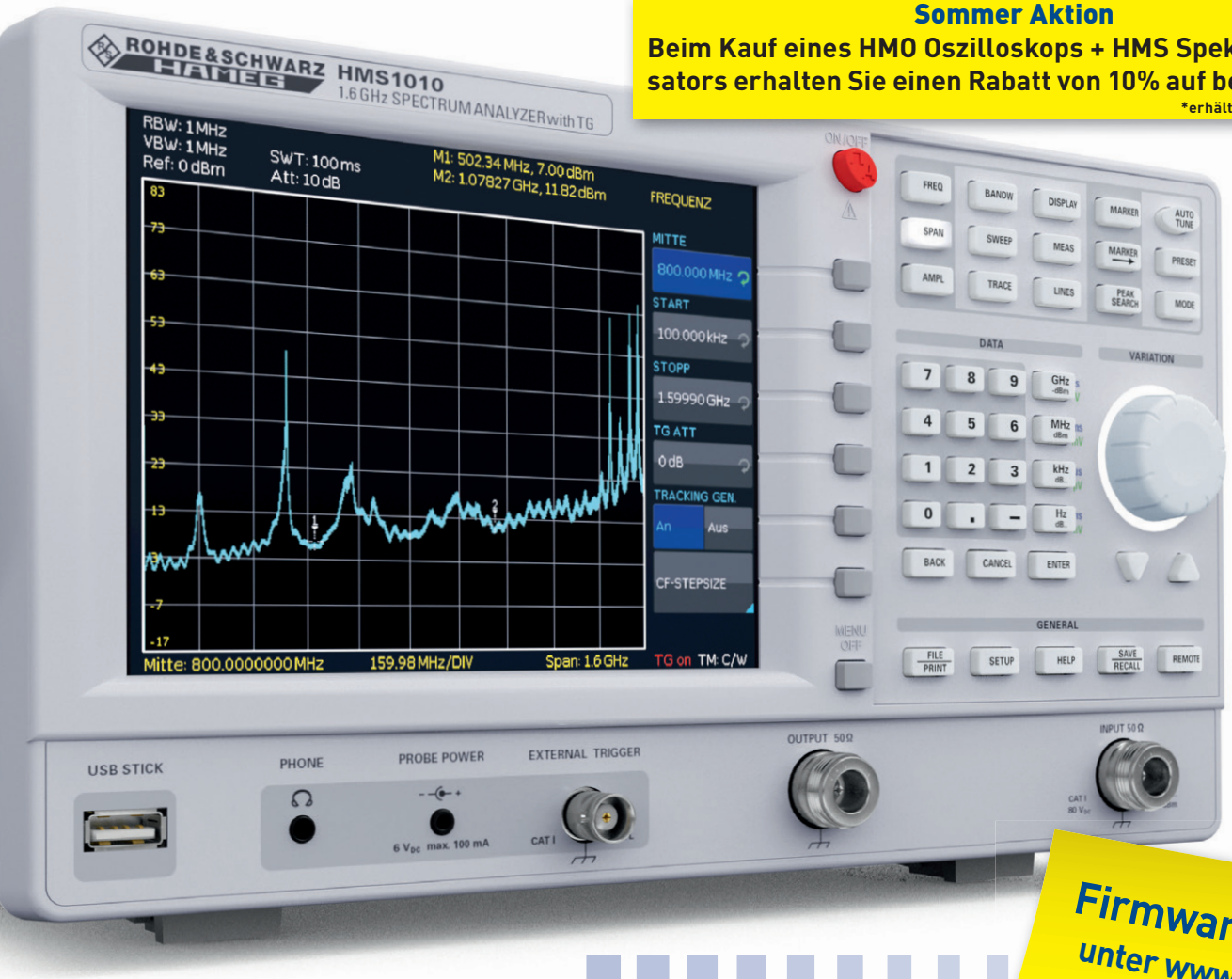
(D) € 7,90 | CHF 14,80 | (A, B, L) € 8,90

4 198630 307906 06

SPEKTRUMANALYSATOREN

HAMEG®
Instruments
A Rohde&Schwarz Company

Sommer Aktion
Beim Kauf eines HMO Oszilloskops + HMS Spektrumanalysators erhalten Sie einen Rabatt von 10% auf beide Geräte!
*erhältlich bis Oktober 2013



NEU!!
Firmware-Upgrade
unter www.hameg.com
kostenfrei verfügbar

+60%

MEHR FREQUENZBEREICH

HMS1010 | HMS1000 | HMS1000E

von 1GHz auf 1,6GHz

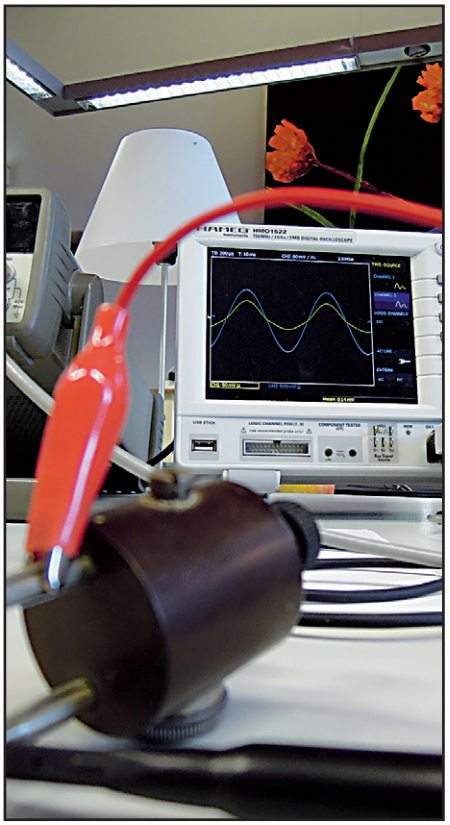
HAMEG Instruments hat den Frequenzbereich bei allen Spektrumanalysatoren der 1000er Serie von 1 GHz auf 1,6 GHz erhöht.

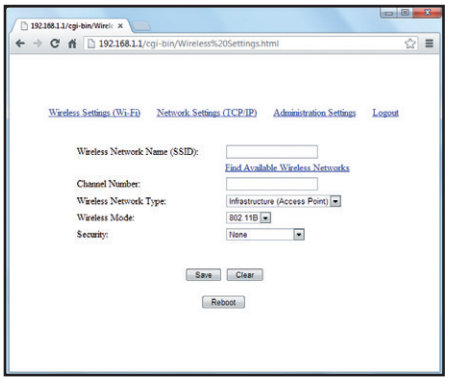
Somit erhalten HAMEG Kunden beim Kauf eines HMS1000, HMS1010 und auch beim HMS1000E ab sofort 60% mehr Frequenzbereich ohne Aufpreis.

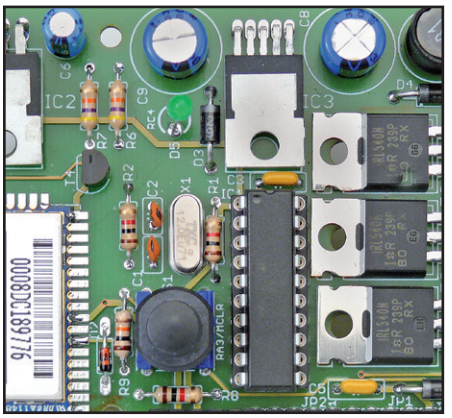
www.hameg.com | Great Value in Test & Measurement

Inhalt

elektor magazine





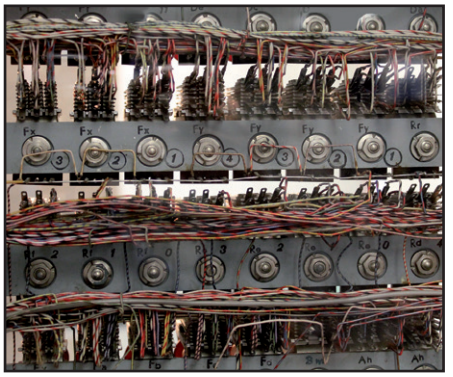


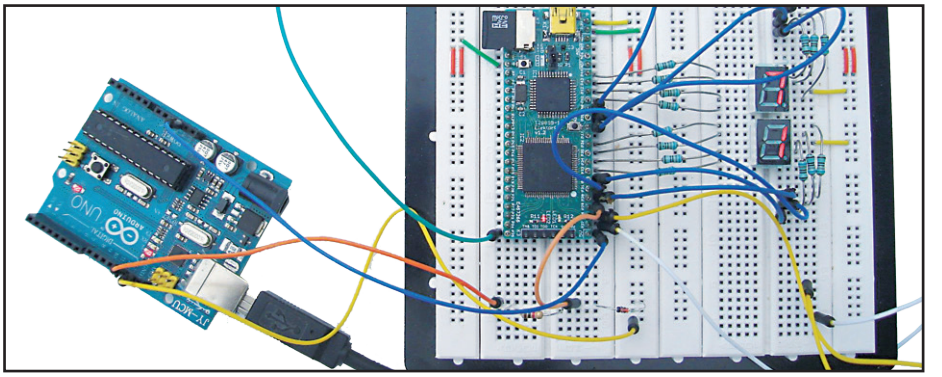
Community

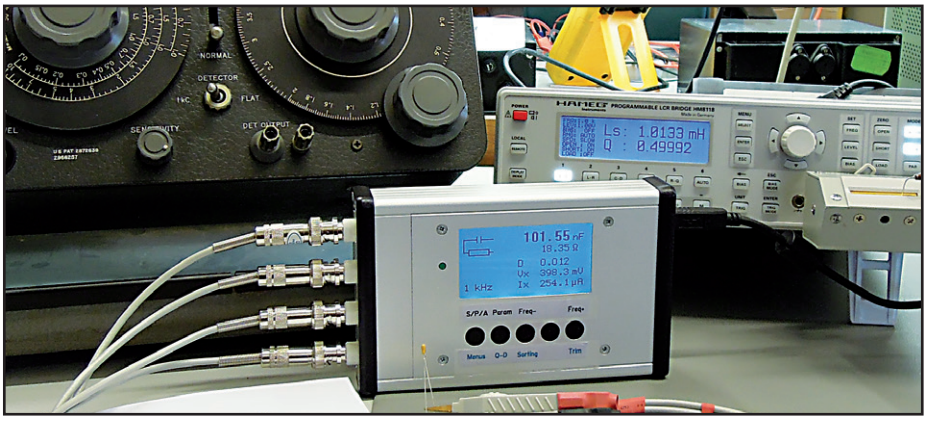
Labs

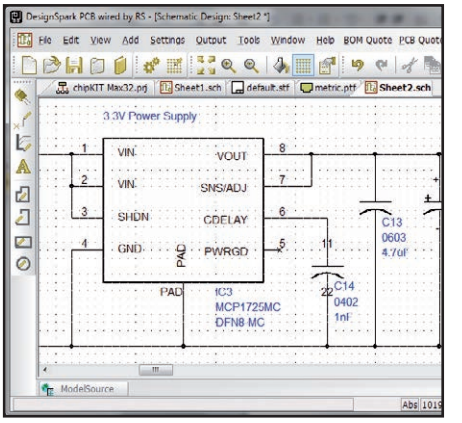
Industry

Projects









Hier ist ein Peilsender/-empfänger nötig, der schnurstracks zum verunglückten Modell führt.

Tech the Future

Magazine

Aus technischen Gründen musste das Projekt „FM-Sender“ auf Ausgabe 7-8/2013 verschoben werden.

Impressum

44. Jahrgang, Nr. 510 Juni 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. 43 ab 01.01.2013

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

Wir geh'n ins WLAN

Auch ohne hellsehen zu können, weiß ich schon jetzt, dass wir mit unserem Titelprojekt den Geschmack der Leser getroffen haben. Mit (RGB-)LEDs und WLAN sind in diesem Projekt gleich zwei Elektronik-Bereiche kombiniert, bei denen viele Elektroniker kribbelige Finger bekommen (lange Jahre habe ich fast alle unsere Webnews verfasst, und da konnte man schön an den Klick-Zahlen sehen, welche Keywords im Titel gut ankommen. Es ist ja auch ein schönes Gefühl, drahtlos etwas steuern zu können, stimmt's? Und da der Mensch ein „Augentier" ist, lässt er sich vor allem durch optische Effekte begeistern.
Da ich auch schon Erfahrungen mit HTML-Steuerungen habe, bin ich natürlich gleich selbst in den Artikel „eingestiegen". Ich habe das Datenblatt des benutzten WLAN-Moduls heruntergeladen und mir auch die Software meines Kollegen Clemens angeschaut. So ein WiFi-to-Serial-Modul macht einem die Entwicklung eigener Anwendungen wirklich einfach. Sogleich hatte ich die Idee einer kleinen Platine, die nur die Beschaltung des Moduls enthält; die Pins der seriellen Schnittstelle sind dabei auf einen kleinen Stecker herausgeführt. Und da wir gerade zufällig eine Stecker-Spezifikation für Seriell-Interfaces in der Mache haben, wird dieses Modul an diverse kommende Controllerboards passen; gleichzeitig aber auch mit RS485- und RS232-Adaptern ausrüstbar sein. Genau an so etwas arbeitet mein Kollege Ton gerade in unserem Labor, aber nicht für WLAN, sondern für den 433-MHz-Funk.
Schön, wenn elektrisch alles so toll kombinierbar ist, aber ein Problem bleibt. Was verwenden wir als Anwendungs-Protokoll, damit sich unsere Boards dann tatsächlich auch verstehen? Dass es für das vielgepriesene „Internet of Things" noch keine richtige Protokoll-Spezifikation gibt, haben auch die Kollegen der Computerzeitschrift c't schon bemängelt. Doch das muss ja nicht so bleiben. Warum rufen wir nicht einfach unsere riesige Elektor-Community zur Hilfe?
Über all das erfahren Sie mehr in den nächsten Heften, im Newsletter und auf unserer Project-Site www.elektor-labs.com!

Jens Nickel

Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Rolf Gerstendorf, Klaus Boda
Internationale Redaktion:	Harry Baggen, Thijs Beckers, Jan Buiting, Eduardo Corral, Wisse Hettinga, Denis Meyer, Clemens Valens
Elektor-Labor:	Thijs Beckers, Ton Giesberts, Luc Lemmens, Tim Uiterwijk, Jan Visser
Grafik & Layout:	Giel Dols, Mart Schroijen

Germany
Ferdinand te Walvaart
+49 241 88 909-17
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
+31 6 1589 4245
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 · techthefuture

Die Elektor-Community

Unsere Partner und Sponsoren

Sie möchten Partner werden?

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

50 % Rabatt für Elektor-Mitglieder

Seminar „Rapid SMD-Prototyping“

Elektor veranstaltet in Kooperation mit Beta LAYOUT und IBF Friedrich ein praxisorientiertes Eintages-Seminar zum Thema „Rapid SMD Prototyping“. Dabei geht es um die Erstellung eines kleinen Platinenprojekts mit anschließender SMD-Bestückung unter realen Bedingungen. Mitarbeit am Laptop bei Entflechtung und Anpassung an das spätere Gehäuse, sowie professionelle Bestückung mit Stencil und Lötofen gehören dazu.

An Vorkenntnissen sind Erfahrungen mit Elektronik und Grundwissen im Umgang mit Windows hilfreich. Zielgruppen sind Ingenieure, Entwickler, Techniker und engagierte Hobbyisten. Es finden zwei zertifizierte Seminare jeweils am 05.06.2013 (in Hanau) und 11.09.2013 (in München) mit begrenzter Teilnehmerzahl statt. Alle Hardwarekosten sowie eine Vollversion von TARGET 3001! in der „PCB-POOL Edition“ sind bereits im Preis enthalten.

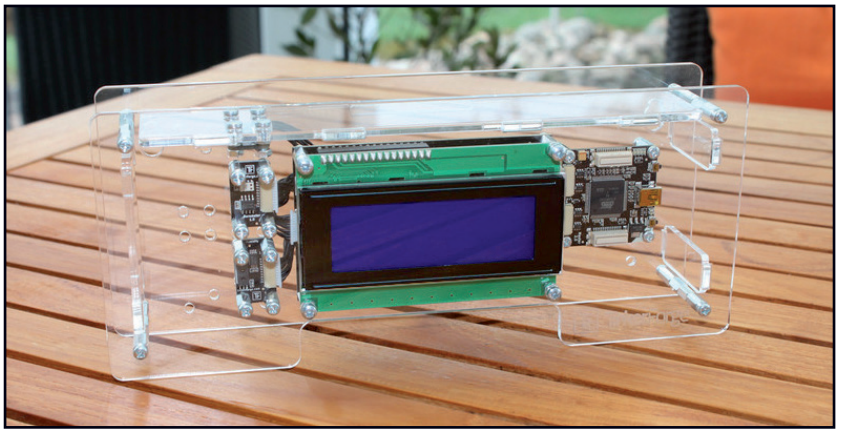
Elektor „Gold“- oder „Green“-Mitglieder erhalten 50 % Sonderrabatt auf den normalen Seminarpreis und zahlen somit nur 149,00 €. Die Teilnahmegebühr für Nicht-Mitglieder beträgt 299,00 €.

www.elektor.de/smd-prototyping

Reichelt: Neuer Katalog mit günstigem Oszilloskop

Das Titelprodukt des neuen Katalogs von Reichelt Elektronik ist ein digitales Speicher-Oszilloskop zu einem außergewöhnlich niedrigen Preis. Das UTD 2025 CL von Uni-T verfügt über ein großes 7“-LCD sowie umfangreiche Anzeige-, Rechen- und Analysefunktionen. Für unter 250 Euro bekommt man ein Zweikanal-DSO mit 25 MHz Analogbandbreite sowie einer Abtastrate von 250 MSamples/s. Das LCD stellt bei einer Diagonale von 7“ und 64 k Farben die Signale mit einer Auflösung von 400 x 240 Pixeln dar. Eine Auto-Setup-Funktion, die deutschsprachige Menüführung sowie die umfangreichen Anzeige- und Auswertefunktionen erlauben eine benutzerfreundliche Bedienung. Bildschirminhalte und Signalverläufe können intern und über einen USB-Anschluss abgespeichert werden. Das digitale Speicher-Oszilloskop ist nur eine der zahlreichen Neuheiten des aktuellen Katalogs. Auf 1.280 Seiten enthält er insgesamt rund 45.000 Produkte, darunter neben elektronischen Bauteilen zahlreiche Produkte aus der Welt der Messtechnik und auch unterschiedlichste Neuheiten aus dem Computer-Bereich und der Gebäudeautomation.

www.reichelt.de



Programmierkit Wetterstation

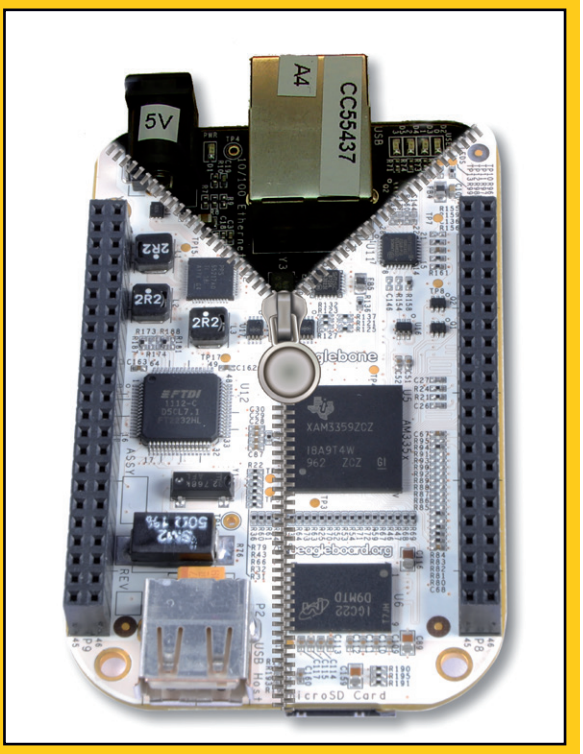
Mit dem neuen Open-Source-Starterkit Wetterstation von Tinkerforge wird das Erlernen einer Programmiersprache viel interessanter. Durch Schritt-für-Schritt-Anleitungen gelangt der Anwender langsam an das Ziel des Projekts: Die Entwicklung einer vollwertigen Wetterstation. Im Gegensatz zu üblichen Wetterstationen kann hier das Verhalten der Station selbst bestimmt werden, sodass sogar das Anzeigen der Wetterdaten auf Webseiten möglich wird.

Das Gehäuse ist aus Kunststoff, verfügt bereits über Befestigungslöcher und ist leicht bearbeitbar. Die Wetterstation kann um zusätzliche Module wie z. B. WLAN

erweitert werden. Sie ist in schwarzem und transparentem Gehäuse für den Einführungspreis von 99,99 € ab sofort erhältlich.
Das Baukastensystem von Tinkerforge besteht aus Sensoren und Motorsteuerungen, die modular zusammengesteckt und von einem (Embedded-)PC, Tablet oder Smartphone gesteuert werden können. Mit einer intuitiven und einfachen API lassen sich diese Bausteine über Sprachen wie C, C++, C#, Delphi, Java, PHP, Python, Ruby und .NET Sprachen steuern. Auch ein direktes Ansprechen per TCP/IP ist möglich.
Die Module können über WiFi, USB, RS485 (Modbus) und bald auch Ethernet angeschlossen und vernetzt werden. Die Wetterstation ist das erste anwendungsorientierte Starterkit, das der Hersteller auf den Markt bringt, weitere sind in Planung.

www.tinkerforge.com

1-GHz-Linux-PC für 45 Dollar

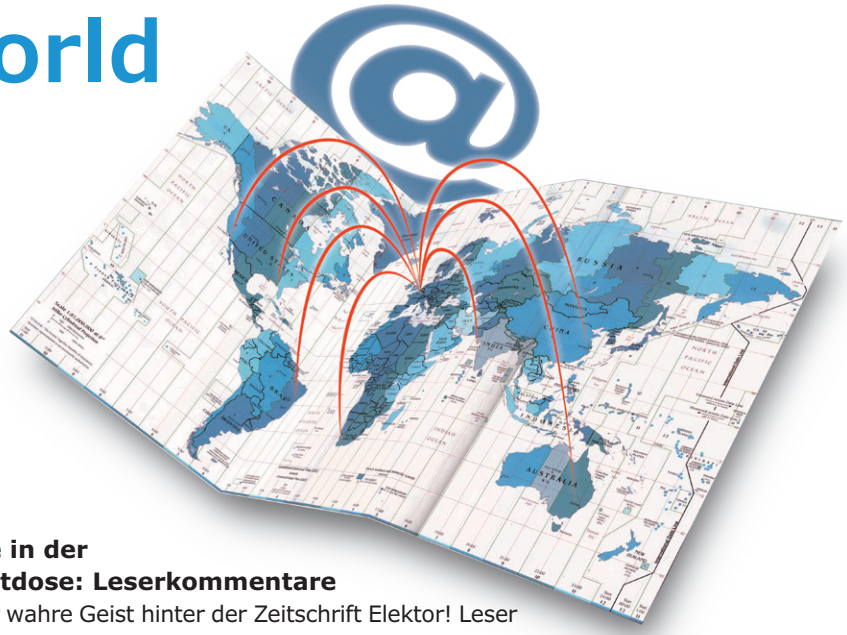


BeagleBoard.org stellt den neuen „BeagleBone Black" vor. Zum Preis von 45 US-Dollar erhält man einen kreditkartengroßen Linux-Computer mit 1 GHz Taktfrequenz. Entwickler können auf den Einfallsreichtum und das Wissen der Anwender in der äußerst aktiven und engagierten BeagleBoard.org-Community zurückgreifen.
Der BeagleBone Black enthält sämtliche Komponenten, die zum Anschluss von Display, Tastatur und Netzwerk erforderlich sind und ermöglicht die sofortige Aufnahme der Entwicklungsarbeit.
Grundlage ist der mit 1 GHz getaktete ARM-Cortex-A8-Prozessor AM335x von TI. Der Sitara AM335x erzielt mehr als doppelt so viel Performance als Lösungen auf ARM11-Basis. Der BeagleBone Black ist mit 2 GB On-Board-Speicher für die werksseitig installierte Linux-Software ausgestattet und verfügt über ein USB-Kabel zur Stromversorgung.
Mit seinen USB-, Ethernet- und HDMI-Schnittstellen kann er an eine Vielzahl von Geräten angeschlossen werden – ob Maus, Tastatur oder LCD-Bildschirm. Mit seinen Erweiterungssteckerleisten, unter anderem mit 65 digitalen I/O-Leitungen, sieben analogen Eingängen und der Möglichkeit zum Anschluss einer Vielzahl analoger und digitaler Peripheriegeräte, bietet der BeagleBone Black den Entwicklern überdies viel Flexibilität.

www.beagleboard.org

Elektor World

Zusammengestellt von
Wisse Hettinga



Die Röhre in der Butterbrotdose: Leserkommentare

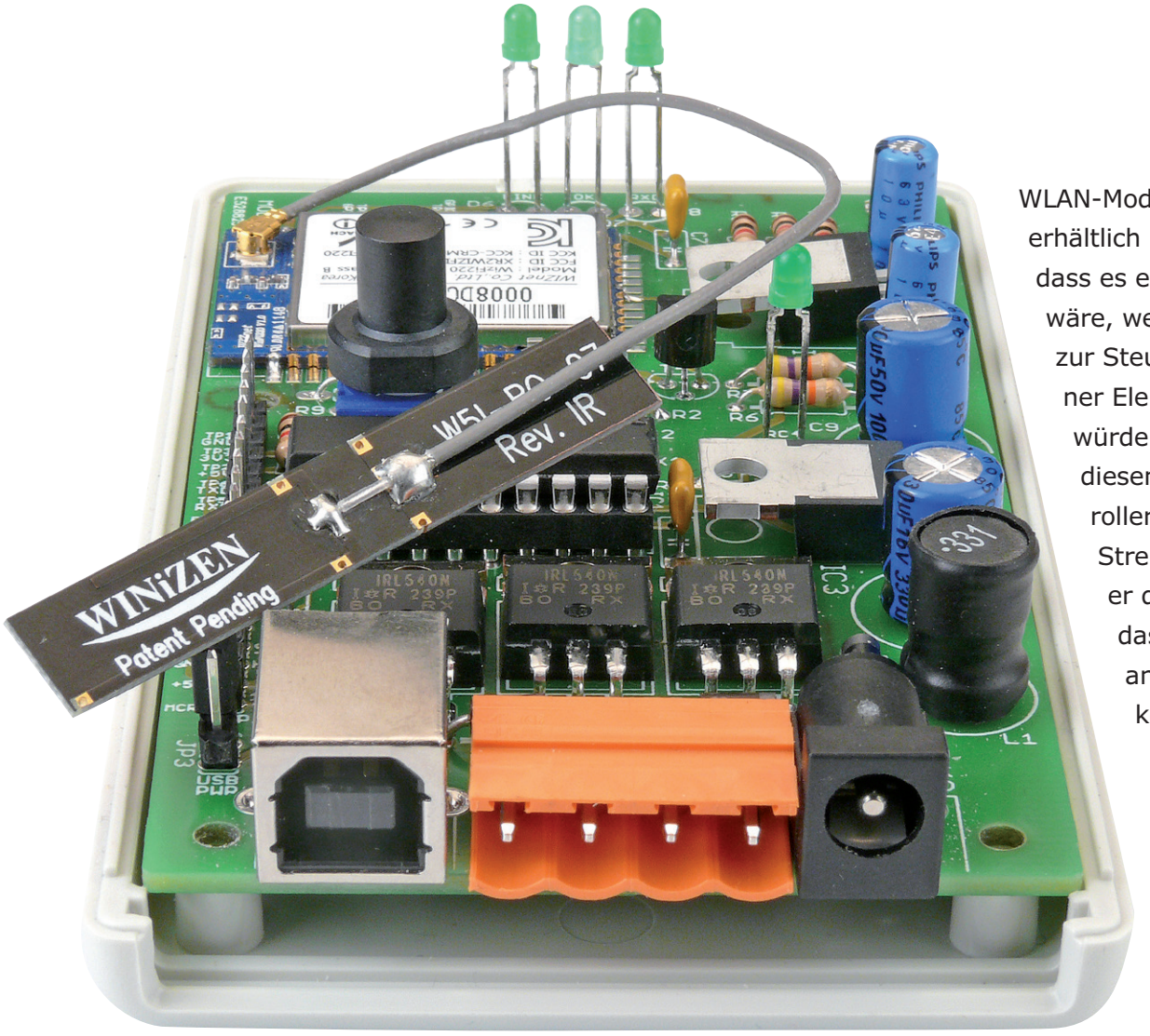
Sie sind der wahre Geist hinter der Zeitschrift Elektor! Leser wie **Sie** verbessern und erweitern den redaktionellen Inhalt durch eigene Untersuchungen und Experimente. Artikel, die sich auf Bauteile beziehen, sind dabei wohl am beliebtesten. Im vergangenen April hatten wir Ihnen von der Röhre in der Lunchbox berichtet und baten Sie um weitere Informationen. Unter anderem haben Christopher Kessler aus Deutschland und Jan Swenker aus den Niederlanden weitere Informationen zu diesem Thema ausgegraben. Christopher fand die Beschreibung dieser Röhre in „Valvo Fotovervielfacher 1978-1979“ auf insgesamt sechs Seiten und einen Preis, der aktuell vergleichbar bei 25 € liegen würde. Jans Informationen bestätigen das Herstellungsjahr 1978. Er wies uns auf ein Buch von 1986 über den Hamamatsu Photomultiplier hin, das als PDF im Internet zu finden ist. Das Buch benennt die Spezifikationen einer Ersatz-Röhre R1450. Es ist an der Zeit, sich die Hände schmutzig zu machen und die Röhre wieder zum Laufen zu bringen! Vielen Dank an Jan und Christopher!

Getting control

„Die Regelungstechnik beschäftigt sich mit der Frage, wie man die Eingangsinformationen eines Systems variieren kann, um sein Verhalten zu ändern. Do-it-yourself'er kennen sich gut mit einfachen kleinen Steuersystemen aus (wie denen in einfachen Schrittmotoren). Aber heutzutage können Bastler aus Projekt-Bausätzen, vom Robotic-Sumo-Auto bis zum auto-pilotierten Flugmodell, mehr über die Theorie komplexer Regelsysteme lernen“, sagt Brian Douglas in seinem Aufsatz „Tech the Future“ in der Zeitschrift *Circuit Cellar* von Juni 2013. „Heimwerker gehen nicht zurück ins Klassenzimmer, sie gehen ins Netz – bilden sich selbst auf Webseiten weiter und wenden sich Open-Source-Software und –Hardware zu.“ Douglas sollte es wissen! Der Regelungstechnik-Ingenieur aus Seattle hat eine YouTube-Website zur Förderung des praktischen Verständnisses der Regelungstheorie entwickelt: www.youtube.com/user/ControlLectures. Douglas ist einer der Autoren der Rubrik „The Future Tech“ in Circuit Cellar, die sich auf Trends und neue Entwicklungen in der Technik konzentriert. Besuchen Sie circuitcellar.com/category/tech-the-future, um mehr von seinen Aufsätzen zu lesen.

WLAN-Controller-Board

Steuert RGB-LED-Streifen, Motoren, Relais - und alles ohne Kabelsalat



WLAN-Module sind so leicht erhältlich und preiswert, dass es eine Schande wäre, wenn man sie nicht zur Steuerung von eigener Elektronik einsetzen würde. Obwohl wir mit diesem WLAN-Controller einen RGB-LED-Streifen ansteuern, ist er doch so universell, dass wir damit viele andere Dinge tun könnten!

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

Ein Projekt mit langer Vorgeschichte. Es begann alles vor einem Jahr mit einem Home Automation System [1], das Koen und Jesper, zwei Trainees im Elektor-Labor entwickelten. Ein Teil ihres Systems war ein RGB-LED-Streifen, der über drahtgebundenes Ethernet angesteuert wurde. Die Idee war gut, aber eine drahtlose Verbindung wäre eine bessere Lösung. Außerdem sollte die Anwendung mit den meisten Web-Browsern kompatibel sein. Die Farben des LED-Streifens sollten mit (virtuellen) Schiebereglern einstellbar sein. Koen machte sich an die Arbeit, aber seine Zeit bei Elektor war vergangen, bevor er das Projekt abschließen konnte. Er berichtete mir von einer Reihe von ungelösten Problemen, die dann auf meinem Schreibtisch landeten und

anschließend monatelang Zeit hatten, sich von einer Staubschicht bedecken zu lassen. Als ich endlich Zeit und Energie fand, mich mit der Problematik zu beschäftigen, waren die meisten von Koens Worten meinem Gedächtnis entschwunden. So musste ich von vorne anfangen...

Koens Schaltung bestand im Wesentlichen aus einem ATmega328 AVR-Mikrocontroller, der drei MOSFETs mit PWM-Signalen versorgte, die die Intensität der drei Farben regulierten. Für die WLAN-Verbindung setzte er ein WizFi220-Modul von WIZnet ein, das mit der CPU über eine einfache serielle Verbindung kommuniziert. Da ich über ein Arduino-Shield für dieses WLAN-Modul [2] verfüge und da ein Arduino Uno auf einem ATmega328 basiert, beschloss ich, das Projekt als Arduino-Anwendung zu erstellen. Alles, was ich brauchte, war ein zweites Shield mit den MOSFETs drauf, so dass ich den LED-Streifen ansteuern konnte. Dies wurde schnell auf einem Stück Lochraster aufgebaut (**Bild 1**).

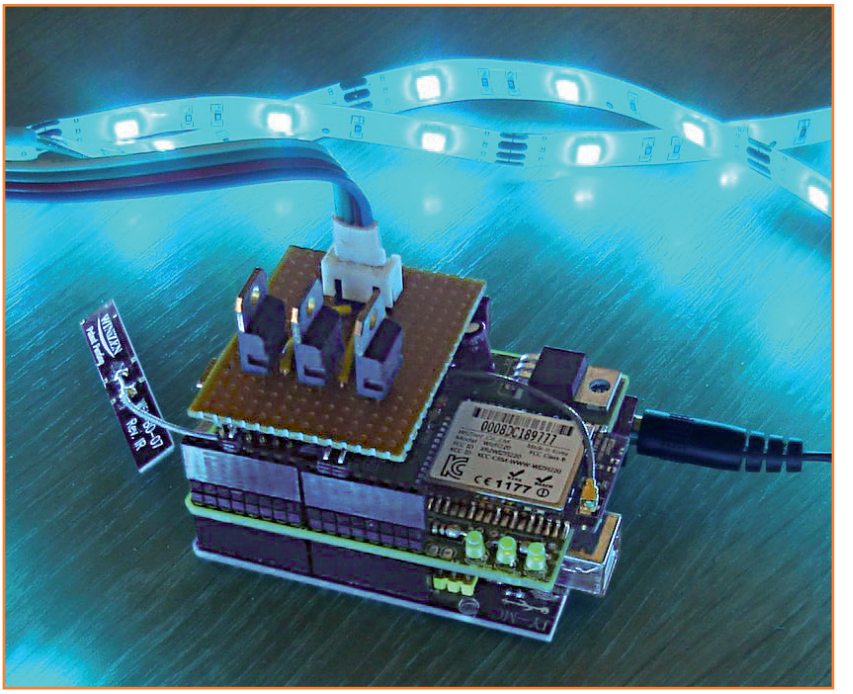


Bild 1.
Der Prototyp auf Arduino-Uno-Basis mit WLAN-Shield (120306) und einem Stück Lochraster.

Dann wurde es Zeit zu programmieren. Die Analyse der HTTP-Befehle vom WLAN-Modul und die Rücksendung der Antworten schien nicht so kompliziert, aber als ich die Web-Seite ändern wollte, lernte ich Koens Probleme kennen. Zwar gehören die Schieberegler, mit denen ich die Farbe der LED-Leiste steuern wollte, (endlich) zum HTML 5-Standard, sie werden aber von Firefox und dem IE nicht interpretiert. Koen hatte dieses Problem mit Hilfe der JavaScript-Bibliotheken jQuery und jQuery-UI [3] gelöst, Online-Bibliotheken zur Implementierung aller Arten von Steuerelementen und anderen Funktionen in Webseiten. Ein Nachteil dieser Bibliotheken ist, dass sie zu groß sind, um sie in einem Programmspeicher des Controllers ablegen zu können. Doch da ich keine andere Lösung fand, entschied ich mich, die Bibliotheken beizubehalten.
Ein Blick auf Koens HTML-Code erinnerte mich an ein Problem, das er erwähnt hatte: Aus irgendeinem Grund brach das WLAN-Modul die Kommunikation nach dem Empfang eines Befehls ab, was verhindert, dass der Browser mehrere Befehle senden kann. Folglich könnte man ohne Neustart des Moduls die Farbe des LED-Streifens nur einmal ändern. Koen hatte eine Lösung mit einem komplizierten Stück JavaScript gefunden, das im Browser vor der Befehlsausgabe den Kommunikations-Port ändert. Diese Umgehung des Problems verwendet den HTTP-GET-Befehl, um die Farbdaten zu senden (tatsächlich ist der Befehl dazu gedacht, Daten vom Server zu erhalten. Um Daten zu senden, wird der Befehl POST oder PUT verwendet.)
Eine viel elegantere Lösung ist es jedoch, die Zeile „Connection: close“ an die Server-Antwort auf einen GET-Befehl anzufügen. Jetzt konnte ich Koens Skript löschen und den Umfang der Webseite um einiges eindampfen, und ich konnte den angemesseneren POST-Befehl verwenden. So sieht der POST-Befehl aus, wenn er von Firefox gesendet wird (die Farbdaten am Ende):

```
POST / HTTP/1.1
Host: 192.168.2.15
User-Agent: Mozilla/5.0 (Windows NT 6.1;
WOW64; rv:17.0) Gecko/17.0 Firefox/17.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: application/x-www-form-urlen-
coded; charset=UTF-8
Content-Length: 23
Origin: null
Pragma: no-cache
Cache-Control: no-cache

red=79&green=10&blue=20
```

Der nächste Schritt war, die Webseite für mobile Geräte anzupassen. Auf meinem Smartphone waren die Regler winzig, so dass man sie nicht präzise bewegen konnte. Die Lösung war der `Viewport`-Meta-Tag im Header der Webseite:

```
<meta name='viewport' content='width=device-width, user-scalable=no' />
```

Mit dieser Zeile im HTML-Code sieht die Oberfläche auf meinem Android-Handy genauso gut aus wie am PC. Auf einem iPad belegt sie nur etwa ein Viertel des Bildschirms; ich habe nicht versucht, dies zu ändern.

Inzwischen hatte ich auch die Größe des restlichen Programms optimiert, nun war das nächste Ziel, alles in einem möglichst kleinen Programmspeicher unterzubringen. Eine wichtige Verbesserung war das Komprimieren der Web-Seite. Dies kann man mit gzip tun und die Zeile „`Content-Encoding: gzip`“ an die Server-Antwort anfügen. Die meisten oder alle Browser wissen, was mit gezippten Webseiten anzufangen ist. Der Nachteil dabei ist, dass sich die Webseite ein wenig komplizierter modifizieren lässt. Daher sollten Sie die Webseite erst komprimieren, wenn das Design abgeschlossen ist.

Jetzt hatte ich einen voll funktionsfähigen Prototyp auf Arduino-Basis mit Software, die innerhalb der Arduino IDE erstellt wurde. Die Details finden Sie unter [4]. Ich hätte mich nun zufrieden zurücklehnen können, aber irgendwie war ich mit einem dreilagigen Platinenstapel für eine solch einfache Schaltung nicht glücklich. Auf der Suche nach einer CPU mit weniger Pins entschloss ich mich, von der Atmel-Plattform abzuweichen, da einige Teile nur schwer zu bekommen waren. In einer Schublade fand ich ein paar 20-Pin-Controller mit USB vom Typ PIC18F14K50 von Microchip (was so alles herumliegt!). Dieser Controller dürfte eine bequeme PC-basierte Konfiguration des WLAN-Moduls erlauben (siehe auch [2]). Ein weiteres interessantes Feature ist der USB-Bootloader, den Microchip gratis abgibt. Er erlaubt eine einfache Firmware-Entwicklung ohne spezielle Programmer. Leider hat dieser Controller nur einen PWM-Kanal, so dass ich die PWM-Farbsteuerung in der Software implementieren musste.

Die Portierung des AVR-Codes zum PIC hätte eine einfache und unkomplizierte Fingerübung sein sollen, aber mitnichten! Natürlich hatte ich die Dinge durch den Bootloader und die USB-Funktionalität ein bisschen kompliziert, aber Microchip hätte es mir ein wenig einfacher machen können. Ich benutze den super-duper voll funktionsfähigen XC8-Compiler, von dem Microchip so schwärmt, aber es war unmöglich, den USB-Code zu compilieren, geschweige denn zum Funktionieren zu bringen. Ich habe diesen Code aus den Application Libraries v2012-10-15 von Microchip und nach vielen Stunden vergeblichen Hantierens mit den Compiler-Einstellungen und Pragma-Direktiven fand ich irgendwo im Internet die Information, dass XC8 (noch) nicht kompatibel mit den Application Libraries wäre. Na prima! Ich brauchte einen anderen Compiler. Aus dem Internet kam die Rettung: Innerhalb von 15 Minuten hatte ich einen offiziellen, voll funktionsfähigen C18-Compiler und eine weitere Viertelstunde später war der serielle USB-Port betriebsbereit.

Nun war es Zeit, meinen WLAN-Code hinzufügen, was ziemlich reibungslos verlief. Probleme bereitete nur die Aufbereitung aller Daten für das segmentierte RAM des PICs. Und nun der erste Test. Es funktionierte! Nicht richtig, aber ein bisschen. Ich konnte die Farbe des Bandes ein- oder zweimal ändern, aber dann brach die WLAN-Kommunikation zusammen. Als ich den LED-Strip über den seriellen USB-Port angesteuert habe, hat doch alles so gut funktioniert? Dann habe ich entdeckt, dass der USB-Interrupt den Interrupt für die serielle Schnittstelle aufhielt, was einen Datenverlust zur Folge hatte. Ich war ziemlich genervt und habe beschlossen, den USB im WLAN-Modus abzuschalten, anstatt dem Problem auf den Grund zu gehen. Wenn Sie das Rätsel lösen können, lassen Sie es mich wissen, es interessiert mich (und die Elektor-Leser) wirklich.

Aufbauarbeit

Jetzt funktionierte mein zweiter, PIC-basierter Prototyp mit den gewünschten Bauteilen (**Bild 2**) und ich konnte mich an den Entwurf des endgültigen Platinenlayouts machen. Die Steuerung für den RGB-LED-Streifen sollte in einem kleinen, unauffälligen Gehäuse Platz finden. Ich habe mich für ein blau-transparentes Gehäuse entschieden, durch das man die Status-LEDs beobachten kann, so dass außer für die Stromversorgung keinerlei Bohrungen erforderlich sind. Es müssen lediglich die Abstandsbolzen angepasst werden.

Da selbst bei einer 20-Pin-CPU mehrere Anschlüsse ungenutzt blieben, habe ich einige zusätzliche Funktionen hinzugefügt. In Kombination mit dem Bootloader verwandelt dies die Ansteuerelektronik in einen echten Einplatinencomputer, eine vielseitige Plattform, die man für alle möglichen Anwendungen nutzen kann:

- WLAN-RGB-LED (Streifen) Controller;
- Drei-Kanal-Controller zur Ansteuerung von Relais oder Stellmotoren über WLAN oder USB (oder beides);
- USB-zu-Seriell-Adapter;
- Prototyping-Platine mit WLAN, Expansionport und drei Leistungskanälen;
- FlowBoard kompatibel mit FlowStone 3 [5];
- etwas, an das ich noch gar nicht gedacht habe.

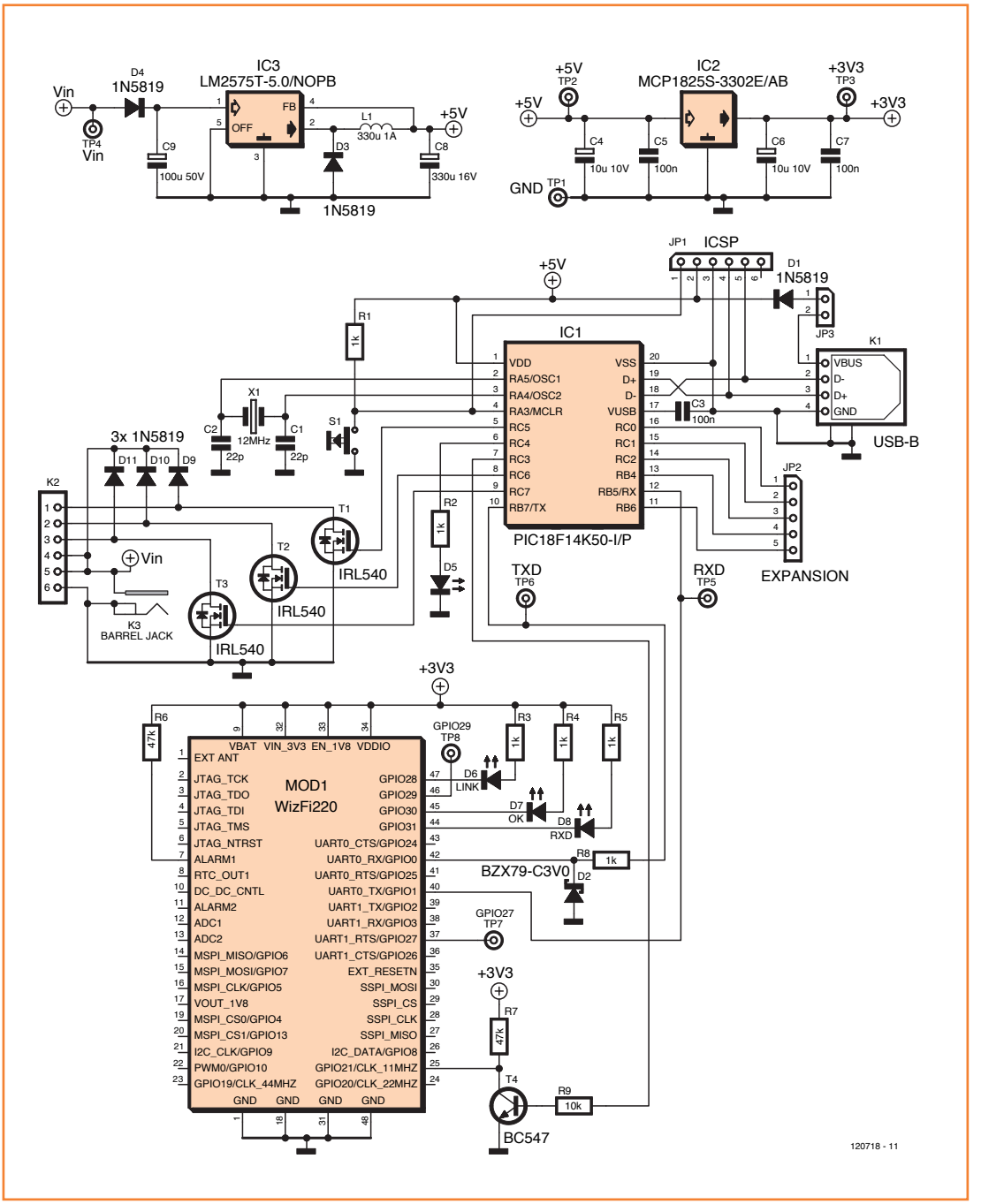


Bild 2.
Schaltung des WLAN-Controller-Boards. Ist Ihnen schon mal aufgefallen, dass die Kleinspannungsbuchse immer falsch herum benutzt wird? Der mittlere Pin sollte eigentlich Masse sein.

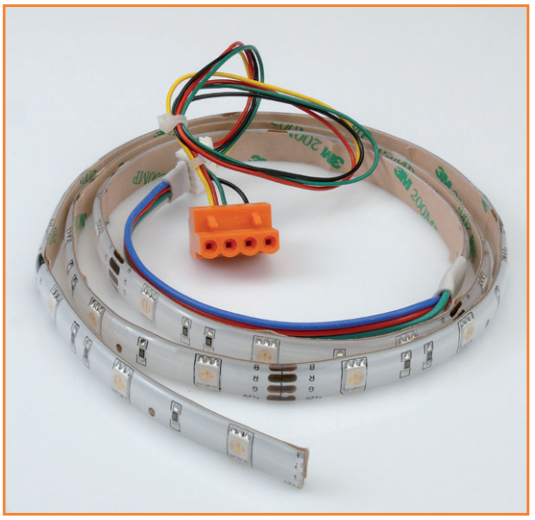


Bild 3.
Ein typischer RGB-LED-Strip mit einem PC-Netzteilkabel als Adapter.

Um viele unterschiedliche Anwendungen zu ermöglichen, wurde das Board mit einem Schaltregler ausgestattet, der Eingangsspannungen in einem weiten Bereich von 7...40 V_{DC} ermöglicht, ohne dabei verschwenderisch mit der Energie umzugehen. Die Schaltung kann auch über den USB versorgt werden, aber man sollte im Kopf behalten, dass das WLAN-Modul ziemlich viel Strom verbraucht, wenn es sendet. Ein großer 3,3-V-Linearregler versorgt das WLAN-Modul (dazu ist der in der CPU integrierte 3,3-V-Regler nicht in der Lage). Das Modul zeigt mit drei LEDs, ob es mit einem Access-Point verbunden ist oder nicht und ob Daten ausgetauscht werden. Wegen der unterschiedlichen Betriebsspannungen ist ein Pegelkonverter in Richtung Modul erforderlich, der mit 5 V versorgte Controller dagegen kann 3,3-V-Signale direkt verarbeiten.

Der Controller muss von einem 12-MHz-Quarz getaktet werden, um das korrekte Timing für den USB zu garantieren. Der Taster am Reset-Pin (MCLR) soll hauptsächlich den Bootloader-Modus aktivieren. Der externe Reset-Eingang muss dazu deaktiviert werden (fuse `MCLRE = OFF`). Wenn der Bootloader nicht benötigt wird, kann dieser Taster auch als Reset-Taste fungieren oder für eine andere Funktion verwendet werden.
Die drei (für diese Anwendung überdimensionierten) MOSFETs verfügen über einen $R_{DS(on)}$ von 0,077 Ω, können bis zu 100 V handhaben und werden von Flyback-Dioden geschützt, so dass auch induktive Lasten geschaltet werden können. Am Ausgang habe ich mich für vierpolige 5-mm-Platinenanschlussklemmen entschieden, obwohl die meisten LED-Streifen Anschlüsse mit kleinerem Raster aufweisen. Die verwendeten Anschlussklemmen sind Standard, Sie können leicht selber ein Adapterkabel (zum Beispiel von einem Diskettenlaufwerk) anfertigen (**Bild 3**). Für die Spannungsversorgung ist eine Klinkenbuchse oder eine zweipolige Platinenanschlussklemme (RM 5) vorgesehen. Die Diode dient dem Verpolungsschutz.
Ein 5-poliger Pfostenverbinder macht die noch freien Ports des Controllers für allgemeine Zwecke zugänglich. Eine Reihe von Testpunkten erweitert diesen Verbinder in die eine, die Anschlüsse zur seriellen In-Circuit-Programmierung (ICSP) in die andere Richtung. Alle zusammen erlauben sie den Zugriff auf zehn Controllerpins und alle Betriebsspannungen. Eine LED an RC4 und ein Taster sind ebenfalls vorhanden.

Das WLAN-Modul ist wegen seiner vielen Anschlüsse etwas fummelig zu montieren und sollte als erstes auf die Platine gelötet werden. Alle anderen Bauteile sind zur unproblematischen Durchsteckmontage geeignet und dürften keine Probleme bereiten. Beachten Sie, dass die Spannungsregler auf dem Bauch liegend oder auf der Lötseite der Platine montiert werden (**Bild 4**). So lassen sie sich im Bedarfsfall leicht an einen Kühlkörper schrauben (natürlich muss dann ein anderes Gehäuse verwendet werden).

Wenn die LEDs durch das Gehäuse ragen sollen, bohren Sie zunächst die Löcher, um die Anschlussdrähte anzupassen, und löten Sie erst danach die LEDs fest. LED D5 (an RC4) und der Taster befinden sich genau unter den Abstandsbolzen, so dass sie die Löcher ohne umständliches Abmessen bohren können. Das Gehäuse verfügt über ein Batteriefach. Wenn Sie die Status-LEDs auf der Lötseite montieren, können Sie sie nur dann sehen, wenn Sie das Batteriefach öffnen. Als Tasterkappe ist ein 16-mm-Modell geeignet, sie ist fast bündig mit der Oberseite des Gehäuses. Um die Platine in das Gehäuse einzupassen, müssen Sie die vier Abstandsbolzen im Gehäusedeckel kürzen. Dies geschieht am besten mit einem dicken Bohrer, aber Vorsicht: Lassen Sie den Schauben genug Material zum „Beißen“. Die beiden Befestigungen für die Batterie sind überflüssig und können beseitigt werden. Das WLAN-Modul besitzt eine kleine Buchse für eine externe Antenne (**Bild 5**). Wenn Sie eine anschließen, können Sie selbige einfach baumeln lassen.

Programmierung

Die Firmware, die Sie für dieses Projekt herunterladen können [6], enthält den Bootloader, die RGB-LED-Streifen-Applikation und die Fuse-Einstellungen in einer einzigen HEX-Datei. Laden Sie die HEX-Datei mit einem geeigneten Programmiergerät (PICKIT, ICD oder andere) in den Controller und schon kann es losgehen. Der Bootloader erlaubt es, eigene Applikationen aufzuspielen. Dies geht so:

- Spannungsversorgung aus, JP3 darf nicht eingesteckt sein;
- Verbinden Sie das Board mit einem freien USB-Anschluss Ihres PCs;
- Drücken Sie Taster S1 und halten Sie ihn gedrückt, während Sie JP3 platzieren. Statt JP3 kann auch ein Taster (Öffner, NC) verwendet werden. In diesem Fall drücken Sie ihn kurz, während Sie S1 gedrückt halten, um in den Bootloader-Modus zu gehen;
- Der PC (Windows, Linux oder Mac) sollte nun das Board als HID erkennen. Beim erstmaligen Anschluss an einen Windows-PC müssen Sie die im Download enthaltene .INF-Datei installieren.
- Auf dem PC starten Sie das HID-Bootloader-Tool (ebenfalls im Download, die Linux- oder Mac-Version kann der Microchip-Website entnommen werden) (**Bild 6**). Es sollte das Board sofort entdecken;
- Suchen Sie die ausführbare Datei (HEX-Datei) und drücken Sie die Programmier-Taste;
- Um den Bootloader-Modus zu verlassen und die Anwendung zu starten, drücken Sie die Reset-Taste oder schalten Sie die Stromversorgung des Boards aus (ohne S1 zu drücken).

Konfiguration des WLAN-Moduls

In [2] wird gezeigt, wie Sie das WizFi220-Modul über eine serielle Schnittstelle mit AT-Befehlen konfigurieren können. Diese Technik ist nützlich, wenn zum Beispiel der Controller das Modul „on the fly“ neu konfigurieren soll. Gewisse Optionen lassen sich auch gar nicht anders einstellen. Das Board verfügt über einen speziellen Modus, um das Modul auf diese Art zu konfigurieren. Das WLAN-Modul bietet einen noch einfacheren Weg, wenn das einzige, was Sie verlangen, eine Verbindung zu einem existierenden drahtlosen Netzwerk ist. Das Modul muss vom Board nur in

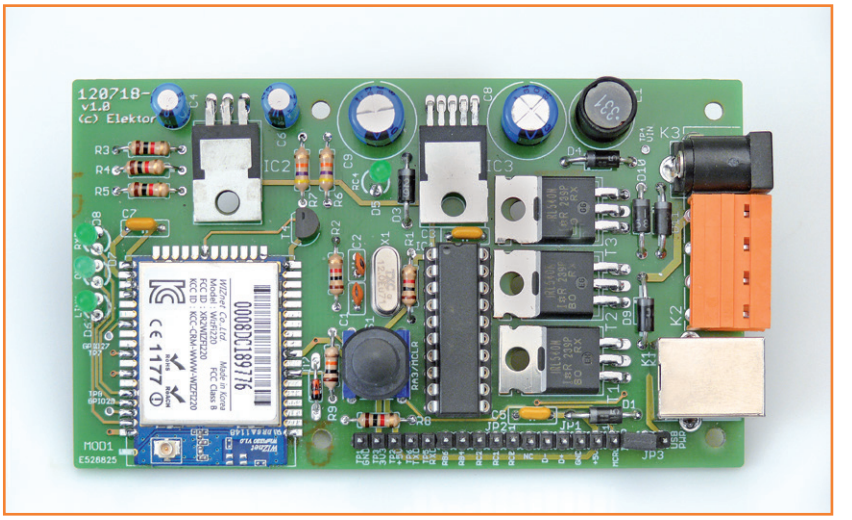


Bild 4.
Die fast fertige Platine, Version 1.0, komplett montiert. Die endgültige Version 1.1 ist beinahe identisch, nur einige Bauteile wurden leicht verschoben, damit alles besser in das Gehäuse passt.

den Modus *Limited Access Point (LAP)* versetzt werden.
Bei beiden Methoden müssen Sie das Board zunächst in den USB-zu-Seriell-Modus bringen. Dazu verbinden Sie RC2 mit +5 V (zum Beispiel mit einem Draht am Expansionsstecker) und dann das Board mit einem freien USB-Port Ihres PCs. Wenn LED D5 blinkt, können Sie AT-Befehle zum neuen virtuellen COM-Port (Nummer/Name erhalten Sie vom Betriebssystem) senden. Wenn das WLAN-Modul mit einem Access Point verbunden ist (Status-LEDs LINK und OK sind beide eingeschaltet), gibt man „+ + +“ am Terminal ein, um das Modul in den Command-Modus bringen. Die OK-LED wird dunkel. Wie das Modul konfiguriert wird, erfahren Sie in [2].

Um das WLAN-Modul in den LAP-Modus zu versetzen, bedarf es einiger transzendenter Anstrengungen. Schauen Sie die blinkende LED an und

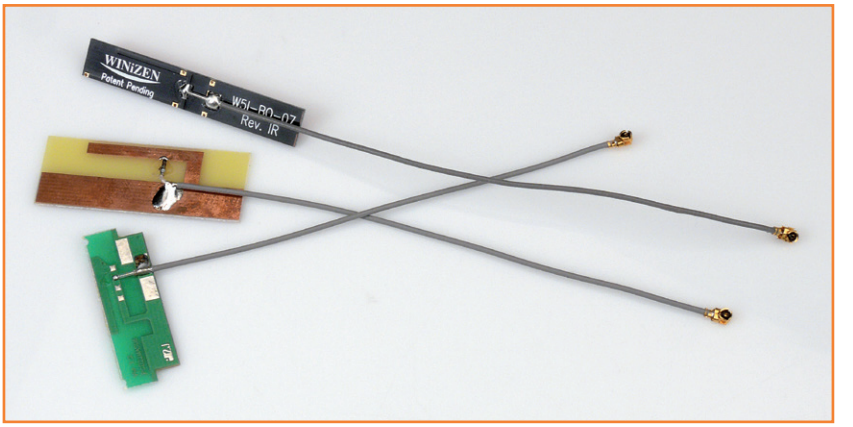


Bild 5.
Drei 2,4-GHz-Antennen, eine von Winizen (oben) und die beiden anderen freundlicherweise zur Verfügung gestellt von 2J (www.2j-antennae.com).

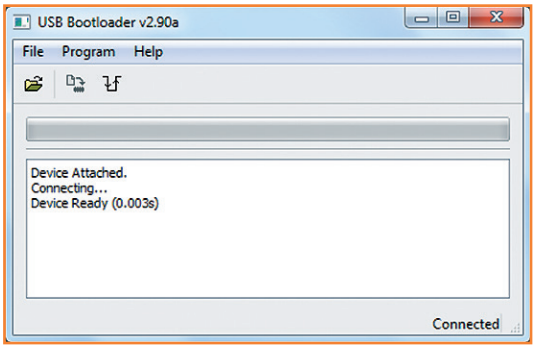


Bild 6.
Mit diesem Tool können Sie neue Anwendungen auf den WLAN-Controller hochladen.

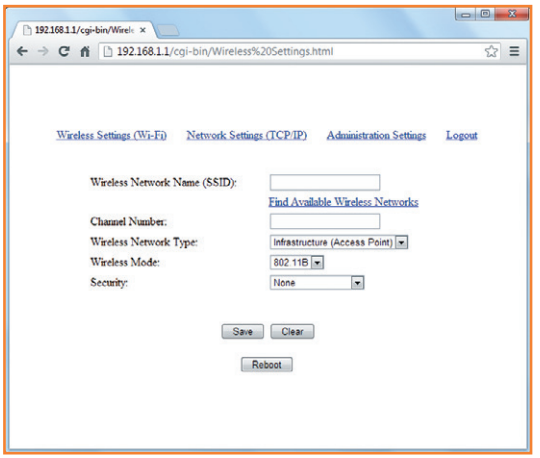


Bild 7.
Der Modus Limited Access Point erleichtert die Konfiguration des WizFi-Moduls erheblich.

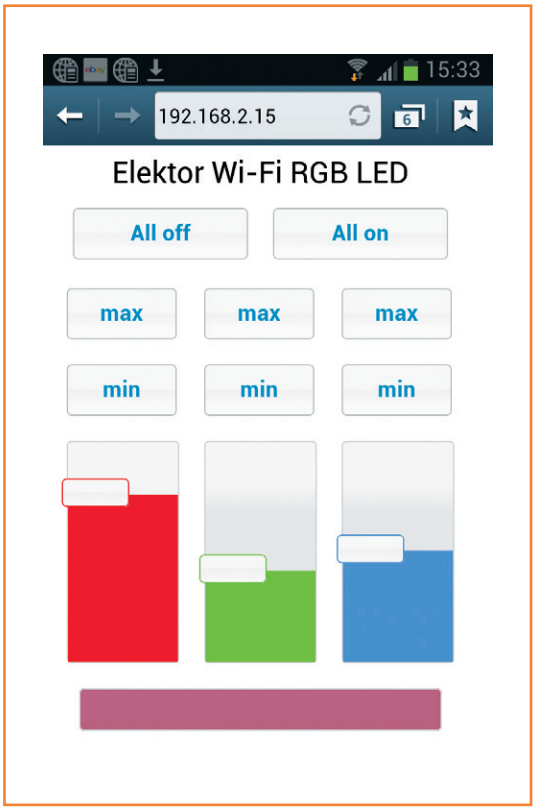


Bild 8.
Die Webseite des WLAN-Controller-Boards von einem Android-Smartphone aus gesehen.

lassen Sie sich von ihrem Rhythmus treiben. Sind Sie bereit? Dann drücken Sie S1, wenn die LED gerade dunkel ist und halten Sie die Taste während zweier Blinkzyklen fest gedrückt. Lassen Sie die Taste nur los, wenn die LED nach dem zweiten Blinken verlischt. Wenn Sie dies richtig gemacht haben, blinken nach etwa einer Sekunde alle drei LEDs wie verrückt und das Modul begibt sich in den LAP-Modus. Übrigens: Wenn Sie drei statt zwei Blinks abwarten, so schaltet das Modul in seine Werkseinstellungen zurück, eine gute Methode, um Ihnen mal aus der Patsche zu helfen, wenn überhaupt nichts mehr gehen will. Ist das Board am seriellen Port angeschlossen, so erscheint nun die Mitteilung:

```
IP            SubNet          Gateway
192.168.1.1:  255.255.255.0:  192.168.1.1
[OK]
```

Jetzt stellen Sie sicher, dass das Board in Reichweite Ihres WLAN-Access-Points ist. Auf dem PC, Tablet oder einem Smartphone sollten Sie nun noch einen neuen Access Point angezeigt bekommen, dessen Name *WizFiAPxxxx* (xxxx ist eine Zahl) ist. Stellen Sie die Verbindung her - der AP ist offen und Passwörter sind nicht erforderlich - und wählen im Web-Browser die Adresse 192.168.1.1. Sie sollten eine Seite ähnlich der in **Bild 7** erhalten. Klicken Sie auf den Link *Find Available Wireless Networks*. Es erscheint eine Liste mit erreichbaren Netzen, aus der Sie das gewünschte auswählen können. Klicken Sie auf *Save And Continue* und Sie landen wieder bei der ersten Seite, die nun die Details des gewählten Netzwerks anzeigt. Abhängig von den Sicherheitseinstellungen des Netzwerks können Sie ein Passwort bestimmen. Klicken Sie abschließend auf *Save*, um die Konfiguration zu speichern.
Klicken Sie *Network Settings (TCP/IP)* und füllen Sie das Formular aus. Sie können es halten, wie Sie wollen, aber ich verwende gerne eine statische IP für das Modul, so dass ich stets seine Adresse weiß. In das Feld *S2W Connection method* tragen Sie *1,1,,80* ein. Damit wird ein serielles Gateway auf Port 80 eingerichtet, dem Standard-HTTP-Port von Webbrowsern.
Sie können auch gerne einen anderen Wert verwenden. Wenn Sie fertig sind, klicken Sie auf *Save*. Mit *Administration Settings* können Sie ein Passwort für das Modul eingeben. Ich habe diese Option allerdings nicht verwendet.

Nach der Konfigurierung klicken Sie auf *Lockout*. Sie sehen die Meldung *Rebooting* und die WLAN-Verbindung wird unterbrochen. Das WizFi-Modul startet erneut und versucht, eine direkte Verbindung zum ausgewählten Netzwerk aufzubauen.

Wenn alles gut geht, leuchten die LEDs LINK und OK und die Verbindung steht wieder. Geben Sie die IP-Adresse des Moduls im Browser ein und warten Sie, bis Sie eine Seite wie in **Bild 8** erhalten. Jetzt ist die ganze Sache betriebsbereit.

(120718)

Weblinks

[1] Elektor Home Control:
www.elektor-labs.com/node/2325

[2] WLAN/Bluetooth-Shield:
www.elektor.de/120306

[3] JQuery(UI):
http://jquery.com & http://jqueryui.com

[4] Arduino: www.elektor-labs.com/node/2373

[5] FlowStone (in dieser Ausgabe):
www.elektor.de/130064

[6] Firmware, Eagle-Platinendateien, Stückliste:
www.elektor.de/120718

Stückliste

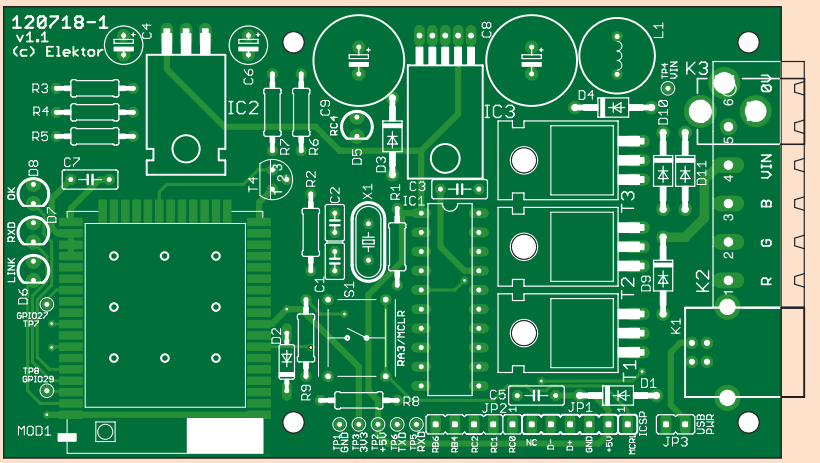
Widerstände (5 %, 0,25 W):
R1,R2,R3,R4,R5,R8 = 1 k
R6,R7 = 47 k
R9 = 10 k

Kondensatoren:
C1,C2 = 22 p, keramisch, 50 V, 2,5 mm
C3,C5,C7 = 100 n, Z5U, 50 V, 5 mm
C4,C6 = 10 µ 63 V, radial, 2,5 mm
C8 = 330 µ 16 V, radial, 3,5 mm
C9 = 100 µ 50 V, radial, 3,5 mm

Induktivität:
L1 = 330 µH 1 A, 5 mm, z.B. Würth Elektronik
Typ 7447452331

Halbleiter:
D1,D3,D4,D9,D10,D11 = 1N5819
D2 = 3 V Zenerdiode, z.B. BZX79-C3V0
D5,D6,D7,D8 = LED, grün, 3 mm
IC1 = PIC18F14K50-I/P
IC2 = MCP1825S-3302E/AB
IC3 = LM2575T-5.0/NOPB
T1,T2,T3 = IRL540
T4 = BC547

Außerdem:
MOD1 = WizFi220 mit Antenne, Elektor 130076-92
JP1 = 6-poliger Pfostenverbinder, 0,1“, vertikal
JP2 = 5-poliger Pfostenverbinder, 0,1“, vertikal
JP3 = 2-poliger Pfostenverbinder, 0,1“, vertikal
Jumper für JP3
K1 = USB-B-Buchse
K2 = Verbinder 1x4, 90 Grad, 0.2“, z.B. MSTBA4
K3 = DC-Kleinspannungsbuchse
20-polige DIP-Fassung für IC1
S1 = Multimec Typ RA3FTL6 mit Kappe Typ S09-16.0
X1 = 12 MHz Quarz, HC49/S-Gehäuse
Gehäuse, Hammond 1593QGY
Platine Elektor 120718-1



ATX-Netzteil-Recycling

Eine Adapterplatine für die Zweitverwendung

Von **Ben Jordan** (USA)

Diese Adapterplatine verwandelt jedes PC-Netzteil nach dem ATX-Standard in eine Stromversorgung für Experimente und für ein Labornetzteil – schlicht und elegant.

Eigenschaften

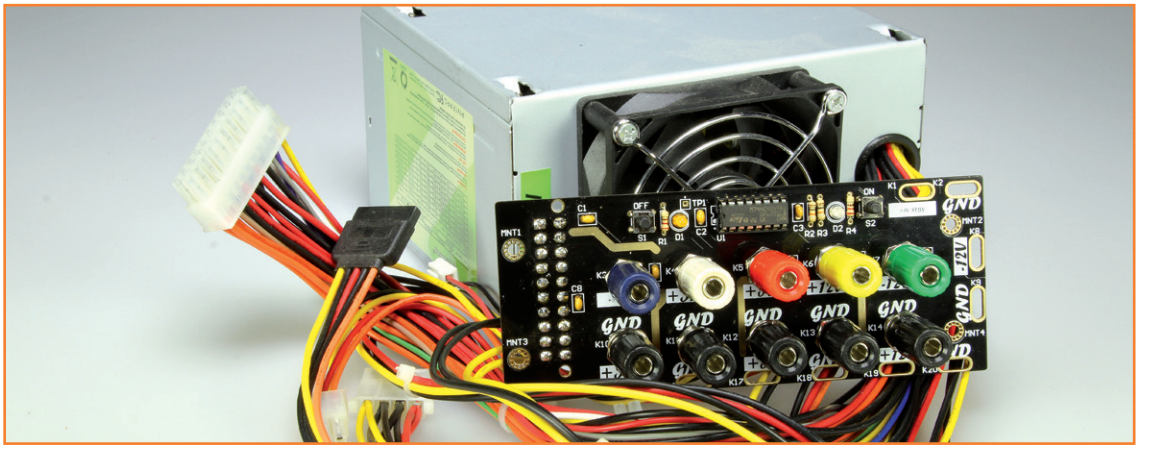
- Ausgänge für ±12 V, +3,3 V, +5 V, +12 V, -12 V und 5 V Standby
- Keine ATX-Modifikation erforderlich
- Einfaches Ein- und Ausschalten des ATX-Netzteils
- LED-Statusanzeige
- Hohe Ausgangsströme verfügbar
- Klemmen für alle Ausgangsspannungen
- Anschlussmöglichkeiten für Krokodilklemmen

Eine der wichtigsten Voraussetzungen für das Entwickeln von und das Tüfteln an elektronischen Geräten ist eine leistungsfähige Stromversorgung. Und wer wie ich mit einer Finanzministerin verheiratet ist, die (leider) nicht Elektrotechnik studiert hat, wird diese kaum von der Notwendigkeit überzeugen können, teures professionelles Equipment für den Hobby-Keller anzuschaffen. Also bleibt Selbstbau. Sich ein kleines Netzteil mit Linearregelung für digitale Schaltungen und Audioverstärker etc. zu bauen ist eine Sache, doch wenn es ein Netzteil mit ordentlich Power sein soll, dann ist das nicht mehr ganz so trivial. Meistens habe ich es mit Mikrocontrollern und Opamps zu tun. Von daher genügen in vielen Fällen eine 5-V-Versorgung für den Controller (und etwas zusätzliche Logik) sowie eine symmetrische ±12-V-Versorgung für die Opamps. Zunehmend bestehen modernere Controller wie ein Freescale DSP56367 oder ein LPC2101 (ARM-7-Mikrocontroller von NXP) auf niedrigeren Spannungen – typischerweise 1,8 V für den Kern und 3,3 V für die I/O-Einheiten. Es würde gewaltig nerven, für jedes Projekt wieder ein Netzteil mit all diesen Spannungen zu bauen. Glücklicherweise stehen diese (außer der 1,8-V-Schiene) schon an einem gewöhnlichen ATX-PC-Netzteil zur Verfügung. Wenn es Ihnen wie mir geht, dann haben sich im Laufe der Jahre einige dieser Netzteile aus ausgedienten PCs angesammelt und warten sowieso darauf, wieder etwas zu tun zu bekommen. Sie liefern die folgenden Spannungen mit ordentlich Strom: +3,3 V, +5 V und +12 V sowie schwächere 5 V für den Standby-Betrieb und -12V für andere Zwecke. Zugegebenermaßen ist die Ausregelung speziell der +12-V-Leitung nicht gerade berauschend, doch für die große Mehrzahl an Opamp-Schaltungen stört dies nicht weiter.

Vorüberlegungen

Naheliegend wäre, einfach eine Ladung Bananenbuchsen in so ein Netzteil einzubauen, wie man das so oft im WWWeb sieht. Fertig wäre das Labornetzteil. Doch das ist keine gute Idee. Hier die Gründe:

- In ATX-Netzteilen geht es innen recht beengt zu. Wenn man da Leitungen zieht, die nicht vorgesehen waren, dann unterschreitet man schnell wichtige Sicherheitsabstände. Das ist gefährlich, weil auf der Primärseite des Trafos durchaus Spannungsspitzen bis zu 1 kV auftreten können. Außerdem arbeitet die Eingangsseite mit einer Gleichspannung von +350 V. Kontakt wäre da nicht nur schlecht für die Finger, sondern auch für den damit ausgestatteten Menschen als Ganzes.
- Es geht ja um ein eigentlich gut funktionierendes PC-Netzteil. Wenn man es nicht modifiziert, dann kann man es neben der Versorgung von elektronischen Schaltungen auch durchaus noch zur Versorgung eines zu untersuchenden Motherboards auf dem Schreibtisch verwenden.



Mir schwebte also eine elegantere Lösung vor, die neben den üblichen Buchsen auch noch einige Kontakte für Kabel mit Krokodilklemmen besitzt.

Aus diesem Grund entschied ich mich für eine Platine, die als Adapter für den Laboreinsatz eines PC-Netzteils dienen soll. Zunächst einige Spezifikationen. Der Adapter sollte:

- Das Netzteil nicht verändern,
- mit ATX-Steckverbindern ausgestattet sein,
- Klemmen für alle ATX-Spannungen + Standby haben,
- für jede Spannung auch eine eigene Masseklemme haben,
- große Ströme vertragen können,
- die Ein-/Ausschalt-Elektronik des ATX-

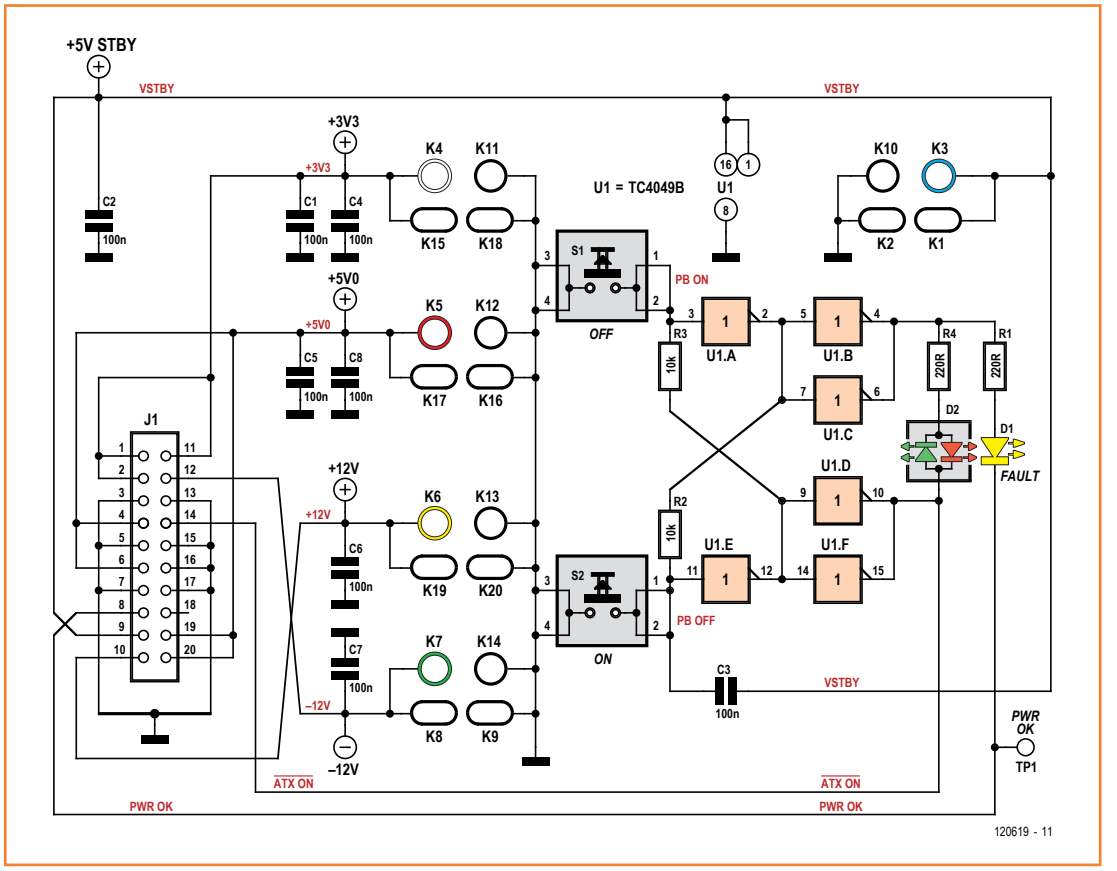


Bild 1.
Die Schaltung zeigt, wie einfach dieser Adapter gestrickt ist

Netzteils steuern können,
- LEDs als Standby-Überwachung und „Power Good Signal“ haben und
- bedrahtete Bauteile verwenden, sodass Ihnen der Nachbau leichter fällt.

Zuerst wollte ich auch noch Voltmeter-Module einbauen, doch dann wurde mir klar, dass dies unnötige Featuritis wäre. Die Spannungen sind ja einigermaßen stabil und wer es genauer wissen will, der kann sie ja immer noch mit seinem Multimeter checken.

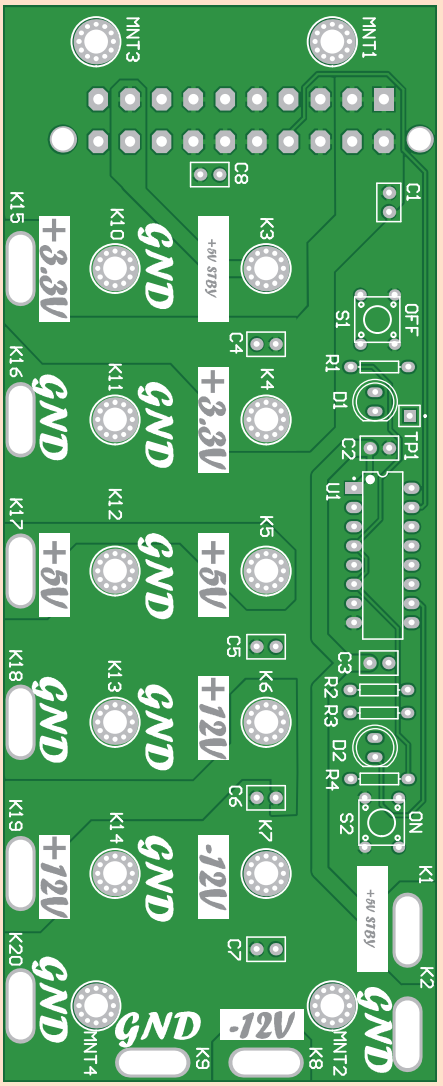
Stückliste

Widerstände:
R1, R4 = 220 Ω
R2, R3 = 10 k

Kondensatoren:
C1..C8 = 100 n, keramisch, RM 1/10“

Halbleiter:
D1 = LED, 3 mm, gelb
D2 = LED, 3 mm, 2-pol., zweifarbig
IC1 = 4049B

Außerdem:
K3..K7, K10..K14 = Bananenbuchsen
J1 = ATX-Stecker für Platinenmontage
S1, S2 = Taster



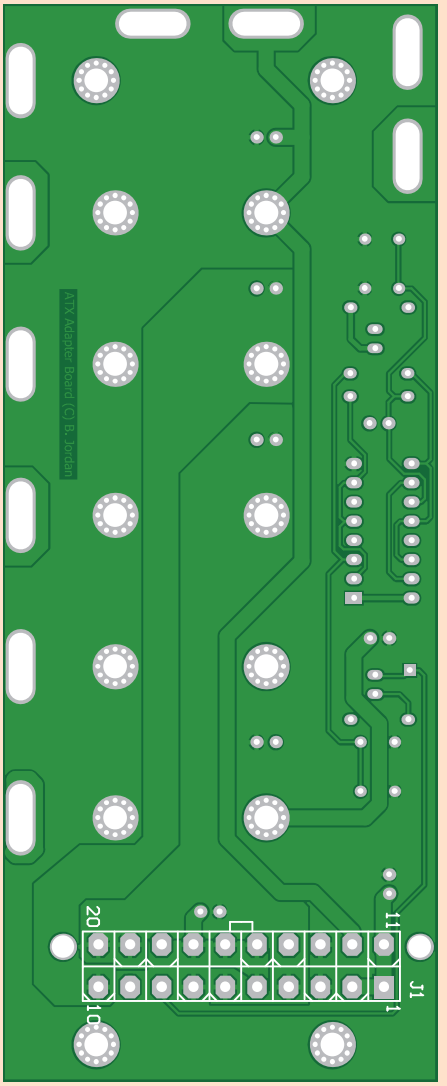


Bild 2.
Der Bestückungsplan lässt keinen Raum für Fehlinterpretationen. Hinweis: Der ATX-Stecker muss auf der Rückseite der Platine bestückt werden.

Einfache Schaltung

Bild 1 zeigt die simple Schaltung. Die Ansteuerung der Schaltelektronik geschieht mit zwei Invertern aus einem Hex-Inverter-IC des Typs 4049, das von der 5-V-Standby-Leitung versorgt wird. Die restlichen vier Inverter treiben das Steuersignal für das Netzteil und die LEDs. Mit S2 schaltet man das Netzteil ein und mit S1 wieder aus. Die zweifarbige LED D2 leuchtet im Standby-Modus grün und rot dann, wenn das Netzteil eingeschaltet ist.

ATX-Netzteile sind zwar mit Schatzmaßnehmen versehen, liefern aber auch ein „Power Good Signal“. Damit man sehen kann, ob alles in Ordnung ist, wäre eine Anzeige dieses Zustands per LED recht praktisch. D1 bleibt dunkel, solange das Signal „high“ ist (es ist „low active“). Wenn diese LED permanent blinkt, dann hat das Netzteil ein Problem.

C3 entprellt die Ein-/Aus-Schaltung. Alle anderen Kondensatoren entkoppeln die Versorgungsspannungen. Es werden grundsätzlich keramische 100-nF-Typen verwendet. R1 und R4 dienen als Vorwiderstände für die LEDs. R2 und R3 sind die Rückkoppelungspfade für das aus zwei Invertern bestehende RS-Flipflop.

Bei einigen der im Internet gezeigten Umbauten kann man sehen, dass Widerstände für eine minimale Last vorgesehen sind. Meiner Erfahrung nach ist das bei ATX-Netzteilen nicht mehr notwendig, da der eingebaute Lüfter als Grundlast ausreicht. Wer sich aber dabei besser fühlt, der kann z.B. einen 10-W-Widerstand mit 10 Ω an die 5-V-Leitung hängen.

Platine

Die Adapterplatine ist eine Art Break-out-Board mit ATX-Steckverbindern und dem nötigen elektronischen Kleinkram sowie Klemmen in Standard-Abständen, die für einfache oder doppelpolige Bananenstecker passen. Bei der Bestückung hilft **Bild 2**. Wichtig ist, dass der ATX-Stecker von der Rückseite und alle anderen Bauteile auf der Vorderseite bestückt werden.

Eine Besonderheit dieser Platine sind die verzinnten Langlöcher an den Rändern der Platine. Sie sind so dimensioniert, dass man dort leicht Kabel mit Krokodilklemmen befestigen kann. Auf dem Bestückungsdruck sind selbstverständlich die anliegenden Spannungen abzulesen. Das Platinen-Layout gibt es als PDF-Datei zum Download [2]. Die verzinnten runden Löcher K3 bis K14 sind für die leitende Rückseite von Bananenbuchsen gedacht. Zur besseren Stromleitung sind rings um diese Löcher etliche Durchkontaktierungen angebracht. Außerdem verkraftet die Platine dadurch höhere Kräfte beim Anziehen der Befestigungsmuttern.

In der Praxis

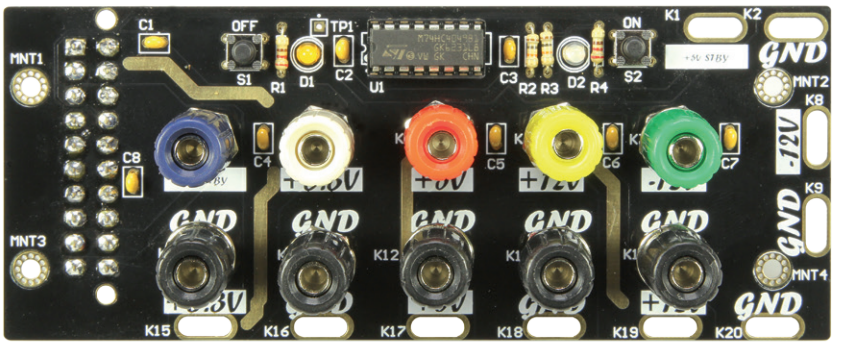
Die Bananenbuchsen sind im üblichen 3/4“-Abstand angeordnet, so dass Leitungen mit doppelpoligen Bananensteckern gut passen. Es gibt sie in farbcodierter Ausführung für +3,3 V, +5 V, +12 V und −12 V sowie 5-V-Standby. In meinem Hobby-Keller habe ich diesen Adapter ausgiebig in Betrieb. Er versorgt sogar mehrere Entwicklungs-Kits auf einmal. Selbst auf der Arbeit versorge ich ein Altium-Nanoboard-II und zwei Altium-Nanoboards-3000-FPGA plus weitere Kleinigkeiten mit einer einzigen Adapterplatine aus einem einzigen ATX-Netzteil. Die Adapter-Platine hat das Thema Stromversorgung für mich deutlich vereinfacht.

(120619)

Weblinks

[1] http://jordandsp.com/ATX-bench-top-power-supply-adapter.php

[2] www.elektor.de/120619

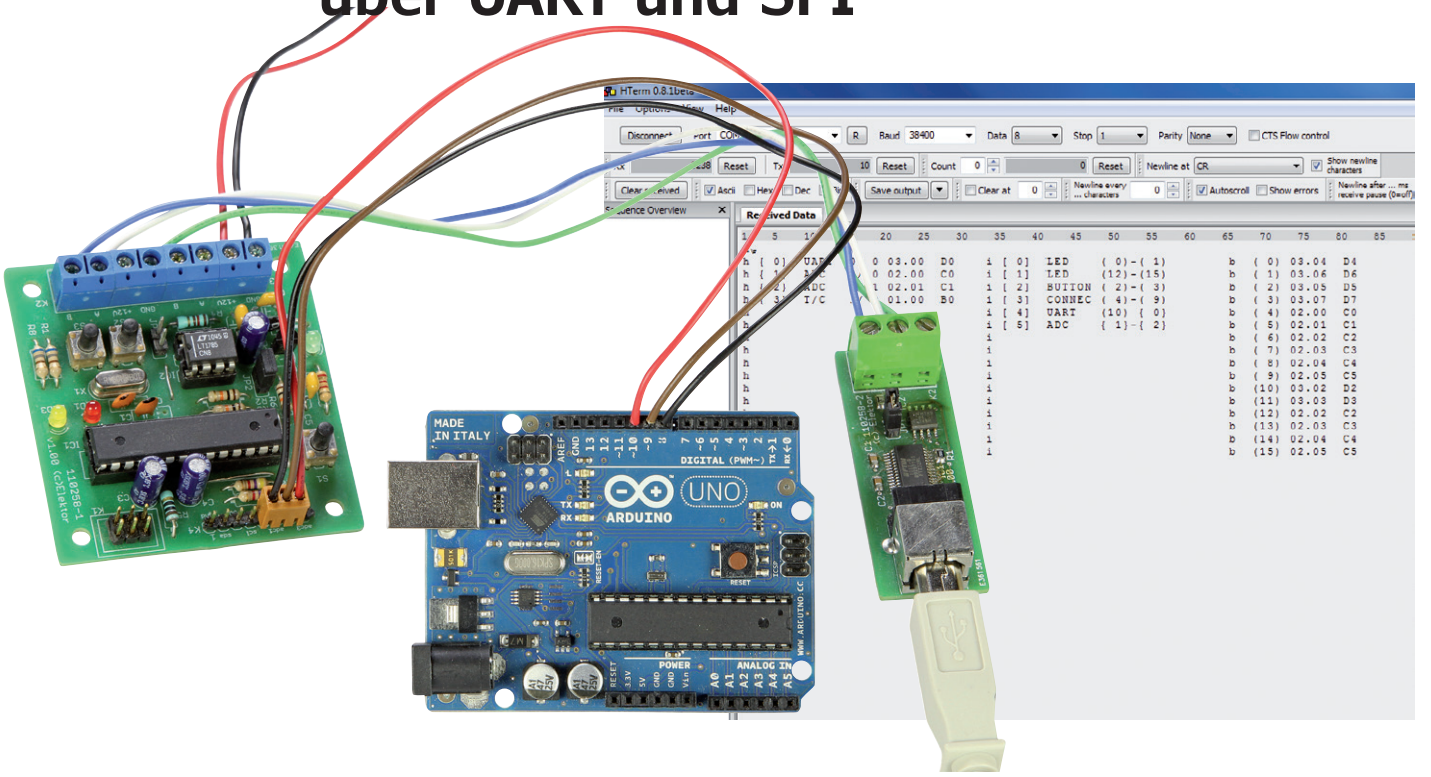


Das will ich auch...

Dieses Projekt ist in begrenzter Stückzahl als komplettes Kit aus Platine mit allen Bauteilen inklusive Klemmen und ATX-Stecker verfügbar. Hinzu kommt eine einfache Aufbauanleitung samt Platine, Bestückungsplan und eine Bohrvorlage für die Frontplattenmontage. Unter [1] gibt es weitere Informationen zu Preisen und Bestellmöglichkeiten.

Elektronik einfach steuern

Vom PC aus mit Boards verbinden - über UART und SPI



Von **Jens Nickel**

Mit einem textbasierten Protokoll und einer PC-Verbindung kann man eigene Elektronik von einem Terminalprogramm aus steuern. Passende Firmware lässt sich mit der „Embedded Firmware Library“ schnell programmieren; und dies unabhängig davon, ob man sich mit dem Controller über UART oder ein anderes Interface verbinden möchte. Das hier vorgestellte Protokoll eignet sich aber auch gut für Test- und Entwicklungszwecke.

Im letzten Heft haben wir die modular aufgebaute C-Bibliothek „Embedded Firmware Library“ vorgestellt [1]. Anfänger und Fortgeschrittene kommen damit schnell zu Code für ihr Embedded-Projekt, der überdies hardwareunabhängig ist und einfach von Board zu Board und von Controller zu Controller portiert werden kann. Hierfür sorgt ein sauber gekapselter Hardware-Layer, der aus einem Codefile für das Board und einem Codefile für den Mikrocontroller besteht.

Die Modularität der EFL geht aber noch weiter. Protokoll-Libraries lassen sich unabhängig vom verwendeten Übertragungskanal programmieren. Es ist also egal, ob die Kommandos und Daten zum Beispiel über UART/RS232/RS485 oder TCP/IP/Ethernet laufen. Um den Übertragungskanal in der Anwendung zu wechseln, genügt eine Änderung von wenigen Codezeilen. Dies wollen wir im

Rahmen dieses Artikels vorführen und gleichzeitig ein Protokoll vorstellen, das sich für einfache Steuerungsaufgaben genauso wie für Entwicklungszwecke eignet.

Mini-Protokoll

Bei der Code-Entwicklung sind einfache Tests wie das Schalten einer LED oder die Abfrage eines Digitaleingangs schon recht hilfreich. Oft verfügt ein Mikrocontrollerboard aber nicht einmal über Buttons, mit denen man einen Test auslösen könnte. Es bietet sich hier an, das Board von einem PC aus anzusteuern, zum Beispiel über eine serielle Schnittstelle. Wenn man sich auf ASCII-Zeichen beschränkt, kann man Kommandos bequem über ein Terminalprogramm absetzen; eine eigene PC-Software ist dann nicht nötig.

Mit unserem Mini-Protokoll („BlockProtocol“ genannt) lassen sich Pins von einem PC-Terminal aus auf High und Low setzen. Wenn man einen Pin ansteuert, an den eine LED angeschlossen ist, sieht man das Ergebnis direkt, ansonsten kann man natürlich ein Multimeter oder Oszilloskop anhängen. Außerdem lässt sich mit einfachen Kommandos der Status von digitalen und der ADC-Wert von analogen Eingängen abfragen (siehe Kasten „BlockProtocol“). Wer mag, kann den Code des zugehörigen Library-Moduls natürlich noch auf vielfältige Weise erweitern und anpassen. Vielleicht lässt sich der eine oder andere Leser dabei ja von den Steuerungsprotokollen von Andreas Eppinger [2] und Uwe Altenburg [3] inspirieren, die um ein Vielfaches mehr Funktionen bieten.

Map, Blocks und Boardpins

Speziell für EFL-Entwicklungszwecke bietet das BlockProtocol das Kommando „x“. Das Board sendet daraufhin einen Dump der EFL-Tabellen zum PC. Diese werden vom Hardware-Layer genutzt, um bei den Funktionsaufrufen die passenden Controllerpins und Register zu ermitteln. Die Werte sind spezifisch für das genutzte Board und eine eventuell angeschlossene Erweiterungsplatine. Der Screenshot in **Bild 1** zeigt, was im Terminalprogramm erscheint, wenn es sich um die daneben gezeigten Boards handelt. Links sieht man die „Map“, mit den beim Board-Init initialisierten Controller-Features wie UART und ADC. In der Mitte sind die Peripherie-Blocks verzeichnet und rechts sieht man eine Liste der Boardpins. Eine wertvolle Information ist das vor allem für denjenigen, der ein EFL-Boardfile für ein neues Board programmiert oder anpasst. Ausführliche Informationen über die EFL-Interna gibt es in dem Extra-Dokument zur EFL, das man unter [1] herunterladen kann.

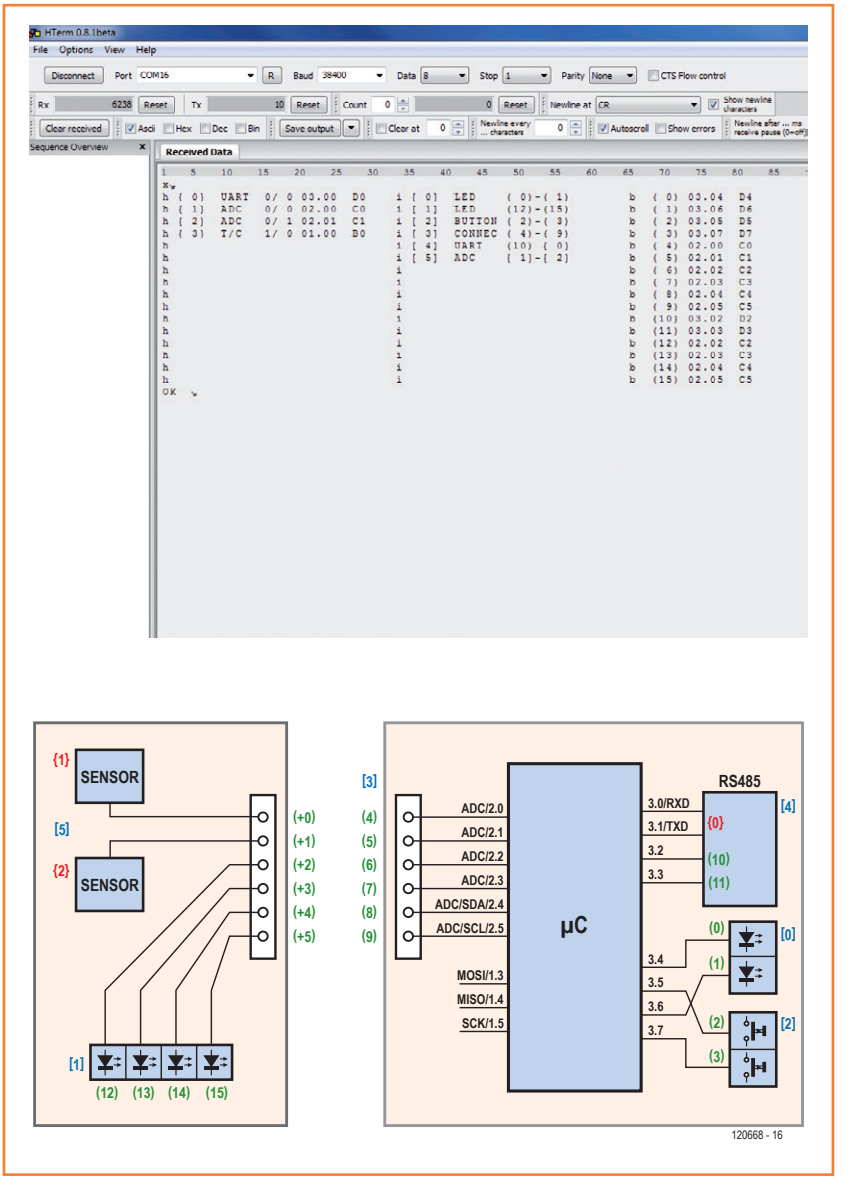


Bild 1.
Ausgabe der EFL-Tabellen im Terminalprogramm. Links sieht man die Map mit den genutzten Controller-Features (rote Zahlen im Blockdiagramm der Boards), in der Mitte die Block-Tabelle (blaue Zahlen) und rechts die Boardpins (grüne Zahlen).

Steuern über UART

An einem Beispiel wollen wir zeigen, wie einfach sich das Protokoll in einer eigenen Controller-Anwendung einsetzen lässt. Wir benutzen den aus der ElektorBus-Serie bekannten Experimentalknoten, ein sehr kompaktes Board, das mit einem ATmega328 ausgestattet ist. Den Experimentalknoten verbinden wir wie immer über RS485 mit einem USB/RS485-Konverter und diesen mit

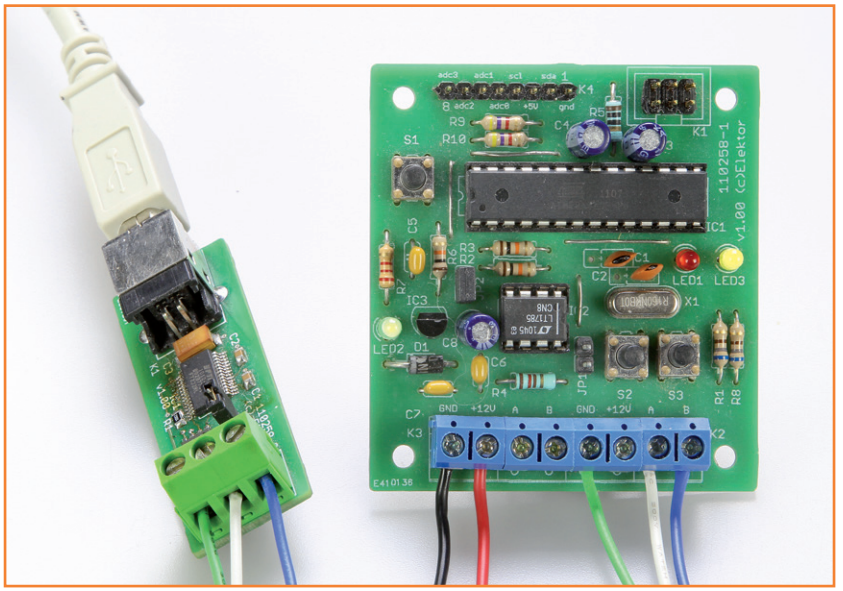


Bild 2.
Über UART/RS485 und ein einfaches ASCII-Text-Protokoll können wir unser kleines Mikrocontrollerboard von einem Terminalprogramm aus steuern.

einem PC (**Bild 2**). Statt des ElektorBus-Protokolls kommt aber nun das textbasierte BlockProtocol zum Einsatz.

Die zugehörige Firmware für den Controller kann man unter [4] herunterladen, außerdem ist die Anwendung in der EFL-Codebasis unter [5] enthalten (einen Datei-Überblick findet man im Extradokument). Ein Doppelklick auf „ExperimentalUART.atsln“ öffnet das Projekt im Atmel Studio, siehe Screenshot (**Bild 3**). Auf der rechten Seite erkennt man die eingebundenen Dateien. Die Files Controller.h/.c und Board.h/c bilden den Hardware-Layer. Es ist derselbe Code wie in der EFL-Beispielsoftware der letzten Ausgabe, weil wir das gleiche Board und den gleichen Controller benutzen. Im Ordner „Libraries“ findet man die Dateien UARTInterface.h/.c. Dies ist die Bibliothek für das UART-Interface auf dem Board, in diesem Fall in Gestalt unseres RS485-Treibers (*Physical Layer* der Kommunikation). Dazu kommt ein Modul für das verwendete Protokoll (BlockProtocolEFL.c/.h). Beide Bibliotheksmodule spielen über eine definierte Code-Schnittstelle zusammen (ein sogenanntes Software-Interface), was wir uns gleich in der Praxis ansehen.

Listing:
Grundgerüst einer EFL-Anwendung.

```
int main(void)
{
  Controller_Init();
  Board_Init();
  Extension_Init();

  ApplicationSetup();

  while(1)
  {
    ApplicationLoop();
  }
};
```

Kurzer Code

Im eigentlichen Anwendungscode (wie immer im Hauptfile der Anwendung angesiedelt) müssen wir die Bibliotheken einbinden mit:

```
#include "UARTInterfaceEFL.h"
#include "BlockProtocolEFL.h"
```

Die Main-Funktion ist bei jedem EFL-Projekt gleich aufgebaut, siehe **Listing 1**. In der Application-Setup-Funktion, die ja immer zu Beginn der Anwendung aufgerufen wird, initialisieren wir die Libraries:

```
UARTInterface_LibrarySetup();
UARTInterface_SetBaudrate(0, 38400);
BlockProtocol_LibrarySetup(UARTInterface_
Send, 0, UARTInterface_GetRingbuffer(0));
```

Mit der zweiten Zeile setzen wir die Bit-Rate des UARTInterface-Blocks #0 (hier ist unser RS485-Treiber angeschlossen) auf 38400 Baud. Die dritte Zeile verlangt etwas mehr Erklärung. Wir teilen der BlockProtocol-Library mit, dass sie sich der Funktion UARTInterface_Send bedienen soll, wenn Daten vom Board aus zu versenden sind. Der zweite Parameter ist die Nummer des zu verwendenden UART-Interface-Blocks; wobei wir beim Experimentalknoten nur über ein UART-Interface verfügen. Der dritte Parameter ist ein Zeiger auf den benutzten Ringbuffer, der die empfangenen Bytes aufnimmt. Den Zeiger auf den entsprechenden Ringbuffer erhalten wir als Rückgabewert der Funktion UARTInterface_GetRingbuffer(0), welche ebenfalls in der UARTInterface-Library implementiert ist.
In der regelmäßig aufgerufenen ApplicationLoop-Funktion benötigen wir nur eine Zeile:

```
BlockProtocol_Engine();
```

Diese Funktion ist die Hauptfunktion der Block-Protocol-Library. Sie schaut nach, ob Zeichen vom PC im Ringbuffer gelandet sind. So bald ein <CR> (ASCII 13) empfangen wurde, wird das als Ende eines Kommandos interpretiert. Der Befehl wird dann ausgeführt und eine Antwort zusammengestellt: Das ist entweder nur „Ok“, ein Wert wie „HIGH“ oder „LOW“ oder eben eine Ausgabe der EFL-Variablen in Tabellenform.

Mal probieren...

Nachdem wir das Programm compiliert und in das Board geflasht haben, probieren wir das Ganze gleich mit einem Terminalprogramm wie zum Beispiel HTerm [6] aus. Zuerst stellen wir dort den COM-Port und die Bitrate ein. Im Eingabe-Bereich des Programms („Input Control“) müssen wir noch einstellen, dass nach dem Drücken der Enter-Taste ein <CR> hinter die eingegebenen Zeichen geschrieben werden soll, bevor der ganze Eingabestring versendet wird (siehe **Bild 4**).

Wir versuchen es zuerst auf der untersten Ebene, mit einer direkten Angabe des zu schaltenden Controller-Portpins. Aus dem Schaltplan des Experimentalknotens [7] wissen wir, dass die rote LED auf dem Board mit dem Portpin PD4 verbunden ist. Port D bei den AVR-Controllern entspricht dem PortIndex 3. Also geben wir ins Terminalprogramm ein:

```
p 3 4 + <ENTER>
```

Jetzt müsste die rote LED auf dem Board angehen. Mit „p 2 0 +“ oder „p 2 0 –“ können wir nun zum Beispiel den Pin PC0 am Extension-Steckverbinder bedienen. Wenn wir eine Erweiterungsplatine anschließen, zum Beispiel unsere Sensor-/LED-Platine aus dem letzten Heft oder die Relaisplatine aus der ElektorBus-Serie [8], dann besitzen wir schon eine kleine Geräte-Steuerung, die vom PC aus bedienbar ist. Statt eines Terminalprogramms könnten wir auch eine eigene PC-Software verwenden, die einfach die Zeichen „p 2 0 + <CR>“ über den passenden COM-Port ausgibt, um etwa das Relais zu schalten. Oder wir schließen an den Experimentalknoten ein Smartphone über die RS485/UART-Bridge-Platine Andropod [9] an und schreiben uns eine kleine Android-App. Als Ausgangsbasis könnte man die Software unter [10] verwenden, wenn man die ElektorBus-Bytes durch die oben genannten Zeichen ersetzt.

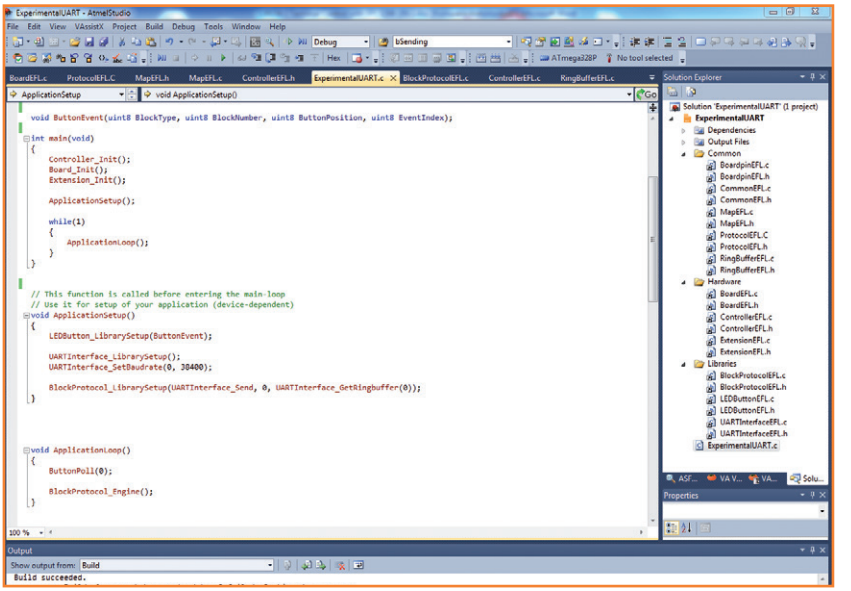


Bild 3.
Code zur Steuerung eigener Elektronik über ein UART-Interface.

Höhere Ebene

Doch die direkte Angabe eines Portpins entspricht natürlich gar nicht dem Wesen der EFL. Sobald wir für unsere Geräte-Steuerung ein anderes Controller-Board mit anderer Verdrahtung verwenden würden, müsste man die ausgegebene Zeichenkette in unserer PC- oder Android-Software wieder ändern.

Daher beherrscht das Mini-Protokoll auch das hardwareunabhängige Ansteuern von digitalen Ein- und Ausgängen, genauso wie wir das im

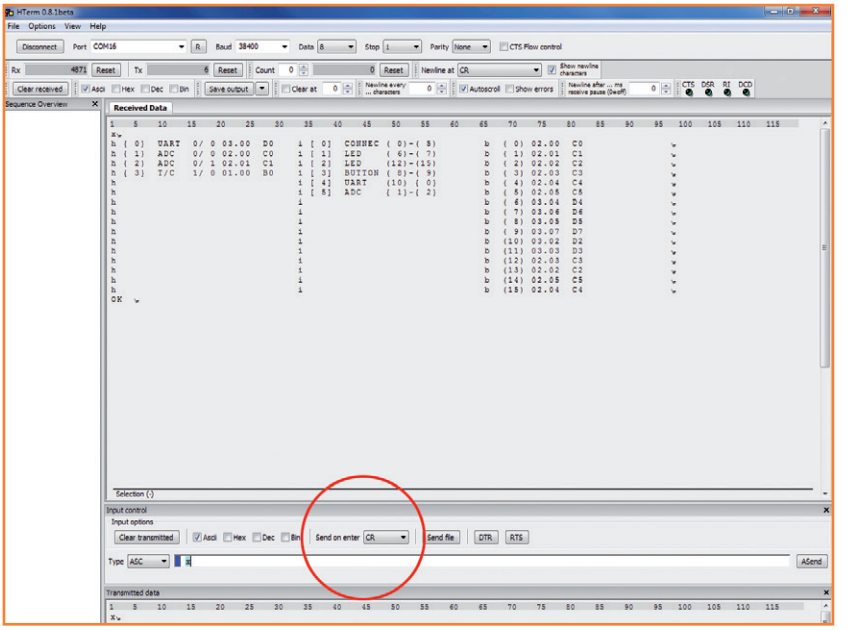


Bild 4.
Im Terminalprogramm HTerm lässt sich einstellen, dass ein <CR> an eine zu versendende Zeichenkette angehängt werden soll.

letzten Heft mit Codezeilen erreicht haben.
Mit der Zeichenkette…

```
L 0 1 + <CR>
```

…schalten wir die LED 1 im LED-Block #0 an (auf unserem Board ist das die gelbe LED).
Mit…

```
B 0 0 ? <CR>
```

…fragen wir den Status des Test-Buttons ab (Button 0 des ersten Button-Blocks #0).
Ein Relais ließe sich schalten mit:

```
R 0 0 + <CR>
```

3-Draht-Schnittstelle

Die eben gezeigte Steuerung setzt voraus, dass wir von außen einen Zugang zu einem UART des Controllers haben; zum Beispiel über herausgeführte RX/TX-Pins. Außerdem muss man bereits über ein weitgehend vollständiges EFL-Controllerfile verfügen, das uns neben den einfachen Pin-I/O- und ADC-Funktionen auch UART-Funktionen zur Verfügung stellt.
Wenn man sich dagegen einem noch unbekannten Controller nähert (etwa weil man ein EFL-Controllerfile entwickeln will, das die anderen Leser nutzen können), wird man vermutlich mit den einfachen I/O-Funktionen beginnen. Ist man nach etwas Studium des Datenblatts in der Lage, zumindest den Level eines Pins zu setzen und zu lesen, und sind zumindest drei GPIO-Pins nach außen geführt (was bei den meisten Boards der Fall sein dürfte), dann ist man schon einen großen Schritt weiter. Denn nun lässt sich das Board ebenfalls über das oben beschriebene Protokoll steuern. Als Übertragungskanal wird dabei eine 3-Draht-Schnittstelle auf Software-SPI-Basis genutzt. Eine der Verbindungen dient als Taktleitung, über einen weiteren Draht fließen Bytes vom Master (PC-Seite) zum Slave (Board). Und auf der dritten Leitung geht es in die umgekehrte Richtung.

Über die SPI-Spezifikation hinausgehend verfügt die 3-Draht-Schnittstelle noch über die Möglichkeit, dass sowohl der Master als auch der Slave die Kommunikation initiieren können. Hierzu wird die eigene „Out"-Leitung einfach auf High gezogen und abgewartet, bis der andere Teilnehmer das mit seiner Leitung genauso macht. Dann beginnt die Kommunikation, bei welcher nach SPI-Art immer der Master den Takt vorgibt und gleichzeitig in beide Richtungen Bytes geschoben werden. Wer keine Daten (mehr) zu senden hat, übermittelt stattdessen das Stopp-Zeichen <LF> = ASCII 10. Falls beide Teilnehmer „10" senden, wird die Kommunikation beendet.

Das Ganze haben wir natürlich schön in einem EFL-Library-Modul namens „ThreeWireInterfaceEFL" gekapselt. Damit ist es nun sehr einfach,

Das BlockProtocol

Alle Kommandos beginnen mit einem einzelnen Zeichen, dann folgen eine oder zwei Dezimalzahlen. Hieraus ermittelt die EFL den Pin, um den es geht. Das abschließende Zeichen bestimmt die jeweilige Aktion.

x
Ausgabe der EFL-Tabellen Map, Blocks, Boardpins (siehe EFL-Zusatzdokument [1])

*p x y +, p x y –, p x y ?, p x y #, p x y **
Controller-Pin Port x Pin y: auf High setzen, auf Low setzen, Abfrage des Levels, ADC-Wert ermitteln (bei einem ADC-Pin), timergesteuertes Blinken

*b x +, b x –, b x ?, b x #, b x **
Boardpin mit Index x in der Boardpin-Tabelle: High, Low, Abfrage, ADC-Wert, Blinken

*i x y +, i x y -, i x y ?, i x y #, i x y **
Pin im Block mit Index x in der Block-Tabelle, Pin-Position y innerhalb des Blocks: High, Low, Abfrage, ADC-Wert, Blinken

*C x y +, C x y –, C x y ?, C x y #, C x y **
Steckverbinder x Pin y: High, Low, Abfrage, ADC-Wert, Blinken

*L x y +, L x y –, L x y ?, L x y **
LED-Block x LED-Position y: High, Low, Abfrage, Blinken

R x y +, R x y –, R x y ?
Relais-Block x Relais-Position y: High, Low, Abfrage

B x y ?
Button-Block x Button-Position y: Abfrage

A x y #
ADC-Block x ADC-Pinposition y: ADC-Wert ermitteln

*
Beenden timergesteuertes Blinken

die Kommunikation vom UART-Interface auf die Software-SPI-Schnittstelle umzustellen.
Die Codezeilen in der Application-Setup-Funktion lauten nun:

```
ThreeWireInterface_LibrarySetup();

BlockProtocol_
LibrarySetup(ThreeWireInterface_Send, 0,
ThreeWireInterface_GetRingbuffer(0));
```

Der BlockProtocol-Library haben wir damit mitgeteilt, dass die Daten über die 3-Draht-Schnittstelle versendet und empfangen werden sollen (man vergleiche die Zeile mit dem BlockProtocol_LibrarySetup-Code oben).

In der Application-Loop-Routine benötigen wir noch den Befehl:

```
ThreeWireInterface_Listen(0);
```

Diese prüft, ob sich der jeweils andere Teilnehmer gerade mit einer auf „High“ gezogenen Leitung meldet. Dann wird die Kommunikation aufgebaut (im Falle des UART-Übertragungskanals haben wir eine solche Listen-Funktion nicht benötigt, da dabei ein Controller-Interrupt erkennt, wenn Zeichen eingehen und die Bytes hiermit automatisch in den Ringbuffer gesetzt werden).

Kontaktaufnahme mit Arduino

Als Demo-Board für das Ganze haben wir uns ein Arduino-Uno-Board ausgesucht; den zugehörigen Code findet man im Projekt ArduinoUnoEFL [4][5]. Als GPIO-Pins, über die wir Zugang zum Board bekommen, verwenden wir PB0, PB1 und PB2, die über den „Digital“-Steckverbinder herausgeführt sind (Digital8 bis Digital10). Ein kleines 3-Pin-Kabelset (z.B. Conrad 741221) leistet dabei recht praktische Dienste (siehe **Bild 5**).

Natürlich benötigen wir nun noch eine PC-Anbindung, denn die wenigsten unter uns dürften über einen Computer verfügen, der eine 3-Draht-Schnittstelle mitbringt ☺. Daher funktionieren wir unseren kleinen Experimentalknoten zu einem Gateway um, der die Daten vom Übertragungskanal „3-Draht“ auf „UART/RS485“ und zurück umsetzt. Die entsprechende Firmware ist natürlich ebenfalls von [4] oder [5] herunterzuladen, das Atmel-Studio-Projekt steckt im Ordner „ExperimentalSPI“.

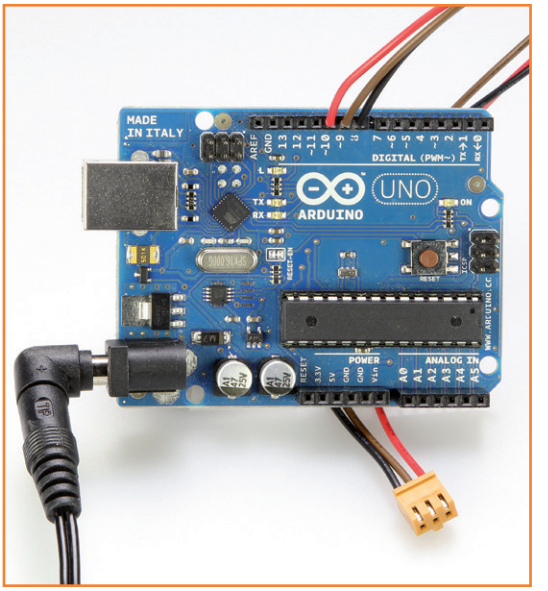


Bild 5.
Über drei Leitungen bekommen wir Zugang zum Arduino-Uno-Board.

Für die Gateway-Funktion gibt es ein eigenes Bibliotheksmodul, nämlich OneToOneGatewayEFL. Mit dem Aufruf von…

```
OneToOneGateway_Engine();
```

…innerhalb der Application-Loop-Funktion wird regelmäßig abgefragt, ob eine Zeichenkette im Ringbuffer des einen Kanals gelandet ist, die mit <CR> abgeschlossen wurde. Ist dies der Fall, dann wird die Zeichenkette über den jeweils anderen Übertragungskanal weitergesendet. Anschließend wird der Ringbuffer dieses anderen Übertragungskanals auf eingegangene Zeichen überprüft und so fort.

Es versteht sich schon fast von selbst, dass das Gateway-Modul ebenfalls unabhängig von den verwendeten Übertragungskanälen programmiert wurde und daher flexibel in allen möglichen Gateway-Anwendungen einzusetzen ist. Die OneToOneGateway_LibrarySetup-Funktion nimmt gleich zwei Parameter-Tripel entgegen, die festlegen, welche Übertragungskanäle über das Gateway gekoppelt werden sollen. In unserem Fall lautet der Aufruf:

```
OneToOneGateway_
LibrarySetup(UARTInterface_Send,
0, UARTInterface_GetRingbuffer(0),
ThreeWireInterface_Send, 0,
ThreeWireInterface_GetRingbuffer(0));
```

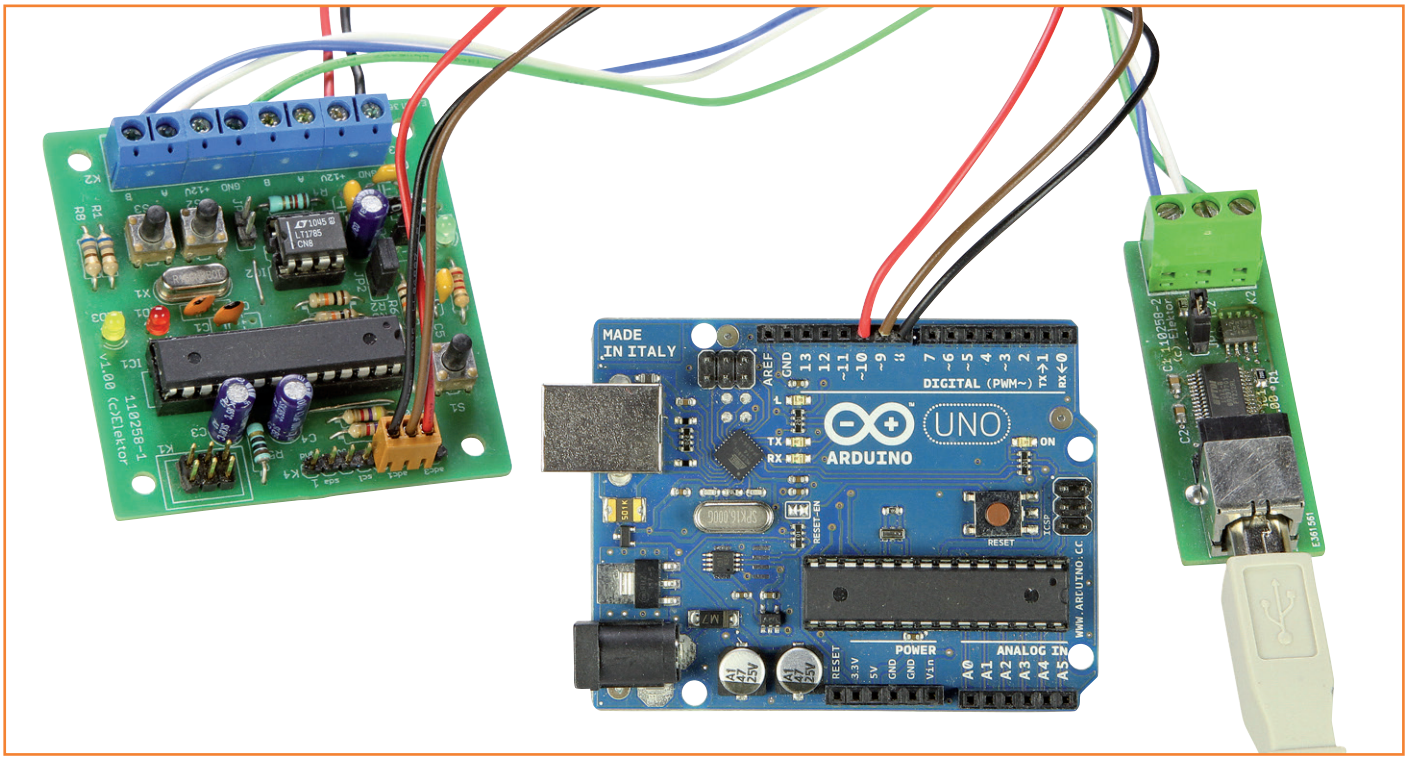


Bild 6.
Der Experimentalknoten dient als Gateway zwischen der 3-Draht-Schnittstelle und UART/RS485.

Wir müssen nun nur noch das Kabelset an den Erweiterungssteckverbinder des Experimentalknotens anschließen (Pins PC0 bis PC2, siehe **Bild 6**) und das zugehörige Hex-File in den Controller des Experimentalknotens flashen.
Die verwendeten Pins für die 3-Draht-Verbindung lassen sich noch auf beiden Boards anpassen. Das 3-Draht-Interface bildet bei beiden Teilnehmern einen eigenen Peripherie-Block und wir erinnern uns, dass das Boardfile die Verdrahtung der Peripherieblocks mit den Controllerpins kapselt. Die entsprechenden Setups der 3-Draht-Schnittstelle mit einer Zuordnung der genutzten Pins findet man demnach jeweils in der Board_Init-Funktion in der Datei Board.c.

Es funzt!

In unser Terminalprogramm geben wir zuerst wieder das Kommando „x“ ein. Das Arduino-Board übermittelt jetzt brav die EFL-Variablen über die 3-Draht-Schnittstelle, was freilich etwas gemächlicher (etwa mit 9600-Baud-Geschwindigkeit) geht wie vorher.
Mit dem Befehl

```
C 0 13 + <CR>
```

können wir den Pin 13 des „Digital“-Steckverbinders auf High setzen, und da beim Arduino-Uno-Board hier auch eine LED angeschlossen ist, sieht man das Ergebnis sofort.
In den nächsten Heften geht es mit neuen EFL-Projekten weiter. Wer Anregungen hat oder selbst etwas beitragen möchte, kann sich wie immer melden unter redaktion@elektor.de.
Den jeweils neuesten EFL-Code findet man auf der Elektor.Labs-Seite unter [5].

(130154)

Weblinks

[1] www.elektor.de/120668
[2] www.elektor.de/100576
[3] www.elektor.de/120296
[4] www.elektor.de/130154
[5] www.elektor-labs.com/efl
[6] www.der-hammer.info/terminal
[7] www.elektor.de/110258
[8] www.elektor.de/110428
[9] www.elektor.de/110405
[10] www.elektor.de/120097

Ortungsmelder

Führt direkt zum Flugmodell!

Von **Robert Budniak**
(Australien)

Jedes funkgesteuerte Flugzeugmodell ist irgendwann schon einmal außerhalb der ihm gestatteten (Flugplatz-)Grenzen notgelandet. Manchmal liegt es dann gut sichtbar auf einer Wiese, meistens aber versteckt sich das kostbare Modell im hohen Gras oder in einem Baum. Also ist ein Peilsender/-empfänger nötig, der schnurstracks zum verunglückten Modell führt.

Zwar gibt es schon eine Reihe von brauchbaren Ortungsmeldern (googeln Sie mal *lost model finder*), mit denen man verlorengegangenen Flugmodellen auf die Spur kommen kann, aber besser als brauchbar geht immer! Die Vorgaben für das Projekt waren:

- Leichtgewichtssender für das Flugzeug
- Ersatzbatterie für den Fall, dass die Versorgung durch die Hauptbatterie unterbrochen ist
- mit kommerziellen, genehmigten UHF/ISM-Funkmodulen
- Reichweite mindestens 200 m
- Handheld-Empfänger mit Funkpeilung (RDF) des Modells.

Aus diesen Anforderungen entstand die hier beschriebene Schaltung.

Wenn Sie besser im Modellfliegen als im Löten sind, sollten Sie in Erwägung ziehen, die Elektronik in der Gemeinschaft ihres Modellbauvereins aufzubauen – es lohnt sich!

Eigenschaften

- Günstiges Preis/Leistungsverhältnis
- Sender in SMD, Empfänger mit bedrahteten Bauteilen
- Geeignet für die meisten 433-MHz-ISM-Module
- Programmierbare individuelle Rufsignale für jeden Sender
- Minimale Reichweite: 200 m
- Maximale Reichweite abhängig von Gelände und den eingesetzten Sender/Empfängermodulen
- Yagi-Richtantenne mit vier Elementen am Empfänger

Der Sender

In Europa ist die UHF-Frequenz von 433 MHz freigegeben als *Instrument, Scientific and Medical Band* (ISM) für den Funkverkehr mit geringer

Reichweite mit *Short Range Devices* (SRD). Da ein großer Bedarf an einfachen und preiswerten Transmittern für dieses Band besteht, gibt es eine Vielzahl von Herstellern, die solche Module auf dem Markt anbieten. In diesem Projekt werden Module verwendet, die mit *Amplitude Shift Keying* (ASK) arbeiten. Diesen Modulen verschiedener Hersteller ist scheinbar eine standardisierte Pin-Konfiguration eigen. Das in der Stückliste genannte Modell stellt deshalb nur eine Empfehlung dar.

Wie Sie in der Schaltung in **Bild 1** sehen, kommunizieren das HF-Modul und der Mikrocontroller über nur einen DATA-Pin. Ich habe mich bei diesem Projekt für einen PICAXE08M entschieden, weil dieser Mikrocontroller leicht verfügbar und die Programmiersprache leicht zu erlernen ist und der Chip keine umfangreiche und teure Programmierumgebung benötigt. Wirklich ein ideales System für kleine und wenig komplexe Projekte! Alles zum Umgang mit diesem Controllertyp finden Sie auf der PICAXE-Webseite [3].

Der Code (**Listing 1**) ist nur ein paar Zeilen lang. Er erzeugt drei kurze 500-Hz-Töne und dann eine Pause von etwa 2,5 s. Wenn Sie mehrere Transmitter aufbauen möchten, können Sie durch Variationen in dieser Zeile jedem Modell seinen unverwechselbaren Hilferuf verpassen.

Normalerweise wird der Sender vom Empfänger des Flugmodells versorgt. Da aber bei einem Unfall die Verbindung zur Batterie leicht unterbrochen werden kann, verfügt die Schaltung über eine Reservebatterie. Dies ist eine kleine, wieder aufladbare LiPo-Batterie mit 130 mAh Kapazität, wie sie oft bei Indoor-Flugmodellen zum Einsatz kommt. Dieser Akku ist billig im (Online-)Handel erhältlich. Normalerweise verwendet man Dioden für die automatische Wahl der Stromversorgung. Doch bei den niedrigen Spannungen in diesem Projekt (5 V und 3,7 V) würde selbst der geringe Spannungsabfall an einer Schottky-Diode von rund 0,4 V einen beträchtlichen Anteil der zur Verfügung stehenden Energie (und damit Zeit, das Modell zu finden) verheizen. Bei der Suche in Online-Datenbanken fand ich kleine Signal-Transistoren vom Typ DTB123YK (T1, T2), die über Basis-Emitter- und Basis-Widerstände im Chip verfügen. Sie können sehr gut die Spannung für unsere Elektronik schalten und verursachen dabei einen Spannungsabfall von nur 100 mV. Der einzige Nachteil: Wenn die Haupt-Batterie abgeklemmt wird, übernimmt die Backup-Batterie automatisch. Deshalb sollte man nie vergessen, die Backup-Batterie auch abzuklemmen, wenn das Flugmodell nicht in Gebrauch ist. Damit der Ortungsmelder dauerhaft an seinem Platz im Modell verbleiben kann (er ist billig genug, um jedes Modell mit einem Sender auszustatten), wurde ein An/Aus-Schalter sehr platzsparend mit einem Schalter in der 2,5-mm-Klinkenbuchse für den Ladevorgang realisiert. Steckt ein Stecker in der Buchse, ist die Verbindung zwischen Elektronik und Backup-Akku getrennt. Wenn man den

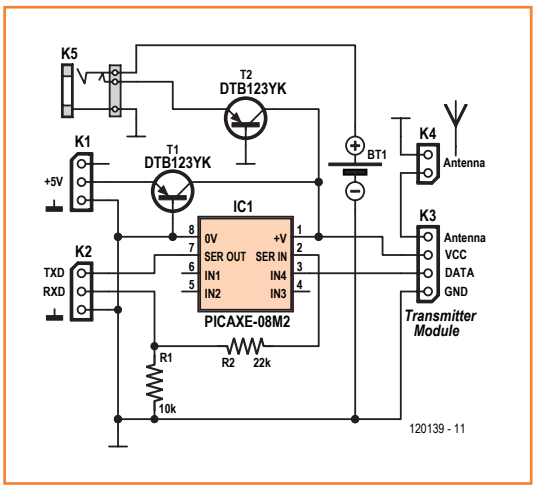


Bild 1.
Der Schaltplan des Senders zeigt nicht viel mehr als den programmierten PICAXE-Mikrocontroller.

Listing 1. PICAXE TX code

```
main:
sound 4,(0,10,120,10,0,10,120,10,0,10,120,10,0,10)
high 4
pause 2300
goto main
```

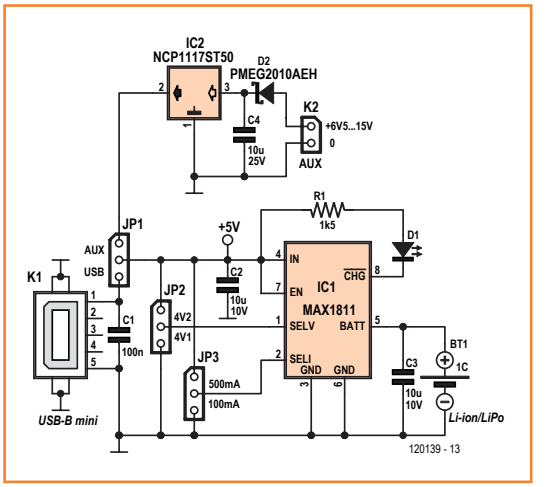


Bild 2.
Eine mögliche Ladeschaltung für den LiPo/Li-Ionen-Akku im Sender, die entweder extern (6,5...15 VDC) oder über den USB mit Strom versorgt wird, je nachdem, wie JP1 gesteckt ist.

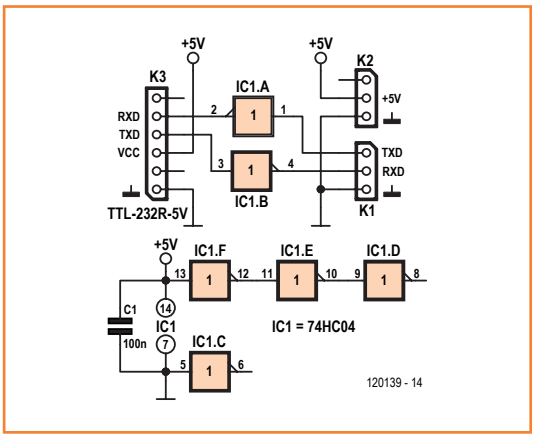


Bild 3.
Wenn Sie einen TTL-zu-RS232-Adapter von FTDI zwischen den Sender-Anschlüssen TX/RX und einem Mikrocontroller verwenden, ist ein solcher Inverter erforderlich.

Stecker mit einem Ladegerät verbindet, dann kann man den Akku laden.

Volle Ladung!

Eine geeignete Schaltung für das Ladegerät ist in **Bild 2** dargestellt. Zwei Eingänge bestimmen den Modus, in dem das Ladegerät arbeitet. Der eine, SELV, legt die Reglerspannung (4,1 V oder 4,2 V, Jumper JP2) fest, der andere, SELI, den Ladestrom (100 mA oder 500 mA, Jumper JP3). Ein nettes Feature dieses ICs ist die Fähigkeit, einen nahezu toten Akku vor dem Laden neu zu formieren. Der Enable-Eingang (EN) wird nicht verwendet und liegt dauerhaft an der Versorgungsspannung. Das Datenblatt besagt, dass der Chip per USB-Versorgung mit einer minimalen Spannung von nur 4,35 V betrieben werden kann. Bei höheren Eingangsspannungen (der MAX1811 kann 6,5 V verkraften) wird der Ladestrom limitiert, um die Temperatur auf einem sicheren Niveau zu halten. Für den Fall, dass nur eine Quelle mit höherer Spannung zur Verfügung steht, ist ein 5-V-Low-Dropout-Regler hinzugefügt (IC2). Mit einem Jumper wählt man die Eingangsspannung für den MAX1811 (JP1: AUX oder USB). Schließen Sie an K2 keine Spannungsquelle an, wenn der USB-Anschluss als Eingangsquelle gewählt ist. Der Eingangskondensator C2 wurde doppelt so groß, der Ausgangskondensator viermal so groß wie im Datenblatt des MAX1811 vorgeschrieben gewählt. So ist der Lader bestens gefeit gegen Rauschen und Instabilitäten.

Das PICAXE-Interface

Der PICAXE-Chip wird über die RXD/TXD-Pins programmiert. Dies kann mit einem TTL-zu-RS232-Adapter von FTDI in Zusammenarbeit mit dem *PICAXE Programming Editor* geschehen. Allerdings ist dann für die Schnittstelle eine Inverterschaltung wie in **Bild 3** erforderlich.

Der Empfänger

Der Empfänger basiert auf einem zum Sender komplementären Modul. Ein wenig mehr Sorgfalt ist bei der Auswahl des Moduls nötig, um sicherzustellen, dass es auch die richtigen Signale für unsere Schaltung liefert.

Wie **Bild 4** zeigt, liegt der erste Ausgang des Empfänger-Moduls auf dem DATA-Pin 14. Das Signal passiert den Kleinsignal-FET T1, dessen Ausgang den Piezo-Transducer Bz1 steuert. Bz1 muss ein Piezo ohne Elektronik sein, also kein Buzzer! Sie nehmen (hoffentlich) die Töne und Pausen des Senders wahr, das persönliche „Rufzeichen“, das Sie in Ihr Flugmodell programmiert haben.

Der zweite Ausgang des Senders führt das RSSA-Signal an Pin 13. Es handelt sich um eine Spannung, die proportional zur Stärke des empfangenen Signals ist und benutzt wird, um die Verstärkung automatisch zu regeln (AGC). Dieses Signal wird hier verwendet, um den LED-Balken LM3914 auszusteuern. Das RSSA-Signal des Prototyps variierte zwischen 0,4 V und 2 V, die obere und untere Grenze des LED-Treibers wurden auf diese Werte festgelegt. Hier ist nicht der Ort, um auf die Details des LM3914 einzugehen, dieses IC wurde unendlich viele Male in Selbstbauprojekten in den letzten Jahrzehnten eingesetzt. Wenn Sie eine detaillierte Anleitung wollen, besuchen Sie einmal Daves EEvblog # 204 [1].

Der Empfänger wird von vier (am besten wiederaufladbaren) AA- oder AAA-Batterien versorgt, doch mit drei Trockenzellen sollte er auch funktionieren. Obwohl laut Datenblatt des Senders die

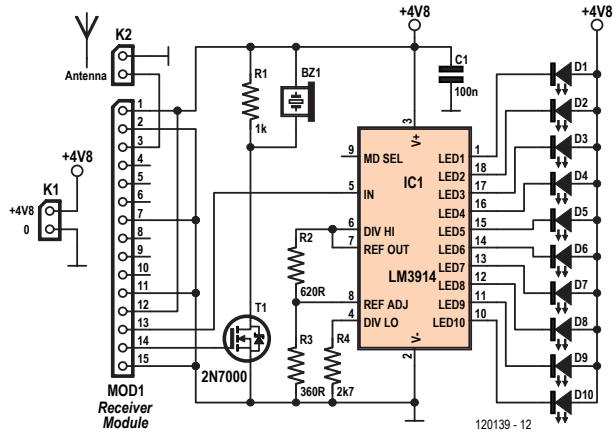


Bild 4.
Der Empfänger des Ortungsmelders verwendet den guten alten LED-Balken LM3914. Der 433-MHz-Empfänger wird auf MOD1 gesteckt.

maximale Spannung 5 V betragen soll, arbeitet er auch mit bis zu 7 V bestens.
Der Receiver ist auch mit einer Antenne verbunden. Es handelt sich um eine Yagi-Antenne mit vier Elementen. Diese Antenne wurde gewählt, da sie trotz ihrer Einfachheit gute Richtwirkung erzielt, eine Eigenschaft, die für unser Projekt natürlich von essentieller Bedeutung ist.

Aufbau

Der **Sender** ist auf einer doppelseitigen Platine (**Bild 5**) aufgebaut und verwendet SMDs. Für die externen Verbindungen werden Stiftleisten im Zehntelzollraster eingesetzt (mit Ausnahme von K3). Das Layout kann kostenlos heruntergeladen werden [2]. Der PICAXE-Controller muss nicht direkt verlötet, sondern kann auch in eine DIL-Fassung eingesetzt werden. Die beiden Transistoren müssen nur eingesetzt werden, wenn man auch den Akku-Backup wünscht. Schließen Sie das HF-Modul noch nicht am mit K3 markierten Footprint an. Jetzt können Sie den Sender testen, indem Sie den Piezo zwischen DATA und Masse anschließen. Nach dem Einschalten sollte das Rufzeichen zu hören sein.
Das HF-Modul kann mit einem kurzen Flachbandkabel, ICD-Verbinder und Stiftleiste angeschlossen werden. Alternativ kann es – wie in **Bild 6** gezeigt - flach über dem Mikrocontroller-Board montiert werden. Stellen Sie sicher, dass sich die Unterseite der Platine und das Modul nicht berühren.
Der letzte Teil ist der Bau einer Viertel-Lambda-Antenne. Ein Stück steifen Drahtes von 173 mm ([300/f]/4) ist ausreichend (etwa der Kern eines Ethernet-Kabels). Das Mikrocontroller-Board, das HF-Modul und die Backup-Batterie (falls verwendet) werden nach einem Funktionstest in einem kurzen Stück Schrumpfschlauch untergebracht.
Das Platinenlayout für den **Empfänger** ist in **Bild 5** zu sehen. Es handelt sich um eine einseitige Platine für bedrahtete Bauteile, deren Bestückung überhaupt keine Probleme bereiten sollte. Das Funkmodul ist vertikal angeordnet. Platine und Batteriehalter für die drei oder vier AA(A)-Zellen können auf dem Längsträger hinter dem Reflektor der Yagi montiert werden. Die Dipol-Elemente der Yagi-Antenne werden am Eingang des Funkmoduls mit dünnem 50-Ω-Koaxialkabel wie RG174/U oder -/CU angeschlossen. Das Kabel sollte so kurz wie möglich sein, um übermäßige Verluste zu verhindern. Verwenden Sie keinesfalls abgeschirmtes Audiokabel!
Das (optionale) **Ladegerät** ist auf der Platine in **Bild 7** aufgebaut. Vergessen Sie nicht, die Jumper entsprechend Ihren Anforderungen zu stecken. Die 4-Element-Antenne ist völlig Marke Eigenbau. Es gibt verschiedene Ausführungen im Internet, und man könnte mechanisch qualifizierte Mitglieder in Ihrem lokalen Modellbau-Verein dazu überreden, ein paar anspruchsvolle Antennen zu bauen. Der Autor hat den Dipol aus Kleiderbügeldraht hergestellt, ein beliebiger starrer (Installations-)Draht funktioniert genauso gut. Schneiden Sie den Dipol mit den Maßen wie in **Bild 8** zurecht. Der Autor hat seinen Prototyp in eine Art Doppelstegplatte mit einer Stärke von etwa

Stückliste

Sender

Widerstände:
R1 = 10 k 1%, SMD0805
R2 = 22 k 1%, SMD0805

Halbleiter:
IC1 = PICAXE-08M2 (programmiert)
T1,T2 = DTB123YK

Außerdem:
K1,K2,K5 = 3-poliger Pfostenverbinder, 0,1''
K3 = 4-poliger Pfostenverbinder, 0,1'' (optional, siehe Text)
K4,BT1 = 2-poliger Pfostenverbinder, 0,1''
Sender-Modul (an K3), ASK, 433 MHz (ISM-Band), geprüft: Quasar Electronics QAM-TX1 (433 MHz), Farnell 1304024.
LiPo-Batterie, 3,7 V, 130 mAh
Platine 120139-1

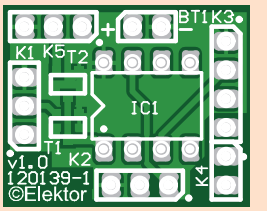
Empfänger

Widerstände:
R1 = 1 k
R2 = 620 Ω, 1%
R3 = 360 Ω, 1%
R4=2k7

Kondensator:
C1 = 100 n, RM5 oder RM7,5

Halbleiter:
D1...D10 = LED, orange, 2,5x5 mm rechteckig, 20 mA
T1 = 2N7000
IC1 = LM3914

Außerdem:
BZ1 = Piezo-Element mit Drahtanschlüssen (nicht auf der Platine) Farnell/Newark 1193640
K1,K2,(BZ1) = 2-poliger Pfostenverbinder, 0,1''
(MOD1) = 15-poliger SIL-Verbinder, gerade, 0,1''
MOD1 = AM SuperHet Receiver, QAM-RX3 (433 MHz), RS Components 742-4484
Batteriehalter für 3 oder 4 AA(A)-Batterien, siehe Text
Platine 120139-2



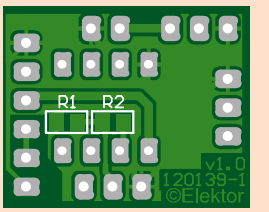
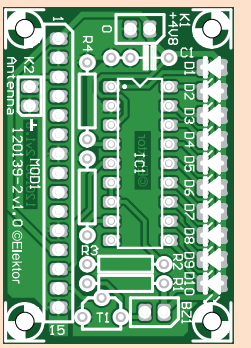


Bild 5. Die Transmitter-Platine ist doppelseitig und mit SMDs bestückt, auf die einseitige Empfängerplatine dagegen kommen nur bedrahtete Bauteile. Der Sender ist hier anderthalb Mal so groß abgebildet, wie er wirklich ist.



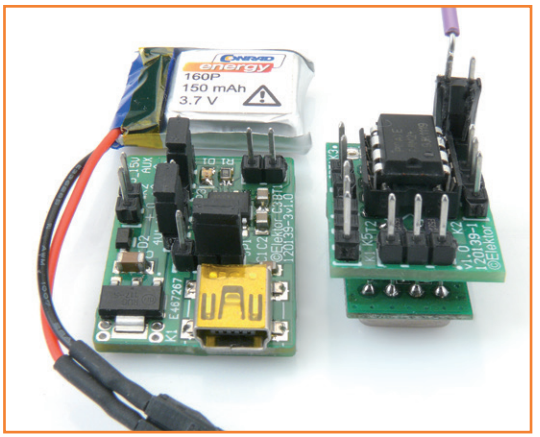


Bild 6.
Die Platine des LiPo-Laders auf der linken und des Senders auf der rechten Seite. PICAXE-Platine und Transmittermodul können ohne Kabel direkt miteinander verlötet werden, wenn man die vier Pins einfach und für K3 passend umbiegt.

3 mm eingebaut. Der gewünschte Abstand der Elemente war nicht besonders präzise zu realisieren, aber die Antenne hat in der Praxis doch ganz gut funktioniert.
Das Elektor-Labor hat die Yagi-Antenne mit Abschnitten von 2,5-mm²-Installationsdraht auf einer 570x53x12 mm großen Sperrholzplatte aufgebaut. Die genauen Positionen zeigt Bild 8. Der Längsträger ist lang und breit genug, um das Empfängerboard und das Batteriefach zu tragen. Die Antenne sollte etwa 7 dB gewinnen, so dass wir den Sender in einem Abstand von etwa 300 m im bebauten Gelände rund um das Elektor-Labor finden konnten.
Verbinden Sie den Dipol der Antenne mit dem Empfänger über das zuvor installierte Kabel. Nanovolt-Pedanten könnten einwenden, ein Balun wäre zur Anpassung des symmetrischen Dipols an das asymmetrische Kabel erforderlich. Doch diese Arbeit können wir uns sparen, die Angelegenheit funktioniert auch ohne Balun tadellos!

Test

Glücklicherweise ist keinerlei Kalibrierung der Schaltung erforderlich. Belassen Sie den Sender so, wie er ist. Wenn Sie den Empfänger einschalten, sollte weißes Rauschen zu hören sein. In der LED-Anzeige leuchtet höchstens die unterste LED. Schalten Sie nun den Sender ein. Das Rufzeichen sollte zu hören sein und die Balkenanzeige sich synchron zum Rufton auf und ab bewegen. Die höchste LED sollte bei einer Entfernung von etwa 3 m zum Sender aufleuchten.

Im Einsatz

Es bedarf ein wenig Übung, um erfolgreich mit dem Ortungsmelder umgehen zu können. Am besten überredet man einen verschwiegenen Freund, das Modell irgendwo im unübersichtlichen Gelände zu verstecken. Dann gehen Sie nach draußen und suchen es.
Die Antenne besitzt eine Richtwirkung, das „scharfe“ Ende ist das mit der größten Empfindlichkeit. Bei der Suche nach dem Sender halten Sie die Antenne vor sich und vollführen eine 360°-Wende. Lauschen Sie dem Rufzeichen und beobachten Sie, bei welchem Winkel die Balkenanzeige am höchsten ausschlägt. Gehen Sie in diese Richtung und wiederholen Sie ab und an die Wende. Sie gelangen so immer näher zum Sender und erhalten bald einen maximal möglichen Ausschlag auf der LED-Skala. Nun halten Sie das unempfindliche (stumpfe) Ende der Antenne in die Richtung, in der Sie den Sender vermuten und achten nicht mehr auf das maximale, sondern das minimale Signal. Irgendwie schleichen Sie sich so von hinten an Ihr Modell heran.
Sie können den Antennendipol parallel zum Boden oder senkrecht oder in jede andere Richtung halten. Die letzten Meter bei der Suche nach dem Modell sind die härtesten, vor allem, wenn es im hohen Gras oder in einem Baum gelandet ist. Man weiß halt nie, wo sich diese Modelle verstecken.

(120139)

[1] Dave’s EEVBlog # 204: www.youtube.com/watch?v=iIKGvHjDQHs&feature=player_embedded
[2] Projektseite: www.elektor.de/120139
[3] Website von PICAXE: www.picaxe.com/

Stückliste

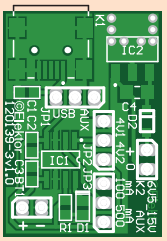


Bild 7. Das Ladegerät ist auf einer doppelseitigen SMD-Platine aufgebaut.

LiPo-Lader (optional)

Widerstand:
R1 = 1k5 SMD 0805

Kondensatoren:
C1 = 100 n, SMD0805 X7R
C2,C3 = 10 µ, 10 V, SMD 0805, X7R
C4 = 10 µ, 25 V, SMD 1206, Y5V

Halbleiter:
D1 = LED, rot, SMD 0805
D2 = PMEG2010AEH, Farnell/Newark 1510673
IC1 = MAX1811ESA+, Farnell/Newark 1593327
IC2 = NCP1117ST50T3G, Farnell/Newark 2112617

Außerdem:
K1 = Mini-USB-Buchse Typ B, gerade, SMD
K2,(BT2) = 2-poliger Pfostenverbinder, 0,1’’
JP1,JP2,JP3 = 3-poliger Pfostenverbinder, 0,1’’, mit Jumper
PCB 120139-3

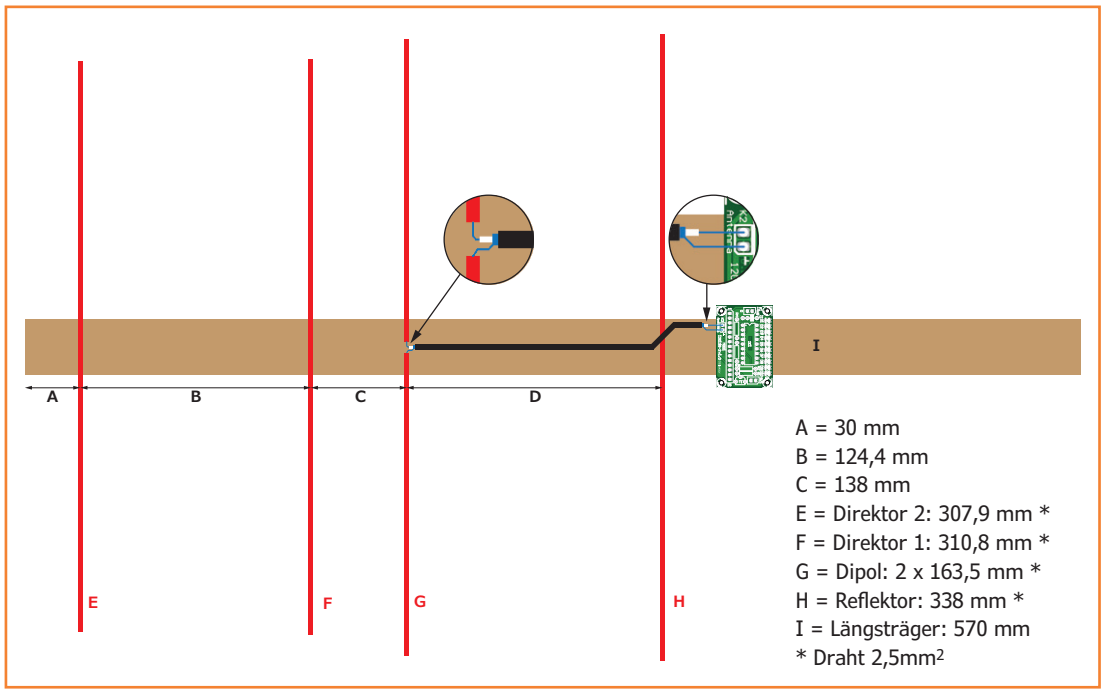
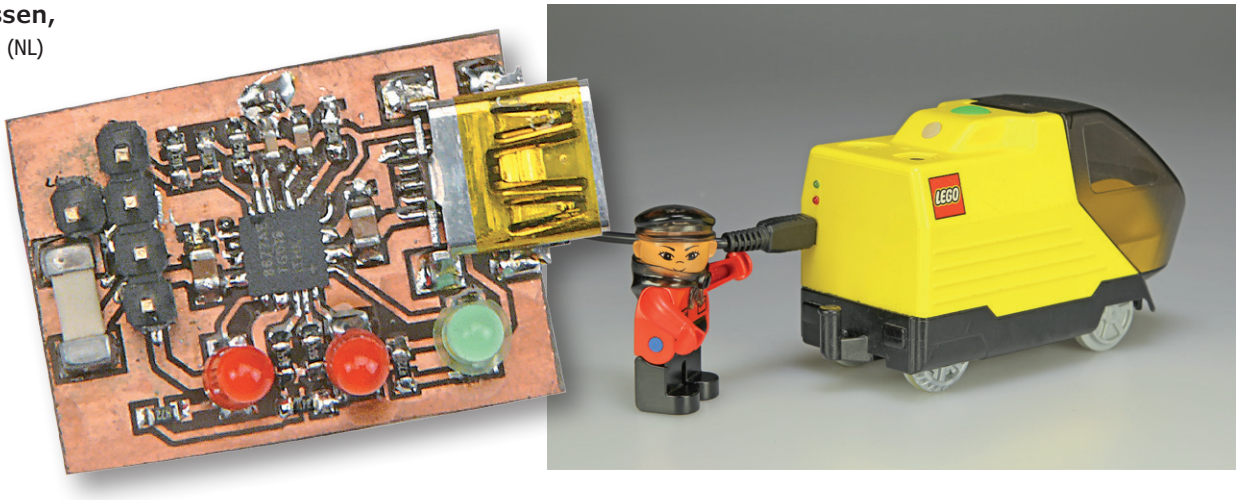


Bild 8.
Aufbau und Abmessungen der experimentellen Yagi-Antenne mit der Empfänger-Elektronik hinter dem Reflektor auf einem hölzernen Längsträger montiert. Die Direktoren sowie Dipol und Reflektor wurden aus starrem 2,5-mm²-Installationsdraht hergestellt und nach sorgfältiger Zentrierung am Längsträger mit Kabelbindern befestigt. Sicherheitshalber sollten die Spitzen mit einem stumpfen Gegenstand oder Kitt entschärft werden. Die Verstärkung beträgt etwa 7 dB.

Alte Akkus neu genutzt!

Lithium-Ion-Akkus laden ohne Mikrocontroller

Von **Fons Janssen,**
Maxim Integrated (NL)



Der Wiederverwendung von Li-Ion-Akkus, die aus veralteten oder defekten mobilen Geräten stammen, steht meistens eine Hürde im Weg: Die Ladeschaltung ist im ausgemusterten System integriert, der zum Gerät gehörende Ladestecker genügt allein nicht. Hier hilft der Bau eines universellen Li-Ion-Akkuladers weiter.

Wer kennt das nicht: In Schubladen oder Schränken fristen mobile, akkubetriebene Geräte ihr Dasein, die nicht mehr auf dem Stand der Zeit sind. Auch ältere mobile Geräte wurden oft schon von Li-Ion-Akkus mit Energie versorgt. Dieser moderne Akku-Typ hat viele Vorteile, er kann in fast beliebigen Bauformen produziert werden, und was die Energiedichte betrifft, ist er anderen Akku-Typen (NiCd, NiMH) überlegen.

Was bleibt zu tun mit dem alten MP3-Player oder dem Mobiltelefon, das seinen Platz schon längst einem neuen Modell überlassen musste? Platine und Gehäuse haben höchstens Elektroschrottwert, doch der Akku könnte noch brauchbar sein. Vielleicht erwacht er in einem Kinderspielzeug zu neuem Leben? Elektroniker sind erfindungsreich, sie erschließen für ausgebaute Li-Ion-Akkus schnell neue Wirkungskreise. Ein Vorbild ist der Autor dieses Beitrags, er hat die drei Mignon-Zellen einer mit Lego konstruierten Modellbahn (Titelfoto) durch einen noch tauglichen Li-Ion-Akku ersetzt. Doch da ist noch ein Problem: Wenn die gespeicherte Energie zur Neige geht, muss der Akku aufgeladen werden. Die ursprüngliche Ladeelektronik befindet sich meistens auf der Platine des ausgemusterten Geräts. Der Ausbau der Bauelemente und der externe Wiederaufbau der Ladeelektronik sind nicht möglich, zumal mobilen Geräten in aller Regel kein Schaltplan beiliegt. Die Lösung ist der Bau eines universellen Li-Ion-Akkuladers!

Schaltung

Unser Akkulader arbeitet mit dem von Maxim entwickelten „Li-Ion-Lademanager“ MAX8677A, das interne Funktionsschema ist in **Bild 1** dargestellt. Das IC erfüllt seine Aufgabe autonom, so dass

der Akkulader ohne Mikrocontroller und folglich ohne Software auskommt. Den Ladezustand des Akkus zeigen verschiedenfarbige LEDs an.
Der MAX8677A ist äußerst flexibel einsetzbar, er ist mit einem so genannten „Smart Power Selector“ ausgestattet (**Bild 2**). So nennt der Hersteller einen dreifachen internen Schalter, der die Lade- und Entladeströme in die notwendigen Richtungen lenkt. Das IC kann die verfügbare Leistung so verteilen, dass der Akku optimal geladen wird, auch während das Gerät in Betrieb ist. Falls der Strombedarf des Geräts den Strom des Netzteils übersteigt, schaltet das IC den Akku hilfsweise zu. Solange das Gerät nicht am Stromnetz angeschlossen ist, wird es allein aus dem Akku versorgt.
Der MAX8677A kann den Strom einem USB-Port entnehmen (Anschlüsse 15 und 16, USB), wobei der Strom auf 500 mA begrenzt ist. Das ist die Obergrenze des Stroms, der über einen USB-2.0-Port fließen kann. Wenn das IC über die Anschlüsse 2 und 3 (DC) an einem Netzteil betrieben wird, ist die Obergrenze bis 2 A einstellbar.
Unser Li-Ion-Lader, dessen Schaltung **Bild 3** zeigt, benutzt den Eingang DC, so dass die Stromgrenzen flexibel wählbar sind. Wenn der Eingangsspannungsbereich 4,1...6,6 V überschritten wird, unterbricht der MAX8677A den Eingangsstrom,

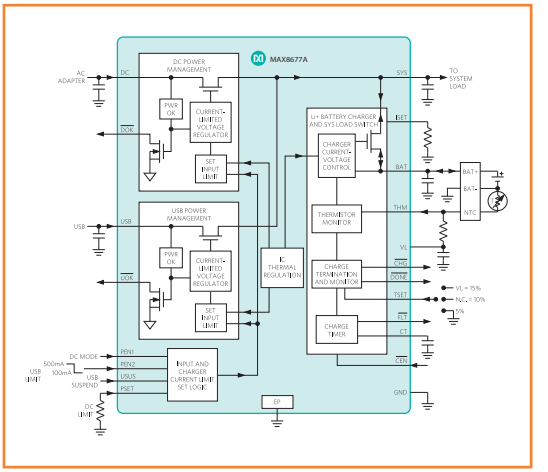


Bild 1.
Internes Funktionsschema des MAX8677A von Maxim.

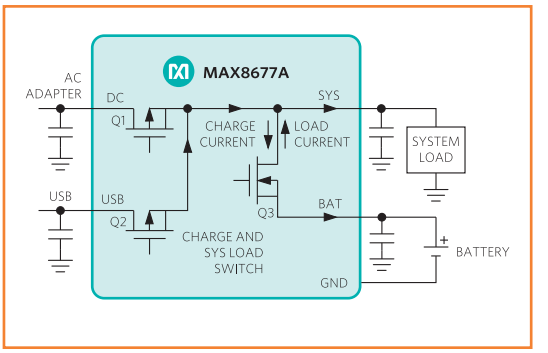


Bild 2.
Der „Smart Power Selector“ verteilt die Lade- und Entladeströme.

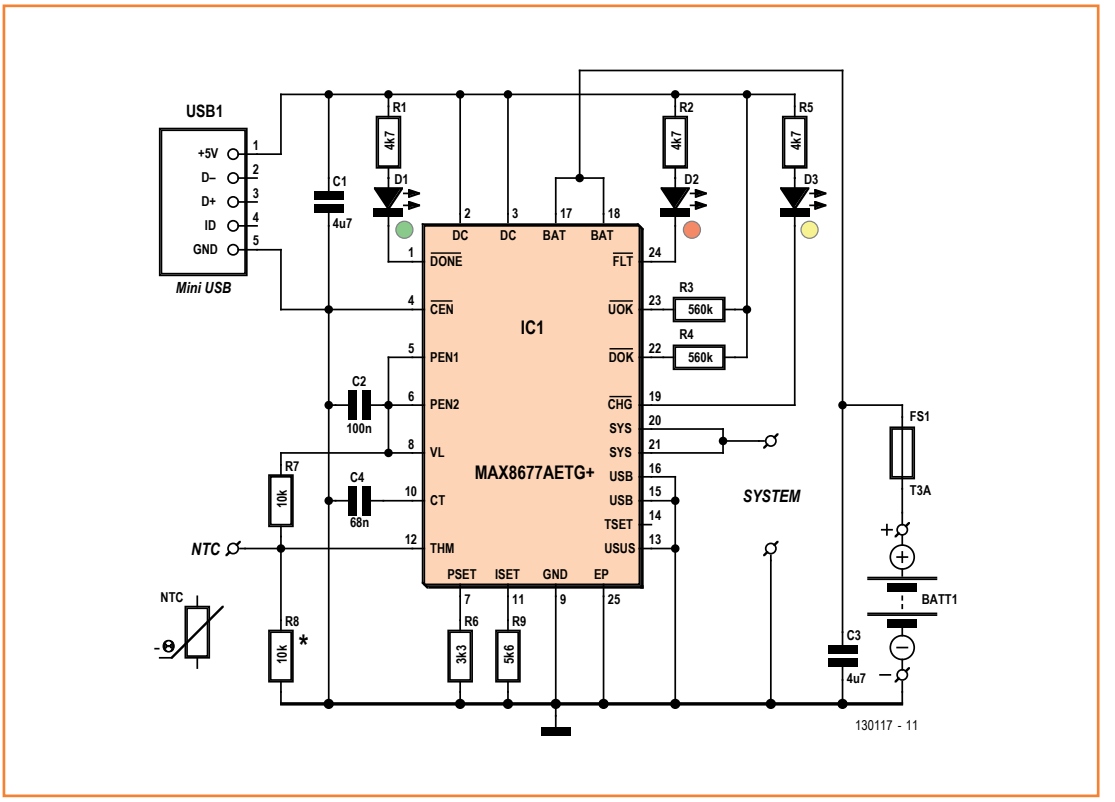


Bild 3.
Mittelpunkt der Schaltung ist der „Li-Ion-Lademanager“ MAX8677A.

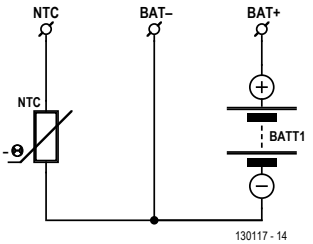


Bild 4.
In dreipoligen Li-Ion-Akkus befindet sich meistens ein NTC-Widerstand zur Temperaturüberwachung.

um thermischer Überlastung entgegen zu wirken. Kurzzeitigen Spannungsspitzen ist das IC bis 14 V gewachsen.
Die LEDs D1...D3 zeigen den Ladezustand des Akkus an. Wenn LED D3 leuchtet, wird der Akku geladen. LED D1 signalisiert, dass der Akku vollständig geladen ist. Auf einen Fehler weist LED D2 hin, der Akku ist möglicherweise defekt.

Stückliste

Widerstände:
(SMD0603, sofern nicht anders angegeben)
R1,R2,R5 = 4k7
R3,R4 = 560 k
R6 = 3k3
R7 = 10 k
R8 = 10 k (nur bei Akkus ohne NTC)
R9 = 5k6

Kondensatoren:
(SMD0603, sofern nicht anders angegeben)
C1,C3 = 4µ7 (SMD0805)
C2 = 100 n
C4 = 68 n

Halbleiter:
D1 = LED grün, 3 mm
D2 = LED rot, 3 mm
D3 = LED gelb, 3 mm
IC1 = MAX 8677AETG+ (24-Pin TQFN)

Außerdem:
USB1 = USB-Mini-Buchse SMD (z. B. Molex 67803-8020, RS-Components 720-6618)
FS1 = Sicherung SMD, Wert abhängig vom Akku-Typ (z. B. LittleFuse Nanofuse 3 AT, Farnell 1596930RL)
Platine 130117-1, Layout auf der Projektseite [2]

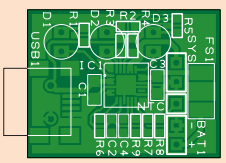


Bild 5.
Die Platine hat Miniformat, so dass sie bequem in das zu versorgende Gerät passt.

Einstellbar sind beim MAX8677A sowohl der maximale Ladestrom als auch der maximale Eingangsstrom. Der Eingangsstrom muss natürlich stets größer als der Ladestrom sein, anderenfalls kann der maximale Ladestrom nicht erreicht werden. Das IC misst die Ströme mit einem Widerstand als Stromfühler, hierfür gilt:

Maximaler Ladestrom = I_{CHGMAX} = 3000/R_{ISET} = 3000/R9 = 3000/5k6 = 535 mA

Maximaler Eingangsstrom = I_{DCMAX} = 3000/R_{PSET} = 3000/R6 = 3000/3k3 = 909 mA

Abhängig von der Leistung des Netzteils, der Stromaufnahme des Geräts und dem gewünschten Ladestrom sind auch andere Werte wählbar. Der MAX8677A unterstützt Ladeströme bis 1,5 A. Die Verbindung zu einem USB-Port stellt eine USB-Mini-Buchse her, so dass viele gängige Ladeadapter anschließbar sind. An dieser Buchse beträgt die Eingangsspannung +5 V. Wenn ein Steckernetzteil zum Einsatz kommt, muss die Obergrenze des Eingangsstroms an das Netzteil angepasst werden. An Netzteilen, die Gleichströme von 1 A oder höher liefern können, arbeitet unser Li-Ion-Akkulader problemlos.

Temperaturfühler

Häufig sind in Li-Ion-Akkus NTC-Fühler eingebaut, sie sollen verhindern, dass der Akku bei zu hohen oder zu niedrigen Temperaturen geladen wird. Deshalb hat der Akku drei Anschlüsse: Pluspol BAT+, Minuspol BAT– sowie ein Anschluss für den internen NTC-Widerstand (**Bild 4**). Doch Vorsicht, bei manchen Li-Ion-Akku-Typen liegt der dritte Anschluss an einem gewöhnlichen Widerstand. Der konstante Wert dient zur Identifikation des Akku-Typs, er ist von der Temperatur unabhängig. Wenn ein NTC-Widerstand im Akku eingebaut ist, wird er an Pin THM gelegt. Von THM führt außerdem ein Widerstand (in diesem Fall R7) zur Referenzspannung V_L, so dass ein Spannungsteiler entsteht. Ist der Wert des Widerstands gleich dem Wert des NTC bei 25 °C, beträgt die Spannung 0,5 V_L bei 25 °C. Mit steigender Temperatur sinkt der Wert des NTC, die Spannung an THM sinkt ebenfalls. Bei umgekehrter Temperaturdrift wandert die Spannung an THM in Richtung höherer Werte. Der MAX8677A lädt den Akku nur, solange die Spannung an THM den Bereich 0,28...0,74 V_L nicht verlässt. Bei gebräuchlichen Li-Ion-Akku-Typen entspricht dies ungefähr dem Temperaturbe-

reich 0...50 °C. Falls im Akku kein NTC-Widerstand eingebaut ist, muss R8 hinzugefügt werden, die Spannung an THM liegt dann konstant auf 0,5 V_L.

Tipps

In Li-Ion-Akkus aus Mobiltelefonen sind bereits Schutzschaltungen integriert, die den Akku vor Überlast und Tiefentladung bewahren. Für einzelne Zellen, beispielsweise aus dem Akkupack eines defekten Notebooks, gilt dies nicht. Die dort eingebaute elektronische Sicherung ist für die Zellen gemeinsam ausgelegt, eine einzelne Zelle bedarf eines eigenen Schutzes.

Eine simple Schmelzsicherung, FS1 in Bild 3, schützt wirksam und zuverlässig gegen Überlast einer einzelnen Zelle. Schmelzsicherungen sind jedoch kein Mittel gegen Tiefentladung, und diese kann Li-Ion-Zellen ebenfalls nachhaltig beschädigen oder zerstören. Die Gefahr besteht insbesondere bei Belastung mit einem ohmschen Verbraucher wie zum Beispiel einer Glühlampe. Dagegen schalten elektronische Verbraucher meistens selbsttätig ab, sobald die Betriebsspannung unter einen Mindestwert sinkt. Daraus folgt, dass die Eigenschaften des angeschlossenen Verbrauchers entscheiden, ob eine einzelne Zelle lediglich mit einer Schmelzsicherung geschützt werden kann.

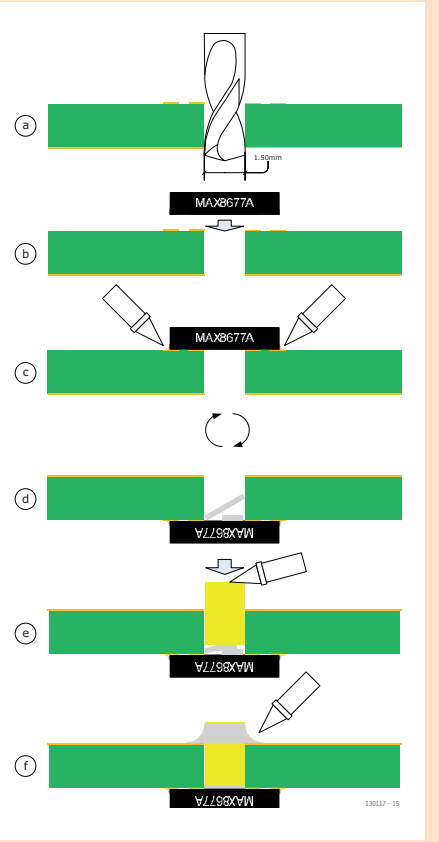
Aufbau

Der Autor hat für den Li-Ion-Akkulader eine Platine entworfen, die mit SMDs bestückt ist (**Bild 5**). Die Abmessungen sind so gering, dass der Akkulader bequem in ein vorhandenes Gerät eingebaut werden kann. Das Platinenlayout, entwickelt mit DesignSpark, haben wir zum freien Download auf die Projektseite [2] gestellt.

Für die Montage der SMDs sind Geschicklichkeit und Löterfahrung notwendig. Der MAX8677A befindet sich in einem Gehäuse der Bauform TQFN, das eigentlich den Einsatz eines Reflow-Ofens erfordert. Die Anschüsse und die wärmeableitende Fläche befinden sich auf der Unterseite des 4 · 4 mm messenden Winzlings. Der Autor hat jedoch eine alternative Lötmethode ersonnen, sie ist nebenstehend beschrieben. Für den Akku, das anzuschließende Gerät und die LEDs sind auf der Platine Lötbohrungen vorhanden, so dass diese Komponenten über Kabel angeschlossen werden können. Die USB-Mini-Buchse hat zwei Nocken, die in den Passbohrungen für die Justierung sorgen. Wenn auf die USB-Mini-Buchse verzichtet wird, können die Bohrungen als Anschlusspunkte für die Betriebsspannung dienen. Auch in diesem Fall muss die Spannung +5 V betragen, sie muss polrichtig angeschlossen werden.

(130117)gd

TQFN von Hand löten



Ein Reflow-Lötofen würde die Arbeit zwar erleichtern, doch dem löterfahrenen Elektroniker gelingt die Montage des MAX8677A auch mit einem Heißluft-Lötgerät. Sogar ein herkömmlicher Lötkolben führt zum Ziel, selbst dann, wenn die Platine in eigener Regie angefertigt wurde und nicht durchkontaktiert ist.
Bringen Sie auf der Fläche des Pads mittig eine Bohrung mit dem Durchmesser 1,5 mm an (a). Positionieren Sie das IC (b) und verlöten Sie die Kontakte an den Seitenkanten (c).
Mit Entlötlitze entfernen Sie die überschüssigen Reste des Lötmittels. Drehen Sie die Platine um und schieben Sie eine kleine Menge Lötmittel in die Bohrung (d). Nehmen Sie ein kurzes Stück Kupferdraht, Durchmesser passend zur Bohrung, und feilen Sie ein Drahtende sauber flach. Stecken Sie dieses Drahtende in die Bohrung und heizen Sie es mit dem Lötkolben auf (e). Nach kurzer Zeit hat das Drahtstück die Temperatur erreicht, bei der das Lötmittel in der Bohrung schmilzt.
Das Drahtstück sinkt nach unten und stellt den Lötkontakt mit dem Pad auf der Unterseite des IC her. Verlöten Sie das andere Drahtende mit der Massefläche auf der Lötseite der Platine (f). Jetzt haben Sie eine zuverlässige elektrische und thermische Verbindung zwischen dem Pad und der Massefläche hergestellt.

Weblinks

[1] http://datasheets.maximintegrated.com/en/ds/MAX8677A.pdf

[2] www.elektor.de/130117

Tag 1:
Machen Sie, was Sie wollen!

Von **Neil Gruending**

Tag 1: RS Components hat soeben die neue Version 5.0 von DesignSpark PCB vorgestellt. Das erste, was mir in den Sinn kam, war, das neue Tool so zu konfigurieren, wie ich es wollte. Denn bei DesignSpark lässt sich alles konfigurieren, per Datei oder global.

Erste Schritte

Bevor wir anfangen, mit den Konfigurationen von DesignSpark zu spielen, ist es wichtig zu erfahren, dass DesignSpark Styles (Stile) verwendet, die Formatierungsregeln für Stammfunktionen wie Shapes, Text und Tracks angeben. Jeder Stil erhält einen Namen, um ihn einfach wiederzuerkennen, ähnlich wie in einem Textverarbeitungsprogramm. Normalerweise versuche ich, allen Stilen sinnvolle Namen zu verleihen, dann weiß man sofort über die Einzelheiten Bescheid, ohne die Eigenschaften durchforsten zu müssen. Beispielsweise ist ein Stil mit dem Namen „Via“ in Ordnung, wenn man nur eine Via definieren will, aber wenn man den Stil „Via (0.45mmx0.95mm)“ nennt, ist sofort klar, dass es sich um ein 0,95-mm-Kupferpad mit einer 0,45 mm-Bohrung handelt. Sie können zwar benutzerdefinierte Stile hinzufügen, aber hier wollen wir uns auf die Besprechung der vorgegebenen Stile wie „[Symbol Names]“ beschränken, den Stilen für Bauteilsymbole und Bauteilnamen. Vergessen Sie nicht, den Pfad auf das DesignSpark „Technology-File“ zu überprüfen, der auf der allgemeinen Registerkarte im Settings->Preferences-Menü geändert werden kann. In meiner Installation musste ich den Pfad in C:\Users\Public\Documents\DesignSpark PCB 5.0\Technology ändern. Wenn der Pfad falsch ist, kann DesignSpark die Technology-Dateien nicht automatisch finden; es ist dann schwieriger zu bedienen. Der Verzeichnispfad ist korrekt eingestellt, wenn er .ptf- und .stf-Dateien enthält.

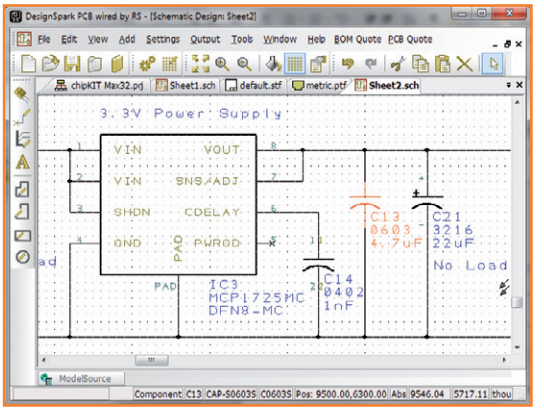


Bild 1.
Schaltplan mit voreingestellten Parametern.

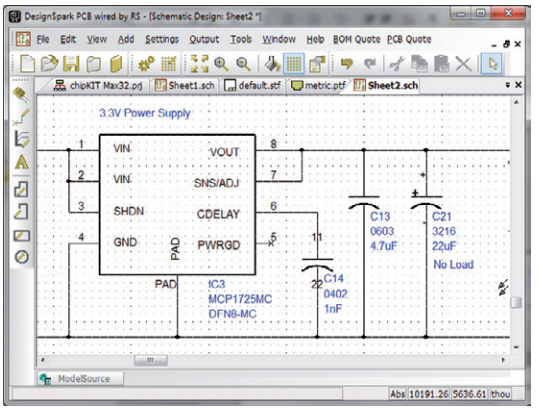


Bild 2.
Die Standard-Schriftart ist auf Arial geändert.

Schematic Technology Files

In den Schematic Technology-Dateien können Sie konfigurieren:

- die Linienarten für Terminal- und Junction-Verbindungen.
- vordefinierte Textformate, die im Schaltplan verwendet werden (Schriftart, Größe, ...)
- wie diverse Linien-Elemente gezeichnet werden (solid, gestrichelt, ...)
- wie die Verbindungs-Leitungselemente gezeichnet werden (solid, Breite, ...)
- alle vordefinierten elektrischen Netze (obwohl ich dies lieber in den Schematics tun würde)

- alle vordefinierten elektrischen Netz-Klassen (Masse, Plus, ...)
- die Farben für verschiedene Elemente

Diese Parameter lassen sich im Settings->Design Technology- und im View->Colors-Menü ändern. Sie können am Beispielprojekt *chipKIT Max32* sehen, wie die Standardeinstellungen (**Bild 1**) aussehen. Ich persönlich finde die Schriftart recht altbacken, ich bevorzuge TrueType-Schriftarten wie Arial. Etwas an den Einstellungen herumspielen und nach ein paar Minuten sieht das Projekt aus wie in **Bild 2**.

Wie setzt man das in der Schematic-Technology-Datei um? Zuerst öffnet man die vorgegebene, noch leere Schematic-Technology-Datei default.stf, die sich normalerweise in C:\Users\Public\Documents\DesignSpark PCB 5.0\Technology befindet. Öffnen Sie nun das Settings->Design Technology-Menü und ändern Netz- und Pinnamen, Pin-Nummern und schließlich die Symbol-Namen-Textstile, um die Schriftart Arial mit einer Höhe von 80 verwenden. Ich habe auch den normalen Text-Stil auf Arial eingestellt, aber mit einer Größe von 120. Dann ging ich in das View->Colors-Menü und änderte die Pin-Namen und Pin-Nummern-Felder auf schwarze Schriftfarbe.

Nachdem Sie alle gewünschten Änderungen vorgenommen haben, speichern Sie die Technology-Datei. Sie können Sie dann für neue Schaltpläne verwenden, indem Sie „default.stf“ im New-Document-Menü-Fenster auswählen.

PCB Technology Files

PCB Technology-Dateien sind eine der besten Eigenschaften von DesignSpark, da sie es ermöglichen, alle Ihre grundlegenden Design-Regeln in Dateien zu speichern, die Sie leicht wieder verwenden können. Ich habe zum Beispiel eine Datei für eine einfache zweilagige Low-Cost-Platine und eine weitere für eine vierlagige Platine angelegt. Ich kann dann wählen, welche Datei ich beim Erstellen einer neuen Platine benutzen möchte. Diese Funktion ist bei anderen Platinenlayout-Paketen nicht zu finden.

PCB Technology-Dateien erlauben die Konfiguration:

- der Einheiten (mm, mil, ...) und der Auflösung
- der Gitter, vor allem des Gitters im Arbeitsbereich

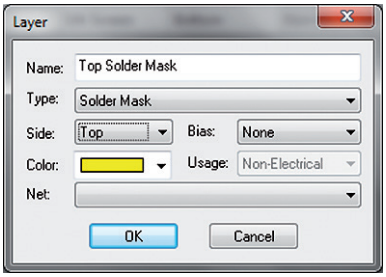


Bild 3.
Konfigurieren der Layer–Reihenfolge.

- des Layer-Aufbaus und der Farben
- der Regeln für Spacing und Clearance
- der Styles für Pads und Tracks (obwohl Sie in der Regel die Pad-Styles in den Bibliothek-Komponenten spezifizieren und nur die Standardeinstellungen für Track-Styles in der Technology-Datei angeben)
- der Standard-Netzklassen (ich gebe sie lieber im Schaltplan an)
- der Autorouter- und Autoplacer-Regeln
- der grundsätzlichen Design-Elemente wie Platinenform, Bohrungen und so weiter

Schauen wir mal, wie das am Beispiel einer einfachen zweilagigen Platine funktioniert. Der erste Schritt ist, eine schon existierende Technology-Datei zu kopieren, damit wir nicht bei Null anfangen müssen. Wir benutzen hier C:\Users\Public\Documents\DesignSpark PCB 5.0\Technology\metric.ptf und speichern die Datei unter dem neuen Namen my2layer.ptf. Dann ändern wir zuerst die Design-Einheiten im Settings->Units-Menü. Ich arbeite immer mit Millimetern und einer Genauigkeit von vier Dezimalstellen, aber wenn Sie es vorziehen, können Sie auch imperiale Einheiten (Kubikfüße und so) wählen oder mil. Sie können dann Ihr bevorzugtes Gitter im Settings->Grids-Menü aussuchen.

Die Konfiguration der Layer wird auf der Registerkarte Layer in Settings->Design Technology vorgenommen. Standardmäßig sind in der metrischen Technology-Datei die Layer Top Silkscreen, Top Copper, Documentation, Bottom Copper und Bottom Silkscreen bereits definiert. Da auf allen meinen Platinen SMDs verwendet werden, habe ich die Layer Top Paste, Top Solder Mask, Bottom Solder Mask und Bottom Paste hinzugefügt. Sie können Ebenen hinzufügen, indem Sie auf die

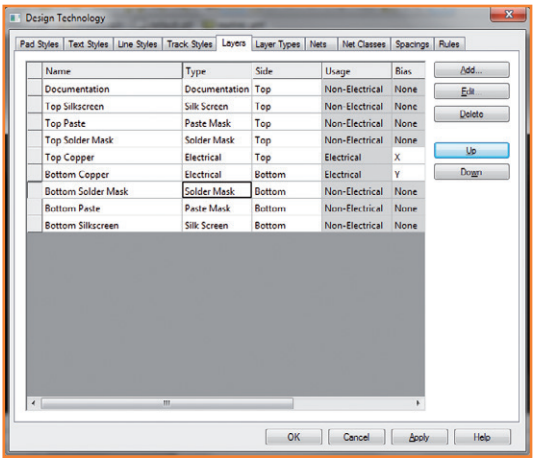


Bild 4.
Regeln für eine einfache zweilagige Platine.

Schaltfläche Add klicken und die Layer-Parameter wie in **Bild 3** hinzufügen.
Sobald Sie alle Layer hinzugefügt haben, können Sie sie richtig anordnen, indem Sie die Layer im Layer-Fenster nach oben oder nach unten ziehen. Wenn Sie fertig sind, sollte das so aussehen wie in **Bild 4**. Als nächstes kommen die Spacing-Regeln in der Registerkarte Spacing an die Reihe. Sie sehen eine Matrix mit Spacing-Regeln für die verschiedenen Objekttypen. Für eine einfache Zwei-Lagen-Platine mit 10-mil-Tracks und 10-mil-Spacing sehen die Regeln aus wie in **Bild 5**.

Vergessen Sie nicht, auch auf die Registerkarte Rules zu klicken. Die wichtigsten Parameter sind „minimum annular ring“ und „component spacing“. Dann stellt man die Trackbreiten in der Registerkarte Track Styles ein. Für eine 10 mil/10 mil-Platine würde ich die minimalen und normalen Signal-Tracks auf 0,25 mm einstellen. Die Stromversorgungs-Leiterbahnen können beliebig breit sein, aber die gleiche Breite ist empfehlenswert, dann lassen sich diese Leiterbahnen besser an die Bauteilpads anschließen. Der letzte Schritt ist es, die Via-Styles in der Registerkarte Pad Styles zu bearbeiten. Normal für eine solche Platine ist eine Drill-Einstellung von 0,45 mm bei einem Pad von 0,95 mm. Sie können natürlich noch andere Styles definieren, wenn Sie mehrere Via-Größen nutzen möchten. Jetzt haben Sie eine Reihe von grundlegenden Design-Regeln und Einschränkungen für eine einfache zweilagige Platine definiert, die für zukünftige Entwicklungen benutzt werden kann. Wenn Sie ein neues Platinendesign erstellen, wählen Sie die entsprechende Technology-Datei, wenn Sie der PCB Creation Wizard dazu aufgefordert.

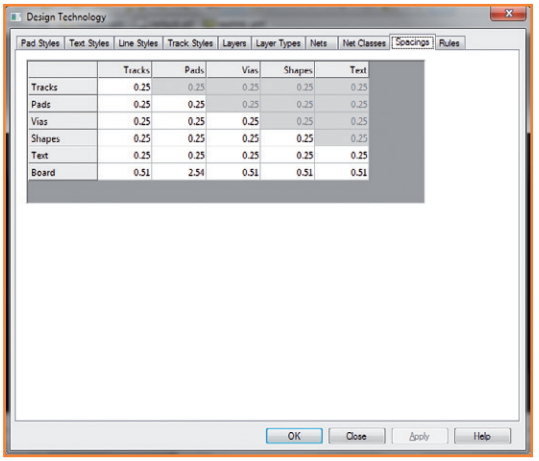


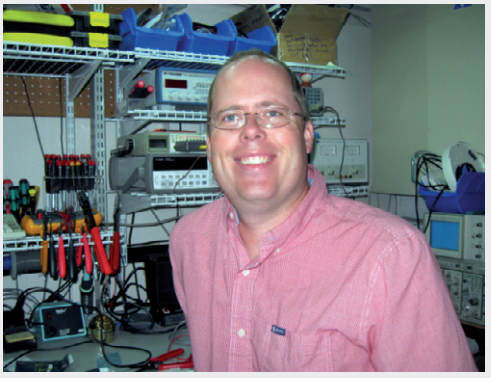
Bild 5.
Design-Regeln.

Fazit

Wir haben jetzt die Standard-Parameter von DesignSpark konfiguriert. Die nächsten Schritte sind die Konfiguration der DesignSpark-Bibliotheken und die Erstellung einiger Dokumentationsvorlagen. Glücklicherweise bietet DesignSpark eine Unzahl von Bibliotheken, die den Einstieg erheblich erleichtern.

(130172)

Hallo, ich bin Neil Gruending und habe als Elektronik-Ingenieur im Laufe der Jahre viele verschiedene Platinen-CAD-Pakete ausprobiert. Ich bin mit meinen Werkzeugen ziemlich gut vertraut und versuche herauszufinden, wie sie meine Produktivität maximieren können. Ich gebe auch gerne mein Wissen weiter auf meiner Website www.gruending.net und auf Twitter als @ngruending.



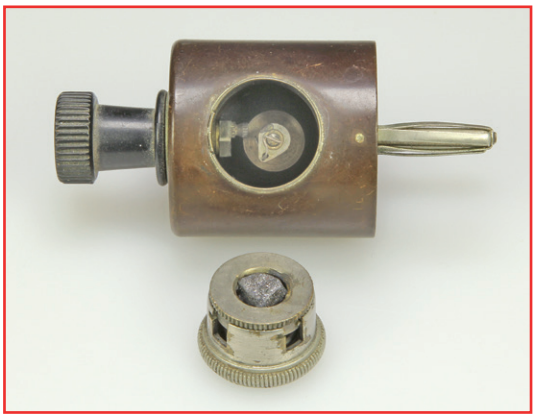
Von den Machern von Elektor!

Oder frei Haus unter www.elektor.de/roehren9 bestellen!

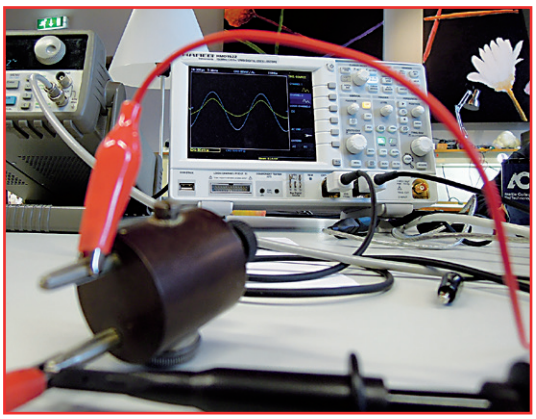
Beim Schnurrhaar der Katze!

Von **Wisse Hettinga**
(Elektor)

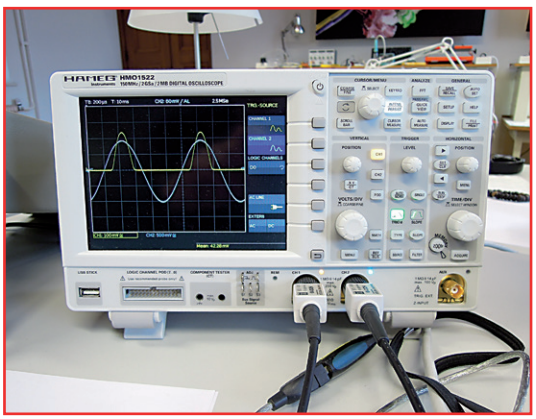
Normalerweise, wenn ich beginne, einen Artikel zu schreiben, lasse ich mich vom Durcheinander auf meinem Schreibtisch inspirieren. Für diese erste Folge der „Verrückten Welt der Bauteile“ fiel mein Blick auf eine Spitzendiode vom Typ FRIHO DRP. Sie wurde schon in einem Radio-Bauer-Katalog aus dem Jahr 1926 erwähnt - wir reden hier über Antiquitäten - und es ist das wichtigste Bauteil eines Detektorempfängers. Um ehrlich zu sein, ich habe keine Ahnung, wie und wann ich dieses Bauteil erstanden habe, wahrscheinlich beim Wühlen im legendären Elektronik-Geschäft von Kwakkelstein in Vlaardingen (Niederlande). Und seitdem hat es wahrscheinlich auf meinem Schreibtisch oder in meiner Schublade vor sich hinvegetiert. Dieses Bauteil führt uns zurück zu den Anfängen des Radios. Die ersten Detektor-Empfänger waren mit einem Galenit-Kristall (Bleiglanz) versehen, in den man einen spitzen Draht steckte und damit den pn-Übergang zur Demodulation des Radiosignals schuf. Dieser Übergang war nicht stabil und musste bei Bedarf manuell „nachgebessert“ werden, mit anderen Worten, man musste mit dem Draht im Kristall herumporkeln, bis der Empfang wiederhergestellt war. Aufgrund seiner Form wurde diese Anordnung *Cat’s Whisker Diode* (Schurrhaar-Diode) genannt (**Bild 1**). Die FRIHO DRP ist sozusagen eine automatisierte Schurrhaar-Diode.



1



2



3

Aber wie gut ist diese Diode im Vergleich zu heutigen? Es sollte wirklich einfach sein, die Eigenschaften einer Diode zu messen. Man benötigt ein Netzgerät, ein Voltmeter und ein Amperemeter. Ein wenig schwierig war es, diese Diode zum Funktionieren zu bringen und die richtige Stelle zu finden, an welcher der Diodeneffekt einsetzt. Ich versuchte vergeblich, diese Stelle mit einem Ohmmeter zu finden - die Messergebnisse waren völlig daneben – und ich wollte schon aufgeben, als ich noch einen 330-Ω-Widerstand, einen HP-Generator und ein Hameg-Oszilloskop HMO1522 ins Spiel brachte. Die Ergebnisse waren vielversprechend: In **Bild 2** ist im Oszilloskop ein sehr unscharfes Signal zu sehen, aber mit etwas Phantasie kann man erkennen, wie die FRIHO-Diode das sinusförmige Eingangssignal abschneidet. **Bild 3** zeigt im Vergleich die Kennlinie einer modernen Diode (bei identischem Messaufbau). Es ist schon erstaunlich, dass ein Bauteil wie die FRIHO DRP tatsächlich funktioniert! Heute gibt es Dioden für alle Arten von Anwendungen: Kleinsignal-Dioden, Z-Dioden, Kapazitätsdioden oder Varaktordioden, Tunneldioden - und alle diese mit einer Vielzahl von Gehäusebauformen. Und es ist schon sehr interessant zu erkennen, dass sie alle von einer solchen Cat’s Whisker Diode abstammen.

(130169)

ECD 7

Bauteilbibliothek mit über 75.000 Komponenten

Diese CD-ROM bietet Ihnen acht Datenbanken für ICs, Germanium- und Silizium-Transistoren, FETs, Thyristoren, Triacs, Dioden und Optokoppler. Weitere elf Anwendungen zur Berechnung von Vorwiderständen bei LEDs, Spannungsteiler, Ohmsches Gesetz sowie Farbcodeschlüssel für Widerstände und Induktivitäten etc. runden das Programmpaket ab.

Jede Datenbank zeigt für (fast) jedes Bauelement eine Gehäuseskizze, die Anschlussbelegung, die technischen Daten (soweit bekannt) und verfügt über eine Suchroutine nach Bauteil-Parameter.

Alle genannten Datenbank-Anwendungen sind interaktiv, d. h. Sie können Bauteile hinzufügen, ändern oder ergänzen.

ISBN 978-90-5381-298-3 · € 29,50 · CHF 36,60

elektor

Weitere Infos & Bestellung unter
www.elektor.de/ecd7

ARM-Mikrocontroller 2

30 Projekte in C für Fortgeschrittene

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 programmieren. Das Einzige, was Sie brauchen, ist ein Webbrowser mit Internetzugang und einen freien USB-Anschluss an Ihrem PC. Sie können von jedem beliebigen Ort der Welt auf Ihr Projekt zugreifen und daran weiterarbeiten. Wenn Sie fertig sind, genügen ein paar einfache Mausklicks, um Ihr Programm auf das mbed-System zu übertragen. Natürlich können Sie die Projekte auch auf Ihren eigenen PC laden und dort speichern.

Die Quelltexte zu den Beispielprogrammen stehen gratis unter www.elektor.de/arm-buch2 zum Download bereit. Das zum Buch gehörige Hardware-Starterkit kann unter www.elektor.de/arm-kit2 geordert werden.

243 Seiten (kart.) • Format 17 x 23,5 cm
ISBN 978-3-89576-271-0
€ 39,80 • CHF 49,40

elektor

Weitere Infos & Bestellung unter
www.elektor.de/arm-buch2

LCR-Meter im Vergleich

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

„Wäre es nicht aufschlussreich, unser 500-ppm-LCR-Meter mit einem Testexemplar von Hameg zu vergleichen?“

„Stimmt, ich fände das interessant!“

„Ich könnte auch noch ein antikes Messgerät aus meinem Retronik-Fundus mitbringen.“

„Das wäre toll, dann können wir schauen, wie sich drei unterschiedliche Messgeräte schlagen!“

Dies ist ein Ausschnitt aus einem Kaffee-Plausch mit einem Kollegen bei Elektor. Das Ergebnis war, dass ich mir drei LCR-Meter mit komplett unterschiedlichem Hintergrund für einen kurzen Test vorknöpfte: Das Modell HM8118 [1] von Hameg war eine Leihgabe von Rohde & Schwarz Netherlands [2] und ist ein professionelles Messgerät in der Preisklasse von gut 2.000 Euro. Beim Oldtimer aus der Retronik-Sammlung handelt es um eine Impedanz-Messbrücke des Typs 1650-A der Firma GRC (**G**eneral **R**adio **C**ompany) [3], die damals für happige 1.000 $ zu haben war. Zum Schluss noch das LCR-Meter mit den 500 ppm von Elektor [4], dessen Teilewert unter 400 Euro liegen dürfte.

Alter geht vor: Den Anfang machte das GRC-Gerät. Da es älter ist als ich und zumindest für mich weniger intuitiv zu bedienen, musste ich zunächst einmal im Handbuch schmökern. Es brauchte gut Zeit, bis ich damit klarkam und vernünftige Messergebnisse erzielen konnte. Zusammengefasst:

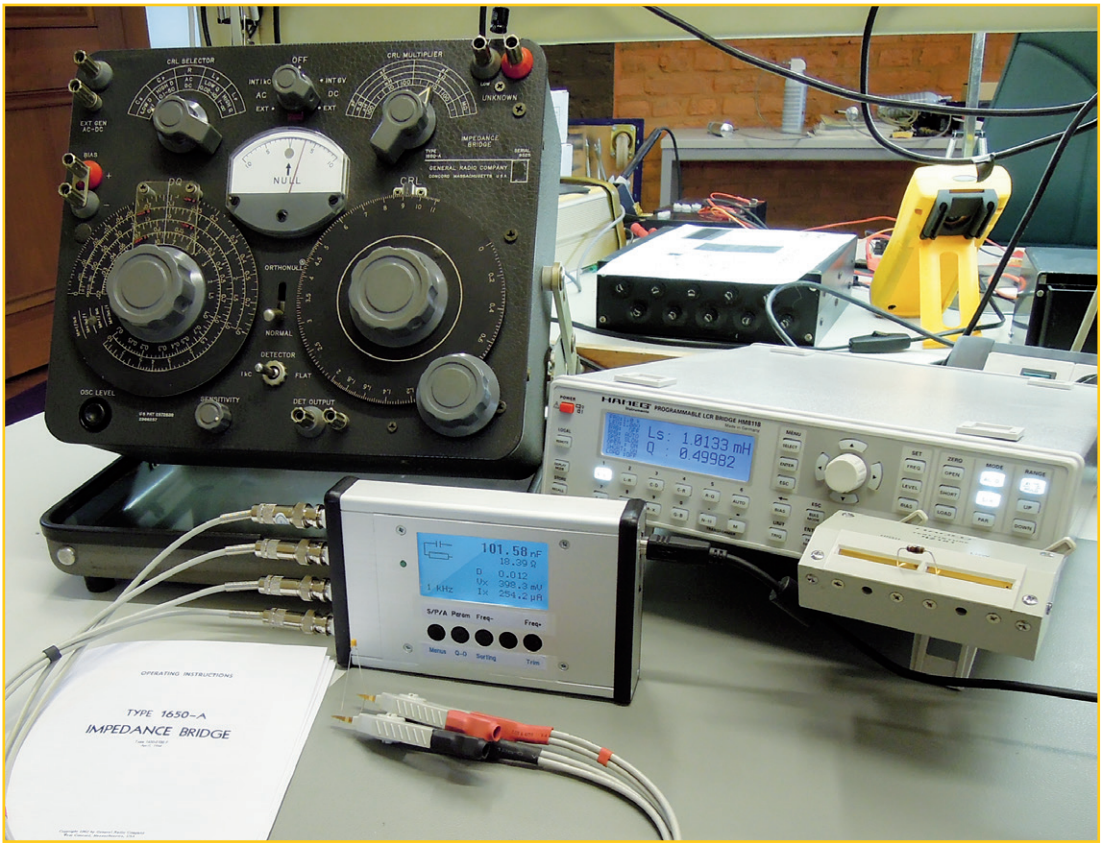
- Handbuch lesen: 10 Minuten.
- Herausfinden, wie man Kapazitäten misst: 5 Minuten.
- Eine Messung durchführen (Knöpfe auf null Auslenkung einstellen): 3 Minuten.

Ehrlicherweise dürften bei täglicher Nutzung des 1650-A gut 80 % der reinen Messzeit wegfallen – bleibt aber immer noch mehr als eine halbe Minute pro Messung. Ich muss aber zugeben, dass das Aussehen und die Knöpfe und Schalter dem Instrument Charakter verleihen.

Dann kam das HM8118. Als modernes Gerät gibt es hier mehr oder weniger „plug-and-play“. Nach der Autokalibrierung kommt das DUT (**D**evice

Vergleichstabelle

DUT	GRC 1650-A	Hameg HM8118	Elektor LCR Meter
Widerstand 8,2 Ω	8,22 Ω	8,2458 Ω (V_x=31,86 mV, I_x=3,862 mA)	8,2379 Ω (V_m=31,13 mV, I_m=3,779 mA)
Kondensator 100 nF	100 nF *D*: 0,0267	102,03 nF *D*: 0,01205 R_s=19,20 Ω (V_x=364,9 mV, I_x=233,9 µA, 1 kHz)	102,20 nF *D*: 0,013 R_s=19,20 Ω (V_m=398,1 mV, I_x=256,0 µA, 1 kHz)
Kondensator 100 µF	92 µF *D*: 0,1	94,475 µF *D*: 0,10099 R_s=170,27 mΩ (V_x=7,301 mV, I_x=4,312 mA, 1 kHz)	92,400 µF *D*: 0,090 R_s=155,4 mΩ (V_m=7,076 mV, I_x=4,091 mA, 1 kHz)
Induktivität 1 mH	1,255 mH *Q*: 0,63	995,85 µH *Q*: 0,49189 R_s=12,722 Ω (V_x=52,60 mV, I_x=3,711 mA, 1 kHz)	993,2 µH *Q*: 0,492 R_s=12,681 Ω (V_m=51,18 mV, I_x=3,621 mA, 1 kHz)



Under **T**est) in die Testhalterung (Hameg HZ181) und schon erscheinen die gewünschten Eigenschaften des Bauteils auf dem blauen LCD. Zeitbedarf 40 Sekunden vom Beginn bis zum Ergebnis, ganz ohne Blick in das Handbuch. Mehrere Bauteile messen? Für das Auswechseln des DUT sind 3 Sekunden fällig und die Wahl des richtigen Messbereichs kostet vielleicht noch 2 Sekunden, sodass 5 Sekunden pro Bauteil zusammen kommen.

Last but not least: Das LCR-Meter von Elektor ist mit einem LCD, einem Einschalter und fünf Tasten bestückt. „Plug-and-play“, wohin man schaut. Die Kalibration muss nur einmal erfolgen und war beim verwendeten Prototypen schon erledigt. Ich konnte also sofort „losmessen“ und ein DUT an die Clips hängen. Geschätzte Zeit: 10 Sekunden. Sequentielle Messungen brauchen mit den Clips wohl etwas mehr Zeit als beim Hameg-LCR-Meter, aber das HZ181 funktioniert auch am Elektor-Gerät ;-)

Mein Fazit ist, dass sich seit 1960 eine Menge getan hat, wenn man die Messergebnisse der Geräte von Hameg und Elektor betrachtet. Unser kleines Elektor-Messgerät hält sich ganz gut im Vergleich mit dem Boliden von Hameg. Allerdings wäre es unfair, die vielen Messmöglichkeiten und die hohe Grundgenauigkeit von 0,05 % des Hameg-Modells zu verschweigen.

Überraschenderweise schlägt sich das 50 Jahre alte GRC 1650-A selbst dann gar nicht so schlecht, wenn es von jemand wie mir bedient wird, der nicht so viel Erfahrung mit Elektronik dieser Epoche mitbringt. Man kann es auch heute noch im Labor gebrauchen. Solche Messgeräte werden heute nicht mehr gebaut, oder? Wenn Sie Ihre Erfahrungen mit antiken Messgeräten teilen möchten, dann schicken Sie uns doch bitte eine E-Mail an editor@elektor.com.

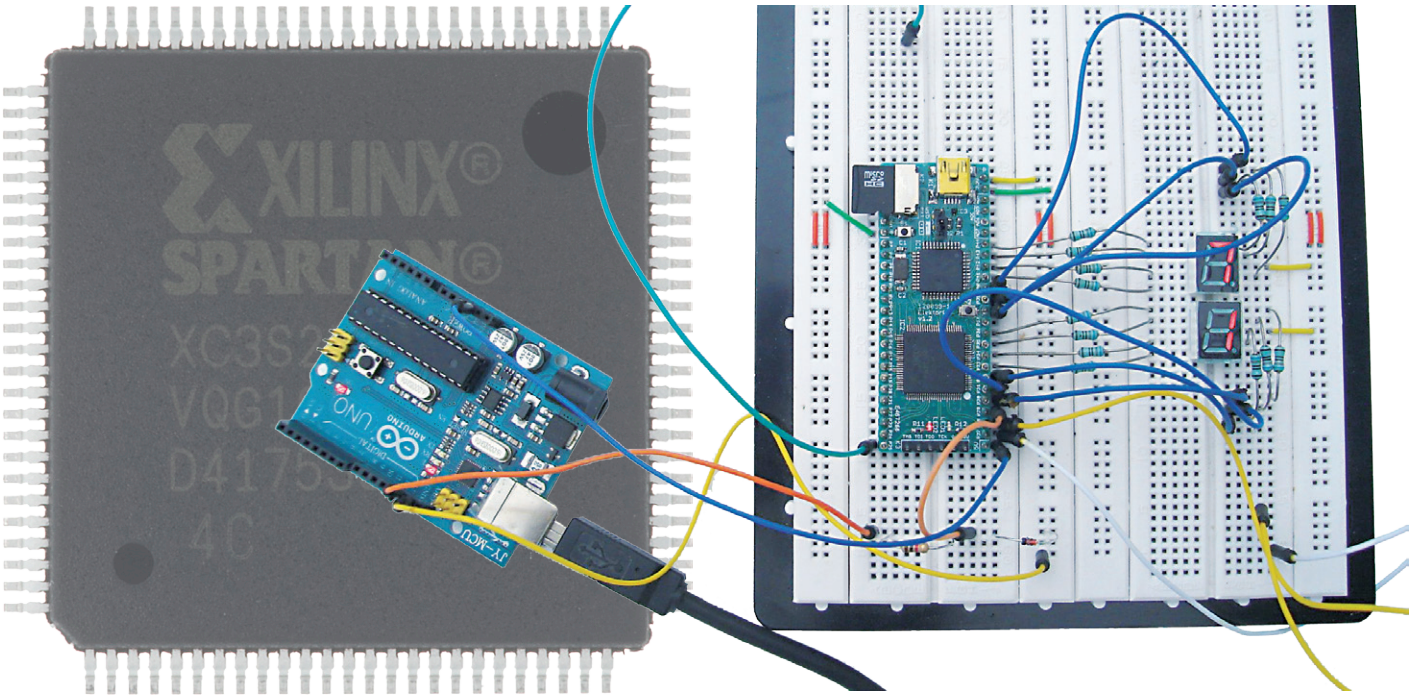
(130166)

Weblinks

[1] www.hameg.com/13.0.html
[2] www.testenmeetwinkel.nl
[3] www.elektor.de/075064
[4] www.elektor.de/110758

Bauen Sie Ihren Chip! (5)

250.000 Gatter nach Programm verknüpft



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

FPGA-Anwendungen lassen sich als Kombinationen logischer Symbole oder alternativ in einer Programmiersprache realisieren. Zu den Stärken der zweiten Methode gehört, dass komplexe Funktionen oft schneller durch Algorithmen als durch Logikschemen beschreibbar sind. Dieser Teil unseres FPGA-Kurses zeigt am Beispiel eines DCF77-Decoders, wie eine FPGA-Anwendung als Programm erstellt werden kann.

Schon in der letzten Folge [4] hatten wir uns mit den Hardware-Beschreibungssprachen VHDL und Verilog beschäftigt, um FPGA-Anwendungen zu testen. Diesmal geht es darum, eine FPGA-Anwendung in einer Programmiersprache zu erstellen. Zwei Sprachen, die global betrachtet dem gleichen Zweck dienen, sind eigentlich für den Anfang des Guten zu viel. Nach reiflicher Überlegung haben wir uns entschieden, die Sprache VHDL als Grundlage unserer weiteren Betrachtungen zu nutzen. VHDL hat die Eigenschaft, dass die für Simulationen nötigen Zeiten vergleichsweise kurz sind, auch weil der Weg zu synthetisierbaren Entwürfen etwas schwieriger ist. Das spricht eigentlich für Verilog, doch da das Simulieren meistens mehr Zeit kostet als das Synthetisieren, lässt sich das Ziel mit VHDL häufig etwas schneller erreichen. In Analogie zur Programmierung von Computer-Software sei angemerkt (Verilog-Freunde, bitte um Nachsicht!): Anstelle einer „blinden“ Programmierung und anschließender mühsamer Fehlerbereinigung im Debugger ist es besser, überlegt zu programmieren und den Debugger wirklich nur zum Debuggen eines schon lauffähigen Algorithmus zu benutzen. Die FPGA-Anwendung, an der wir VHDL erproben wollen, ist ein Decoder für das Zeitzeichen des Senders DCF77. Der im unteren Langwellenbereich angesiedelte Sender, Standort Mainflingen

bei Frankfurt am Main, ist im zentralen Europa fast überall mit wenig Aufwand empfangbar. Der Aufbau des Zeitzeichensignals ist nicht kompliziert: Die Information steckt in einer Impulsfolge, bestehend aus 59 Impulsen im Sekundenabstand. Die Impulse können 100 ms (= logisch 0) oder 200 ms (= logisch 1) lang sein. Das Ende der Impulsreihe ist dadurch gekennzeichnet, dass der 60. Impuls (Bit 59) fehlt. Der nächste Impuls gibt den Anfang der nächsten Minute an. Die Zeit und das Datum sowie einige weitere Informationen sind im BCD-Format codiert. Mit mehreren gesendeten Paritätsbits lässt sich auf der Empfangsseite feststellen, ob gültige Daten empfangen wurden oder Störungen aufgetreten sind. Ein so beschaffenes Signal ist gut geeignet, um mit VHDL (oder auch Verilog) einen Decoder und eine Anzeige auf einem Siebensegment-Display zu realisieren. Für die Leser, die außerhalb der DCF-Reichweite leben, haben wir ein Programm geschrieben, das den Zeitzeichensender DCF77 auf einem Arduino-System simuliert [5].

Wir wollen mit einem funktionellen Entwurf des DCF77-Decoders beginnen. Dabei setzen wir voraus, dass das empfangene und zu decodierende Signal die vom Betreiber des Zeitzeichensenders propagierten Eigenschaften mitbringt und nicht gestört ist.

Wie schon erwähnt, besteht das Signal aus Impulsen, die entweder 100 ms oder 200 ms lang sind. Jeder kurze Impuls steht für logisch 0, jeder lange Impuls ist gleichbedeutend mit logisch 1. Wenn wir die Impulse 150 ms nach ihrer ansteigenden Flanke abtasten, können wir, wie in **Bild 1** dargestellt, die zugehörigen logischen Signale bestimmen. Das Kriterium für den Beginn einer Impulsreihe ist der fehlende 59. Impuls. Um diese Lücke zu finden, genügt das Messen der Zeiten zwischen zwei aufeinander folgenden, ansteigenden Impulsflanken. Die aus den Impulslängen resultierenden logischen Zustände schieben wir in ein Schieberegister. Am Ende jeder Impulsreihe filtern wir die Zeit- und Datum-Informationen aus und stellen sie auf einem Siebensegment-Display dar. In einem auf der Programmiersprache C basierenden Pseudocode könnte dies wie folgt aussehen:

```
führe aus bei jedem taktimpuls
{
  zähler = zähler + 1;
  wenn (zähler == 150_ms)
  {
    schieberegister = (schieberegister << 1)
+ eingangssignal;
  }
wenn
(ansteigende_flanke(eingangssignal) == wahr)
  {
    wenn (zähler >= 1750_ms)
    {
      zeige_an_inhalt(schieberegister);
    }
    zähler = 0;
  }
}
```

Der Zähler wird mit jedem Taktimpuls inkrementiert, also um 1 erhöht. Wenn der Zählerinhalt 150 ms erreicht, wird das Eingangssignal abgetastet, das Ergebnis wird ins Schieberegister geschoben. Bei jeder ansteigenden Flanke des Eingangssignals prüft der Algorithmus, ob der maximale Zählerstand erreicht wurde. In diesem Fall steht die Information vollständig im Schieberegister, Zeit und Datum werden ausgegeben. Zum Schluss wird der Zähler auf Null zurückgesetzt. Dieses Grundgerüst eines DCF77-Empfangsalgorithmus muss normalerweise durch Algorithmen zur Fehlererkennung und Empfangssicherheit ergänzt werden. Dies soll jedoch bei unserem nur der Übung dienenden Projekt außer Betracht bleiben.

Und nun in VHDL

In VHDL ist der Algorithmus für den Empfang des DCF77-Zeitzeichens nicht schwierig, der Pseudo-

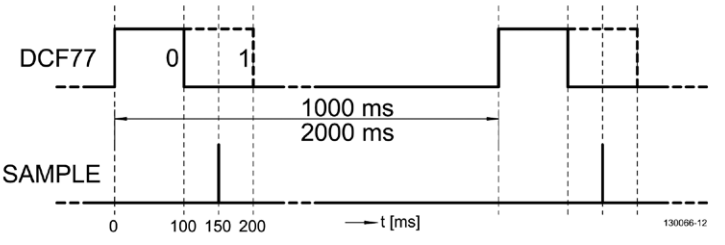


Bild 1.
Das Signal des Zeitzeichensenders DCF77 wird abgetastet.

code ist in den VHDL-Code fast direkt umsetzbar:

```
 1  process (clock) is
 2  begin
 3    if rising_edge(clock) then
 4      counter <= counter + 1;
 5      if (counter=t150ms) then
 6       bits <= input & bits(58 downto 1);
 7      end if;
 8      if input_rise='1' then
 9        if (counter>=t1750ms) then
10          data <= bits;
11        end if;
12        counter <= 0;
13      end if;
14    end if;
15  end process;
```

Hier fehlen nur noch die Definitionen und einige andere Elemente der VHDL-Syntax. Die Zeilennummern haben wir hinzugefügt, um die nachfolgenden Erklärungen zu erleichtern. Der Zähler heißt hier `counter`, und das 59 bit breite Schieberegister haben wir `bits` genannt.
In Zeile 1 steht, dass dies ein *Prozess* ist. Es handelt sich um Code, der vom FPGA ausgeführt werden soll, ohne Prozess findet keine Aktivität statt. Erlaubt sind auch mehrere Prozesse, sie werden gleichzeitig ausgeführt. Prozesse werden prinzipiell von oben nach unten abgearbeitet. Dieser Prozess hängt vom Signal `clock` ab. Deshalb wird der Prozess nur ausgeführt, wenn sich der Wert von `clock` ändert. Der Prozess beginnt mit Zeile 2, er endet mit Zeile 15.
Zeile 3 bewirkt, dass alle Prozessschritte synchron zu den ansteigenden Flanken des Taktsignals ablaufen, verantwortlich ist die Funktion `rising_edge`. Häufig enthält VHDL-Code die Konstruktion

```
if clock'event and clock='1' then
 ..
end if;
```

Die Funktion stimmt zwar mit `rising_edge` überein, doch diese Konstruktion gilt als veraltet. Parallel zu `rising_edge` gibt es in VHDL auch die Funktion `falling_edge`.
In Zeile 4 wird der Zähler inkrementiert, der Zählerstand wird um 1 erhöht. Hier ist wichtig, dass die Addition, mathematisches Zeichen „+", das Einbinden der Bibliothek `numeric_std` verlangt. Wie dies geschieht, werden wir später erklären. Zeile 5 vergleicht den Zählerstand mit einem konstanten Wert, der 150 ms bei der FPGA-Taktfrequenz 8 MHz entspricht. Wenn der Zähler 150 ms erreicht hat, wird der logische Zustand des Eingangssignals in das Schieberegister `bits` geschoben (Zeile 6). Anders als beim Pseudocode verläuft die Schieberichtung hier von links nach rechts. Das hat seinen Grund darin, dass der DCF77-Code mit dem niederwertigsten Bit (LSB) beginnt. Nach 59 eingetakteten Zuständen des Eingangssignals befindet sich Bit 0 an Position 0. Die Realisierung der Schiebeoperation ist ungewohnt, denn hier wird die Verknüpfungsfunktion „&" benutzt. Diese Funktion fügt den links stehenden Ausdruck zum rechts stehenden Ausdruck hinzu. Auf der rechten Seite steht die 58 bit lange Kette mit den Bitnummern 58...1 (ohne Bit 0), diese Bits werden in die Positionen 57...0 verschoben. Der linke Teil ist der 1 bit breite logische Zustand des Eingangssignals, er wird in Position 58 gespeichert. Wie ISE (oder eigentlich XST) diese Konstruktion behandelt, wollen wir an dieser Stelle nicht betrachten.
Zeile 8 prüft, ob das Eingangssignal seinen Zustand mit einer ansteigenden Flanke ändert. Wenn das zutrifft, prüft Zeile 9, ob der Zähler den Wert 1750 ms überschritten hat. Trifft auch dies zu, wird der Inhalt des Schieberegisters in ein Datenregister kopiert. Die Auswertung dieser Daten übernimmt ein anderer Programmteil, wir kommen darauf zurück. In Zeile 12 wird der Zähler in den Anfangszustand zurückgesetzt. Das Schieberegister muss nicht rückgesetzt werden, denn seine Länge entspricht einer Minute, dies ist die Zykluszeit. Die hier nicht erwähnten Programmzeilen dienen lediglich dazu, der VHDL-Syntax zu genügen.
Damit dieser Programmteil lauffähig ist, sind eine Flankenerkennung, einige Ergänzungen für die VHDL-Syntax sowie Definitionen zu den Eingangs- und Ausgangssignalen des Prozesses nötig. Unser Programm generiert nebenbei Signale, die auf dem FPGA-Board die LEDs blinken lassen. Die blinkenden LEDs signalisieren, dass der FPGA in Aktion ist.

Flanken erkennen

Für den korrekten Datenempfang hat die Flankenerkennung eine zentrale Bedeutung. Zuerst hatten wir uns eine simple Konstruktion überlegt, die den aktuellen Zustand des Eingangssignals mit dem Wert zum Zeitpunkt des vorangegangenen Taktsignals vergleicht. Diese Lösung arbeitete unzuverlässig, häufig setzte die Synchronisation aus. Abhilfe brachte das Hinzufügen eines Flipflops, so dass nicht der aktuelle und der voran-

gegangene Wert miteinander verglichen werden, sondern der vorangegangene Wert mit dem Wert davor. Anders ausgedrückt: Das Signal wird nicht bei t = n mit dem Signal bei t = n-1 verglichen, sondern t = n-1 mit t = n-2.

Leider ist die Funktion `rising_edge` hier nicht einsetzbar, in diesem Fall würde der Synthesizer XST davon ausgehen, dass es sich um ein Taktsignal handelt. Da das nicht zutrifft, würde der Synthesizer die Operation mit einer Fehlermeldung quittieren.

Wir haben die Flankenerkennung als separates Modul (eine Funktion in VHDL, siehe **Listing 1**) zum Projekt hinzugefügt, obwohl dies nicht unbedingt nötig ist. Module lassen sich jedoch unkompliziert auch in andere Projekte übernehmen.

Am Anfang des Moduls stehen die Instruktionen **`library`** und **`use`**. Diese Anweisungen bewirken, dass das Modul über die nötigen Standardfunktionen und Standardsignale verfügt. Falls nötig können hier Bibliotheken hinzugefügt werden, wie zum Beispiel `numeric_std` für die Operation „+“ (siehe oben...). Anders als in den meisten Programmiersprachen sind diese Instruktionen nicht global für das gesamte Programm gültig, sondern nur für den nächsten Block **`entity`** und den zugehörigen Block **`architecture`**. Normalerweise gehen jedem Block **`entity`** Instruktionen zum Einbinden von Bibliotheken voraus.

Ein Block **`entity`** darf im Prinzip als Symbol betrachtet werden, das in ein Funktionsschema eingebaut werden kann. Hier werden die Eingänge und Ausgänge der Komponente im Abschnitt **`port`** definiert. Alle Signale sind vom Typ `std_logic` der Bibliothek `std_logic_1164`. Das bedeutet, dass es sich um logische Signale handelt, sie können die in der Bibliothek definierten Werte annehmen (unter anderem 0, 1 oder Z). Wenn den Signalen **`in`** folgt, sind Moduleingänge gemeint, wenn **`out`** folgt, handelt es sich um Ausgänge.

Einem Block **`entity`** schließt sich generell ein Block **`architecture`** an (oder auch mehrere), dort wird die Funktion festgelegt. Dieser Block hat eine Bezeichnung (`behavioral`), er ist eine Implementierung der angegebenen Entität `edge_detector`. Der Name `behavioral` wird vergeben, wenn ISE ein VHDL-Modul generiert. Gebräuchlich ist auch `rtl` (von *Register Transfer Level*, die Ebene, in die der VHDL-Code compiliert wird). Die Vergabe eines anderen Namens ist ebenfalls möglich.

Die Flankenerkennung ist als **`process`** implementiert, der vom Signal `clock` abhängt. Immer wenn `clock` seinen Wert ändert, werden die zum Prozess gehörenden Instruktionen ausgeführt.

Listing 1. Das Modul der Flankenerkennung.

```
library ieee;
use ieee.std_logic_1164.all;

entity edge_detector is
  port (input : in std_logic;
        clock : in std_logic;
        rise : out std_logic);
end edge_detector;

architecture behavioral of edge_detector is
begin
  process (clock)
    variable history : std_logic_vector(1 to 3);
  begin
    if rising_edge(clock) then
      rise <= history(2) and not history(3);
      history := input & history(1 to 2);
    end if;
  end process;
end behavioral;
```

Nach der Deklaration des Prozesses folgt eine Liste der Variablen, die innerhalb des Prozesses, jedoch nicht außerhalb benutzt werden. Hier existiert nur die Variable `history`, dies ist ein 3-bit-Schieberegister. Die Variable ist ein Vektor, was

Listing 2. DCF77-Decoder mit Flankenerkennung.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity dcf77_decoder is
    port ( input : in std_logic;
           clock : in std_logic;
           data : out std_logic_vector (58 downto 0) );
end dcf77_decoder;

architecture behavioral of dcf77_decoder is

  component edge_detector is
    port (input : in std_logic;
          clock : in std_logic;
          rise : out std_logic);
  end component edge_detector;

  constant t1750ms: integer := 14000000; -- 1750 ms @ 8 MHz
  constant t150ms: integer := 1200000; -- 150 ms @ 8 MHz
  signal counter : integer := 0;
  signal bits : std_logic_vector(58 downto 0) := (others => '0');
  signal input_rise : std_logic := '0';

begin
  edge_detect: edge_detector port map (input => input,
                                       clock => clock,
                                       rise => input_rise);

  process (clock) is
  begin
    if rising_edge(clock) then
      counter <= counter + 1;
      if (counter=t150ms) then
        bits <= input & bits(58 downto 1);
      end if;
      if input_rise='1' then
        if (counter>=t1750ms) then
          data <= bits; -- Transfer data.
        end if;
        counter <= 0; -- Clear counter.
      end if;
    end if;
  end process;
end behavioral;
```

bedeutet, dass sie aus mehreren Bits besteht. Definiert ist der Vektor als `(1 to 3)`, während das Schieberegister des DCF77-Decoders die Definition `(58 downto 0)` hat. Natürlich dürfen die einzelnen Bits nicht miteinander verwechselt werden. Die Flankenerkennung ist nur nach den ansteigenden Flanken des Taktsignals aktiv. Der Wert des Ausgangssignals `rise` wird abhängig von den Bits 2 und 3 der Variablen `history` bestimmt, anschließend wird der aktuelle Wert des Eingangssignals in das Schieberegister übernommen. Das geschieht in gleicher Weise wie beim DCF77-Decoder.
Jeder Block wird mit dem Statement **end** abgeschlossen (beispielsweise **end if**), gegebenenfalls folgt die Bezeichnung des Blocks. Hier macht sich eine kleine und deshalb hinnehmbare Schwäche von VHDL bemerkbar: Die Tipparbeit ist umfangreicher als bei Verilog.
Aus **Bild 2** geht hervor, wie ISE die Flankenerkennung als Schema darstellt. Wie es scheint, betreibt ISE das Erstellen von Funktionsschemen aus VHDL-Code bisweilen nicht mit der gebotenen Präzision.
Die Flankenerkennung haben wir vorstehend ausführlich beschrieben, weil dieses Modul beispielgebend für den Grundaufbau von VHDL-Modulen ist. Module haben in VHDL stets die Struktur **library**, **entity** und **architecture**, dies gilt auch für den DCF77-Decoder.

Module anwenden

Damit die Flankenerkennung in den DCF77-Decoder eingebaut wird, muss XST über diesen Wunsch informiert werden. Zu den verbreiteten Methoden gehört das Einbinden einer Bibliothek, wir haben hier jedoch die Component-Methode gewählt. Aus **Listing 2** ist ersichtlich, wie sich das Ganze bewerkstelligen lässt. Dort ist auch das DCF-Decoder-Modul enthalten, einschließlich des Blocks **entity** und des zugehörigen Blocks **architecture**, zusammen mit den Eingangs- und Ausgangssignalen, den lokalen Variablen und den Konstanten. Die Flankenerkennung bauen wir in das Modul ein, indem wir sie zu Beginn des Blocks **architecture** als **component** deklarieren. Dazu kopieren wir den Block **entity** und ersetzen das Wort **entity** durch **component**. Danach müssen wir diese Komponente anschließen. Das geschieht, indem wir nach **begin** im Block **architecture** eine Kopie instanziieren. Wir müssen ein Label wählen und anschließend in einer **port map** angeben, welche Signale (Gatter) des DCF77-Decoders (in der *map* rechts) mit den Signalen (Gattern) der Flankenerkennung (in der

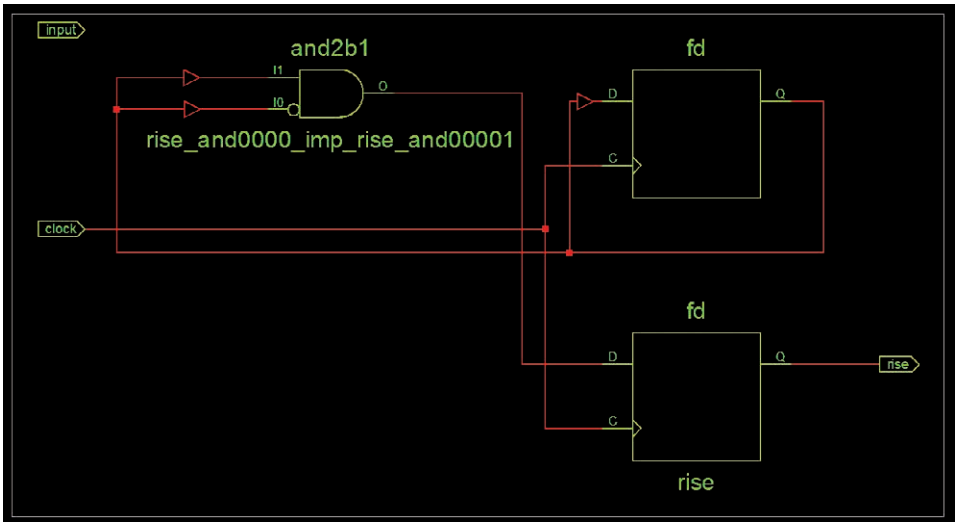


Bild 2.
Dieses überraschende Schema entsteht, wenn der RTL Viewer von ISE die Flankenerkennung erstellt (Tab *Design*, *Synthesize – XST → View RTL Schematic*). Das Signal *input* schwebt, insgesamt ist die Funktionsweise unklar. Wo liegt der Fehler?

map links) gekoppelt werden müssen. Das Taktsignal und das Eingangssignal sind bereits vorhanden, denn sie betreffen auch die Eingänge (Gatter) des DCF77-Decoders (siehe dort Block **entity**). Für das Ausgangssignal `rise` der Flankenerkennung müssen wir ein lokales Signal des gleichen Typs hinzufügen, wir haben es `input_rise` genannt. Mit diesem Signal können wir nun in unserem Prozess arbeiten. Wenn eine ansteigende Flanke im DCF77-Signal erkannt wird, erhält Signal `input_rise` für die Dauer einer Taktperiode den Wert 1.

Falls Sie versäumen, das Eingangssignal einer Komponente zu „mappen“, kann eine etwas kryptische Fehlermeldung erscheinen. Die Fehlermeldung will darauf hinweisen, dass zum fehlenden Signal kein Default-Wert existiert, es muss deshalb angeschlossen werden.

Display

Die Länge dieses Beitrags muss im Rahmen bleiben, trotzdem wollen wir in dieser Folge vollständig beschreiben, wie unsere Anwendung „DCF-Deco-

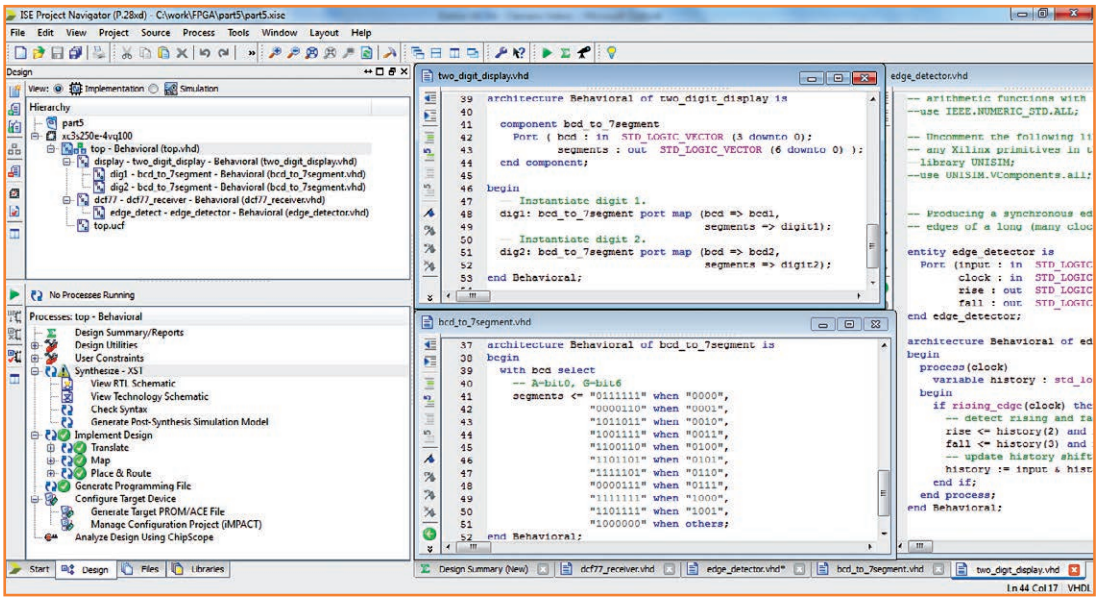


Bild 3.
ISE lässt hier nicht nur den Code des Siebensegment-Displays erkennen, die Hierarchie des Projekts und die grünen Haken beweisen, dass das Projekt zu einer Bit-Datei compiliert werden kann.

der“ in VHDL programmiert werden kann. Da noch einiges vor uns liegt, übergehen wir an dieser Stelle das Umsetzen in den Siebensegment-Code. Wir haben das gleiche zweistellige Display wie in Folge 3 [3] verwendet, für das wir einen BCD-nach-Siebensegment-Umsetzer geschrieben hatten. Das ist eine Übung, wie sie im Lehrbuch steht, deshalb erscheinen uns weitere Erklärungen unnötig. Dem Bildschirmfoto in **Bild 3** sind Details entnehmbar, weitere Informationen enthält die Projektseite [5], die wir zu dieser Folge unserer FPGA-Einführungsreihe ins Netz gestellt haben.

Listing 3. Ebene *top* des DCF77-Decoders.

```
library ieee;
use ieee.std_logic_1164.all;

entity top is
    port ( dcf77_input : in std_logic;
           clk_in : in std_logic;
           hour_month : in std_logic;
           time_date : in std_logic;
           led1 : out std_logic;
           led2 : out std_logic;
           t_sample : out std_logic;
           digit1 : out std_logic_vector (6 downto 0);
           digit2 : out std_logic_vector (6 downto 0) );
end top;

architecture behavioral of top is

  component two_digit_display is
    port ( bcd1 : in std_logic_vector (3 downto 0);
           bcd2 : in std_logic_vector (3 downto 0);
           digit1 : out std_logic_vector (6 downto 0);
           digit2 : out std_logic_vector (6 downto 0) );
  end component;

  component dcf77_decoder is
    Port ( input : in  std_logic;
           clock : in  std_logic;
           tick : out  std_logic;
           sync : out  std_logic;
           data : out  std_logic_vector (58 downto 0) );
  end component;

  signal data : std_logic_vector (58 downto 0);
  signal bcd1 : std_logic_vector (3 downto 0);
  signal bcd2 : std_logic_vector (3 downto 0);
  signal tick : std_logic;

begin
  display: two_digit_display port map (bcd1 => bcd1,
                                       bcd2 => bcd2,
```

Top ist oben

Wie schon in der vorangegangenen Folge laufen alle Fäden in der Ebene *top* zusammen, an die Stelle eines Funktionsschemas tritt jedoch ein VHDL-Modul. Diese Eigenschaft muss in ISE über die *Design Properties* festgelegt werden. Zuerst generieren Sie, wie zu Beginn der Folge 3 [3] beschrieben, auf der Basis des letzten Projekts ein neues Projekt. Löschen Sie alle Schemas, behalten Sie jedoch die UCF-Datei bei. Öffnen Sie die *Design Properties* (beispielweise über das Menü *Project*) und setzen Sie den *Top-Level Source Type* auf HDL und die *Preferred Language* auf VHDL (falls noch nicht geschehen).

Nun können Sie die neuen Quelldateien hinzufügen. Öffnen Sie das Menü *Project* oder klicken Sie mit der rechten Maustaste auf den Tab *Design*. Wählen Sie *New Source…*, anschließend *VHDL Module*, geben Sie den Namen der Datei ein (zum Beispiel „top”), achten Sie darauf, dass *Add to project* angehakt ist, und klicken Sie auf *Next*. Jetzt erscheint eine Maske, die Sie ausfüllen können, wenn Sie die Eingangs- und Ausgangssignale bereits kennen. Anderenfalls lassen Sie die Maske leer. Fahren Sie fort, indem Sie auf *Next*

```
                                    digit1 => digit1,
                                    digit2 => digit2);

  dcf77: dcf77_decoder port map (input => dcf77_input,
                                 clock => clk_in,
                                 tick => tick,
                                 sync => led2,
                                 data => data);

  process (clk_in) is
  begin
    t_sample <= tick;
    led1 <= tick;
    if rising_edge(clk_in) then
      if time_date='1' then
        -- Show time.
        if hour_month='1' then
          -- Show hours.
          bcd1 <= data(32 downto 29);
          bcd2 <= "00" & data(34 downto 33);
        else
          -- Show minutes.
          bcd1 <= data(24 downto 21);
          bcd2 <= "0" & data(27 downto 25);
        end if;
      else
        -- Show date.
        if hour_month='1' then
          -- Show month.
          bcd1 <= data(48 downto 45);
          bcd2 <= "000" & data(49 downto 49);
        else
          -- Show day of month.
          bcd1 <= data(39 downto 36);
          bcd2 <= "00" & data(41 downto 40);
        end if;
      end if;
    end if;
  end process;

end behavioral;
```

und dann auf *Finish* klicken. ISE generiert nun eine Datei, in der sich ein Abschnitt zum Einfügen Ihres VHDL-Codes befindet. Tragen Sie so viele Dateien ein, wie Sie Module erzeugen wollen. In unserem Projekt sind dies fünf Module: top, DCF77-Decoder, Flankenerkennung, BCD-nach-Siebensegment-Decoder und 2-Digit-Display.

Das Modul *top* ist in **Listing 3** zusammengefasst. Die Bezeichnungen der Eingangs- und Ausgangssignale von *top*, also die im Block **entity** des Moduls *top* aufgeführten Signale, müssen mit den Bezeichnungen in der UCF-Datei übereinstimmen. Die Namensgleichheit stellt die Kopplungen zwischen den FPGA-Anschlüssen und dem VHDL-Code her. Alle in der UCF-Datei enthaltenen Signale müssen vorhanden sein, anderenfalls gibt ISE eine Fehlermeldung aus. Neu ist hier die Anwendung von Vektoren für die Steuerung der Display-Anschlüsse. In der UCF-Datei geschieht dies durch Indizes zusammen mit dem Vektornamen, beispielsweise ist `digit1(0)` das Bit 0 des Vektors `digit1` in *top*.

Im Block **architecture** von *top* werden die Komponenten `two_digit_display` und `dcf77_decoder` aufgerufen, jede Komponente arbeitet mit einem gesonderten Vektor. Bei größeren, beispielsweise sechsstelligen Displays kann `two_digit_display` dreifach (mit unterschiedlichen Labeln) instanziiert und angeschlossen werden. Ferner haben wir noch weitere Signale definiert, die nur innerhalb von *top* eine Bedeutung haben. Zum Beispiel lässt Signal `tick` die LED1 bei jedem empfangenen Bit kurz aufleuchten. Da diese LED nicht über einen Pin des FPGA-Board zugänglich ist, haben wir die LED auch an das Signal `t_sample` gekoppelt. Dieses Signal ist in der UCF-Datei mit Pin P86 des FPGA verknüpft.

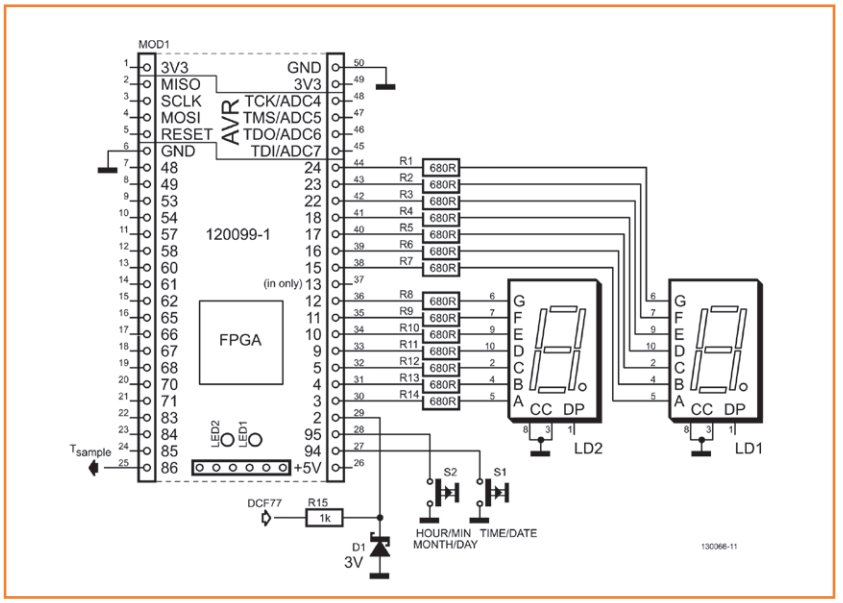
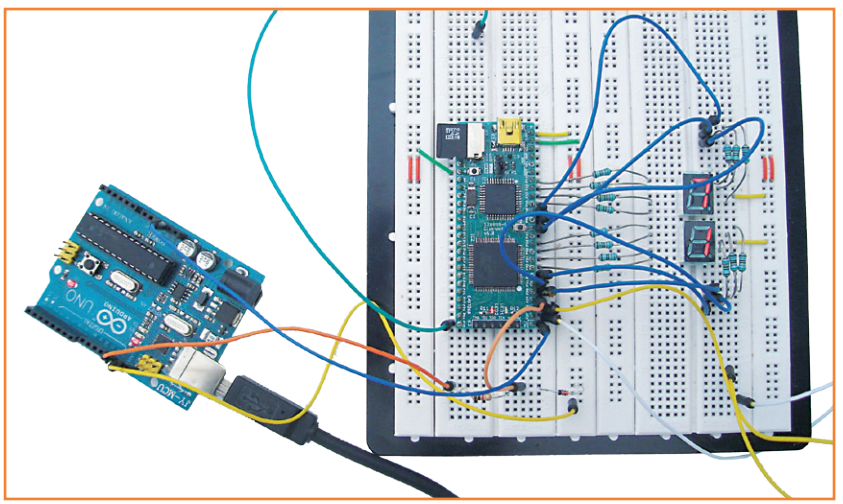


Bild 4. Die Hardware des DCF77-Decoders ist weitgehend identisch mit der Hardware aus Folge 3 [3]. Da der FPGA mit 3,3 V arbeitet, begrenzt eine Zenerdiode das FPGA-Eingangssignal auf etwa 3 V.

Bild 5. Versuchsaufbau, mit einem Arduino-Board, das als DCF77-Simulator dient. Das Display zeigt die Stunden an, es ist hier kurz nach 17 Uhr. Die Stromversorgung übernimmt das Arduino-Board.



Auf einem angeschlossenen Oszilloskop lässt sich der Abtastzeitpunkt innerhalb des DCF77-Eingangssignals beobachten.

Der Prozess von *top* ist wenig spektakulär. Er kann als Multiplexer betrachtet werden, der abhängig von den Steuersignalen `time_date` (P94) und `hour_month` (P95) schaltet. Mit den Signalen ist wählbar, ob Stunde, Minute, Tag oder Monat auf dem zweistelligen Siebensegment-Display erscheint. Die relevanten Bits werden dem DCF77-Schieberegister entnommen und zu zwei 4-bit-BCD-Feldern kombiniert. Anschließend werden die BCD-Informationen in Siebensegment-Signale umgesetzt. Anzumerken ist hier, dass die Vektoren `bcd1` und `bcd2` nicht explizit mit der Display-Komponente gekoppelt werden müssen, dies ist bereits über die **`port map`** geschehen.

Während wir unseren VHDL-Entwurf synthetisieren, erscheinen auf dem Bildschirm diverse Warnungen. Dies hat seine Ursache darin, dass nicht alle Bits des DCF77-Schieberegisters in Gebrauch sind und dass Bit 3 des Vektors `bcd2` immer 0 ist. Dieses Bit ist 0, weil die Datenfelder aus weniger als 8 Bits bestehen (nur das Feld „Jahr“ hat 8 Bits, es wird hier nicht verwendet).

Was auffällt, ist die ausgeprägte Hierarchie im Tab *Design* (Bild 3), ähnlich wie beim Erstellen einer FPGA-Anwendung in Gestalt eines Funktionsschemas. Wir haben schon erwähnt, dass in ISE der so genannte *RTL Viewer* integriert ist. Er kann VHDL-Code als grafisches Funktionsschema darstellen (siehe Bild 2). Damit ist der Kreis geschlossen: Ein Funktionsschema ist funktional identisch mit VHDL-Code, VHDL-Code ist funktional identisch mit einem Funktionsschema.

Wir haben uns bemüht, Ihnen alle Informationen an die Hand zu geben, die Sie zum Erstellen von FPGA-Anwendungen in VHDL zwingend benötigen. Uns bleibt nur noch, Ihnen viel Erfolg zu wünschen!

(130066)gd

Weblinks

[1] Teil 1: www.elektor.de/120099
[2] Teil 2: www.elektor.de/120630
[3] Teil 3: www.elektor.de/120743
[4] Teil 4: www.elektor.de/130065
[5] Teil 5: www.elektor.de/130066

Von BASIC nach Python (2)

Ein Erfahrungsbericht

Von
Jean-Claude Feltes
(LU)

Im ersten Teil haben wir beschrieben, was bei Python anders als bei BASIC ist. Wir haben gezeigt, wie die Installation funktioniert und auch erste Programme laufen lassen. Nun geht es weiter mit Diagrammen und der Fourier-Synthese. Zum Schluss gehen wir noch auf grafische Benutzerschnittstellen ein.

Wenn es richtig ist, dass Python eine für die Elektronik gut geeignete Programmiersprache ist, dann wird man als Elektroniker damit sicherlich häufig Daten visualisieren. Denn Kurven sind nun mal für den menschlichen Geist einfacher zu interpretieren als nackte Zahlenkolonnen. Damit man nun nicht alle Räder immer wieder von neuem erfinden und jedes Bit einer Kurve „manuell" setzen muss, gibt es für viele Programmiersprachen geeignete Bibliotheken, die einem viel Kleinarbeit abnehmen. Python ist da keine Ausnahme.

Diagramme

Bei Python hört das Standard-Modul für 2D-Diagramme auf die Bezeichnung „Matplotlib". Während bei Python selbst die Devise gilt: „Für jede Aufgabe nur eine einzige Methode!", ist das bei den Zusatzmodulen leider anders, was mich schon viele Stunden Arbeit kostete. Speziell bei Matplotlib gibt es ein einfaches prozedurales und ein komplizierteres objektorientiertes Interface. Beispiele im Internet und in Büchern benutzen entweder die eine oder die andere Methode, so dass die gezeigten Listings leicht verwirren können.
Das einfache Interface „pyplot" erlaubt auch sehr einfache Programme. Das Programm von **Listing 1** erzeugt eine gedämpfte Sinusschwingung und zeichnet die sich ergebende Kurve grafisch in ein Fenster (siehe **Bild 1**).
Die erste Zeile importiert das Interface „pyplot" als Objekt „plt". Für mathematische Aufgaben ist das Modul „Numpy" gedacht. In der zweiten Zeile werden davon drei Funktionen importiert.
Mit „linspace" kann ein Intervall (hier von 0 bis 7) in beliebig viele (hier 1.000) Teile zerlegt und als Vektor (Array) dargestellt werden. Dies erlaubt eine schnelle und einfache Berechnung der Funktionswerte. Numpy-Funktionen können auch Vektoren verarbeiten. Hier werden in der Zeile:

```
y= sin(5*x)*exp(-x)
```

alle 1.000 Werte des y-Vektors auf einmal berechnet.
Damit kann man wunderbar auf For-Schleifen verzichten und den Code schnell und übersichtlich gestalten.
Mit dem Befehl „plt.plot(x,y)" werden die Daten als Kurve dargestellt. Damit das Diagramm schließlich sichtbar wird, muss der Befehl „plt.show()" ausgeführt werden. Will man mehrere Kurven in einem Diagramm darstellen, braucht man die Plot-Funktion nur wie in **Listing 2** mehrmals aufzurufen.
Das Diagrammfenster (siehe Bild 1) enthält übrigens automatisch eine Toolbar-Leiste, die ein Zoomen und Sichern des Diagramms erlaubt. Außerdem werden die Koordinaten unter dem Cursor angezeigt. Will man aber ein Diagramm ordentlich mit GUI-Elementen aufpeppen, wird es komplizierter. Dann sollte man objektorientiert vorgehen.

Beispiel: Frequenzgang

Möchte man beispielsweise die Durchlasskurve eines RC-Tiefpasses darstellen, dann kann man den komplexen Frequenzgang *F* nach der Spannungsteilerformel mit komplexen Impedanzen ermitteln:

$$F = 1 / (1 + j \omega R C)$$

Die Durchlasskurve entspricht dann dem Betrag von *F* als Funktion der Frequenz (siehe **Bild 2**). Bei diesem Beispiel kann man sich von Pythons Fähigkeiten im Umgang mit komplexen Zahlen unterstützen lassen.
Im Programm nach **Listing 3** wird zunächst das Array „f" mit logarithmisch gestuften Frequenzen berechnet. Bei einer linearen Verteilung würde man hier „linspace()" verwenden. Anschließend wird das Array der komplexen F-Werte und das Array des Betrags „Fabs" berechnet. Auch hier glänzt Numpy wieder durch seine Vektorfunktionen. Man braucht keine For-Schleife zur Berechnung und der Code wird kurz und übersichtlich. Zum Schluss wird das Diagramm gezeichnet, wobei die Frequenzachse passenderweise logarithmisch skaliert wird. Für eine schöne Darstellung ist es wichtig, das Gitter richtig zu setzen:

```
ax.grid(True, which = "both", linestyle
= "-")
```

Hier wird das Gitter eingeschaltet, mit „both" werden beide Gitter (Haupt- und Nebengitter für die Unterteilung) aktiviert und zuletzt werden durchgezogene Linien gewählt.
Durch Hinzufügen der folgenden Code-Zeilen:

```
# plot phi = f(f)
phi = angle(F)*180.0/pi
ax2 = fig.add_subplot(212)
ax2.plot(f, phi)
ax2.grid(True, which = "both", linestyle
= "-")
ax2.set_xscale ("log")
ax2.set_xlabel("f/Hz")
ax2.set_ylabel("phi/degrees")
```

kann man auch noch den Phasengang ergänzen.

Beispiel: Fourier-Synthese

Elektroniker, die Mathe mögen, wird es freuen zu hören, dass auch umfangreichere Dinge wie die Fourier-Synthese in Python recht handlich ausfallen können.
Im Beispiel von **Listing 4** wird durch die geeignete Gewichtung von 30 Harmonischen eine schon recht gut einer Rechteck-Welle entsprechende Kurve synthetisch erzeugt. **Bild 3** zeigt, wie ansprechend das Ergebnis des doch verhältnis-

Listing 1: Sinus.py

```
import matplotlib.pyplot as plt
from numpy import sin, exp, linspace

x=linspace(0.0, 7.0, 1000)
y= sin(5*x)*exp(-x)

plt.plot(x, y)
plt.show()
```

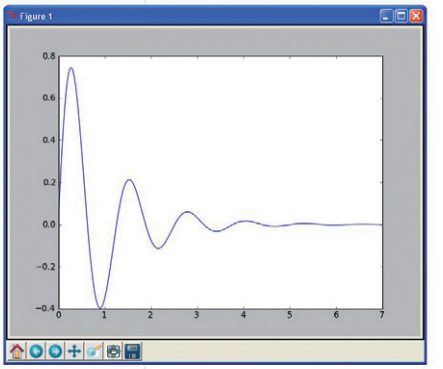


Bild 1.
Grafische Darstellung einer gedämpften Sinusschwingung nach Listing 1.

Listing 2: Multigraph.py

```
mport matplotlib.pyplot as plt
from numpy import sin, exp, linspace

x=linspace(0.0, 7.0, 1000)
y1 = sin(5*x)*exp(-x)
y2 = y1* 0.5

plt.plot(x, y1)
plt.plot (x, y2)
plt.show()
```

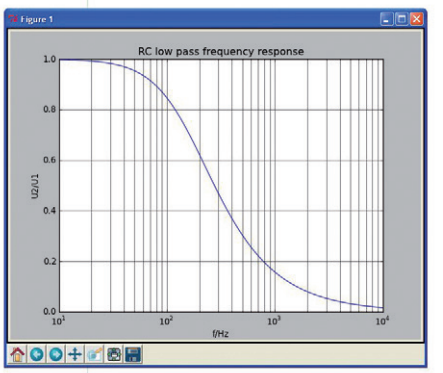


Bild 2.
Der Frequenzgang eines RC-Tiefpasses nach Listing 3.

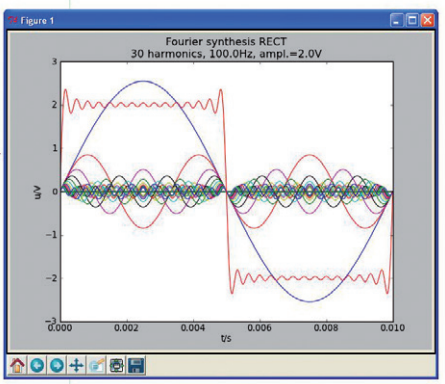


Bild 3.
Fourier-Synthese einer Rechteck-Welle nach Listing 4.

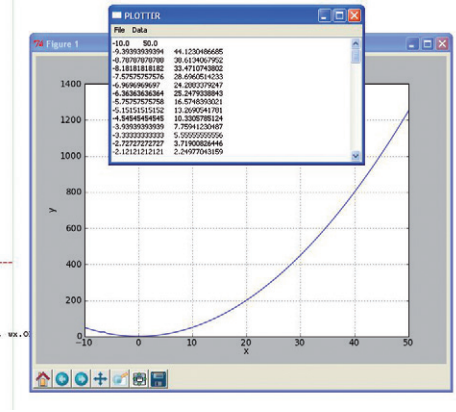


Bild 4.
Daten-Plotter: Terminal-Fenster mit Zahlenwerten und grafische XY-Darstellung.

mäßig begrenzten Code-Aufwands ist. Im Code lassen sich auch mehr oder weniger Harmonische einstellen.

To GUI or not to GUI?

Eingeschworene Linux-User stehen im Ruf, auf GUIs mit der verächtlichen Bemerkung über „Klicki-Bunti“ zu reagieren und Mäuse für Teufelswerk zu halten. Ungeachtet des Wahrheitsgehalts solcher Klischees schätzen sehr viele Menschen schön gestaltete Programme, die sich einfach bedienen lassen. Und ganz ehrlich: Eine grafische Auswahlmöglichkeit für Dateien ist doch unverzichtbar, oder?

Wie dem auch sei: bei Python kann man grafische und nichtgrafische Elemente in einem Programm kombinieren. Beim Programm von **Listing 5** handelt es sich um einen Daten-Plotter. Eingelesene Daten werden mit Print-Befehlen in einem Terminal-Fenster als Text ausgegeben und schließlich in einem grafischen Fenster als Diagramm dargestellt (siehe **Bild 4**).

Bei den GUI-Bibliotheken hat man die Qual der Wahl. Zunächst habe ich „Tkinter“ benutzt, da es beim Python-Interpreter mitgeliefert wird und seine Verwendung einfach zu erlernen ist. Doch beim Kopieren eines Diagramms in die Zwischenablage gab es Probleme, weshalb ich dann auf „wxPython“ umgestiegen bin. Es gibt aber beispielsweise mit „PyQt“ und „GTK“ noch weitere Alternativen.

Der Daten-Plotter bietet sich als geeignetes Bei-

Listing 3: RC.py

```
import matplotlib.pyplot as plt
from numpy import pi, linspace, log10, logspace
from numpy import  complex, abs    # these allow
vector operations

#  EDIT
R = 10.0E3
C = 100.0E-9

# END EDIT

RC = R*C

# create f values equally spaced on a log scale
f = logspace ( 1, 4, 100)   # 100 values from 10**1
to 10**4

# calculate F (complex) and absolute value Fabs
F = 1 / (1 + 1j*  2 * pi * f * RC)
Fabs = abs(F)

# plot Fabs = f(f)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(f, Fabs)
ax.grid(True, which = “both”, linestyle = “-”)
ax.set_xscale (“log”)
ax.set_xlabel(“f/Hz”)
ax.set_ylabel(„U2/U1“)
ax.set_title(“RC low pass frequency response”)
plt.show()
```

spiel zur Verwendung von GUI-Elementen an. Das Programm ermöglicht die Anzeige und das Editieren von in einer Text-Datei abgelegten Messwerten. Die Werte können dabei als zweidimensionales XY-Diagramm dargestellt werden. X- und Y-Werte werden zusätzlich zweispaltig als Zahlen angezeigt.
Der Einfachheit halber wurde auf „pythonische“ objektorientierte Programmierung verzichtet.

Nach dem Import der benötigten Module werden die Funktionen definiert, und dann folgt ganz unten das Hauptprogramm.
Das Hauptprogramm erstellt als erste Maßnahme eine Instanz „app“ des App-Objekts. Als nächstes wird ein Frame-Objekt als Fenster für die Anwendung erstellt. Dieses Fenster wird mit Menü und Text-Box zum Editieren und Anzeigen der Daten versehen. Die Text-Box ist eigentlich

Listing 4: Fourier.py

```
"""FOURIER SYNTHESIS FOR RECT VOLTAGE"""
#-----------------------------------------------
-------
# EDIT HERE

n = 30                # number of harmonics
nb_points=1000           # horizontal resolution
frequency = 100.0   #Hz
amplitude = 2.0     #V

# END OF EDIT AREA
#-----------------------------------------------
--------
print "Importing modules"
import matplotlib.pyplot as plt
from numpy import sin, exp, linspace, pi
from numpy import zeros
#-----------------------------------------------
---------
def calc_amplitude(amplitude, i):
    """ Calculate amplitudes of harmonics"""
    # take only odd harmonics
    if i % 2 == 0:
        ai = 0
    else:
        ai = (4/pi)*amplitude / i
    return ai
#-----------------------------------------------
--------
def calc_harmonics(nb_points, n):
    """ Calculate harmonics and resulting voltage
    returns
    uharm = array nb_points * n
    ug = array nb_points
    """

    # init  arrays for resulting voltage and
harmonics
    ug = zeros(nb_points)
    uharm = zeros((nb_points, n+1))

    # harmonics  and total voltage
    for i in range(1,n+1):
        ai = calc_amplitude(amplitude, i)
        fi = frequency * i
        uharm[:,i] = ai * sin(2 * pi * fi * t )
        ug = ug + uharm[:,i]
    return uharm, ug
#-----------------------------------------------
----------
""" Main program"""

T=1/frequency

# equally spaced time array  for 1 period
t = linspace(0.0, T, nb_points)

# plot harmonics
uharm, ug = calc_harmonics(nb_points, n)
for i in range(1,n+1):
    plt.plot (t, uharm[:,i])

# plot resulting voltage
plt.plot (t, ug)

s=str(n)+" harmonics, "+str(frequency)+"Hz,
ampl.="+str(amplitude)+"V"
plt.title("Fourier synthesis RECT\n"+s)
plt.xlabel(„t/s“)
plt.ylabel(„u/V“)

# make plot visible
plt.show()
```

Listing 5: Dataplot.py

```
#!/usr/bin/env python
""" Plot data from file """
import wx
import os.path
import matplotlib.pyplot as plt

def create_menu(frame):
    # create menu
    menubar = wx.MenuBar()
    # main menus
    mnuFile = wx.Menu()
    mnuData = wx.Menu()
    menubar.Append(mnuFile,"&File")
    menubar.Append(mnuData,"&Data")
    # submenus
    m_Open = mnuFile.Append(-1,"&Open")
    mnuFile.AppendSeparator()
    m_Exit = mnuFile.Append(-1,"E&xit")
    m_Plot = mnuData.Append(-1,"&Plot")
    # attach menu to frame
    frame.SetMenuBar(menubar)

    # bind menu events to procedures
    frame.Bind(wx.EVT_MENU, OnExit, m_Exit)
    frame.Bind(wx.EVT_MENU, OnOpen, m_Open)
    frame.Bind(wx.EVT_MENU, OnPlot, m_Plot)
#---------------------------------------------------------------------------
# Event handlers
def OnExit(event):
    frame.Close()

def OnOpen(event):
    # ask for filename
    dlg = wx.FileDialog(None,"Open data file", os.getcwd() ,"", "*.*", wx.OPEN)
    dlg.ShowModal()
    filename = dlg.GetPath()

    # open file, get data and  put it into textbox
    try:
        f = open(filename, "r")
        data = f.read()
        f.close()
        textbox.SetValue(data)
    except:
        wx.MessageBox("Could not open file!")
```

```
def OnPlot(event):
    # plot data
    x,y = fill_xy_with_values(textbox)
    plot_xy(x, y)
#---------------------------------------------------------------------------
def fill_xy_with_values(textbox):
    # """ get values from textbox
    #    returns  arrays x, y  and number of data points"""
    text=textbox.GetValue()
    lines=text.splitlines()

    x=[]
    y=[]
    for line in lines:
        columns = line.split()      #separator can be one or more " " or "\t"
        x.append (float(columns[0]))
        y.append (float(columns[1]))

    return x,y
#---------------------------------------------------------------------------
def plot_xy( x, y):
    """ Plot arrays x, y with matplotlib"""
    plt.figure(1)
    plt.subplot(111)                # 1 row, 1 col, plot nb. 1
    plt.grid(True)
    plt.plot(x, y)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()
#---------------------------------------------------------------------------
#   Main
#---------------------------------------------------------------------------
app = wx.App()

# create frame
frame = wx.Frame(None, title='PLOTTER', pos=(350,300))

create_menu(frame)
# editor textbox for data
textbox=wx.TextCtrl(frame, style = wx.TE_MULTILINE)

# show frame and run event loop
frame.Show()
app.MainLoop()
```

schon ein kleiner Editor: Hier können Werte editiert oder hinzugefügt werden, und sogar das Handling der Zwischenablage ist mit den üblichen Tastenkürzeln möglich. Die rechte Maustaste liefert ein Popup-Menü mit den klassischen Editierfunktionen.

Mit „frame.Show()“ wird das Fenster angezeigt und mit „app.MainLoop()“ eine Event-Loop gestartet. Das Programm wartet dabei auf ein Ereignis (Mausklick oder Tastendruck) und verzweigt dann auf die Event-Handler (Funktionen, die beim Eintreten der Ereignisse aufgerufen werden).

Die erste Funktion definiert die Menüs und deren Bindung zu den mit „On“ beginnenden Event-Handlern. Die Funktionen „OnExit“, „OnOpen“ und „OnPlot“ werden per Menü oder passender Taste aufgerufen. Bei „OnOpen“ wird zunächst der Dateiselektor „wx.FileDialog“ aufgerufen. Es wird hier noch „os.getcwd“ (**get c**urrent **w**orking **d**irectory) benutzt, um die Datei zunächst im aktuellen Verzeichnis zu suchen.

Wenn der Dateiname gewählt ist, wird die Datei versuchsweise geöffnet. Hier wird mit „try – except“ gearbeitet, um Dateifehler abzufangen. Lässt sich die Datei öffnen, wird der ganze Inhalt in die Variable „data“ gelesen und in die Text-Box eingefügt.

Hier können die Messwerte nun editiert, kopiert oder eingefügt werden. Natürlich kann man hier auch Werte eingeben statt sie aus einer Datei zu laden.

Bei „OnPlot“ sollen die Daten als Diagramm dargestellt werden. Hierzu werden die Funktionen „fill_xy_with_values“ und „plot_xy“ aufgerufen. Erstere liest die Daten aus der Text-Box und spaltet sie in ein Array von Zeilen auf. In der For-Schleife wird über alle Zeilen iteriert. Diese werden mit „line.split()“ weiter aufgespalten und die so erhaltenen einzelnen Werte in die Arrays „x[]“ und „y[]“ eingefügt. Diese werden als Funktionswerte zurückgegeben.

Die Funktion „plot_xy“ zeichnet das Diagramm (Bild 4). Das kleine Programm ist natürlich noch recht rudimentär, kann aber schon vieles, für das in Visual Basic wesentlich mehr Aufwand getrieben werden muss. Man kann es leicht selbst um einen Menüpunkt zum Speichern von editierten Daten ergänzen.

Fazit & Ausblick

An dieser Stelle sollten Sie einen grundlegenden Eindruck von der einfachen Handhabung und der Leistungsfähigkeit von Python haben. Wenn Ihnen Python gefällt, dann können Sie leicht über das in diesen Beispielen gebotene Niveau hinausgehen. Schließlich eignet sich Python sehr gut für Aufgaben wie die klassische Datenerfassung und Verarbeitung. Sachen wie Filter oder eine FFT stellen mit Python keine große Hürde dar, und man kann eigene Projekte damit recht einfach mit einer grafischen Benutzerschnittstelle aufwerten. In der nächsten Ausgabe zeigen wir, wie sich mit Python eine einfache Messwerterfassung und Steuerung über RS485 und den ElektorBus programmieren lässt.

(120143)

Über den Autor

Jean-Claude Feltes unterrichtet als Diplom-Ingenieur für Elektronik am Lycée Technique des Arts et Métiers in Luxemburg. Diese Berufsschule für Technik und Kunst führt zum Gesellen, Techniker oder BTS-Abschluss. Er beschäftigt sich auch in seiner Freizeit viel mit Elektronik und Programmierung (siehe [2]).

Weblinks & Literatur

[1] Listings etc.:
www.elektor.de/120143

[2] Homepage des Autors:
http://staff.ltam.lu/feljc/home.html

[3] Doku zu Python:
https://pypi.python.org/pypi/RPi.GPIO

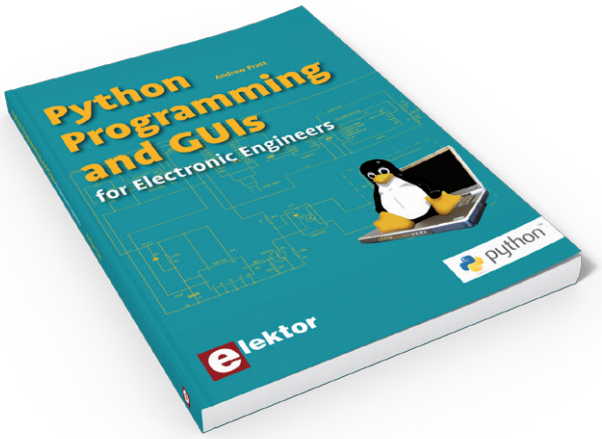
[4] Python-Tutorials:
www.awaretek.com/tutorials.html

[5] Referenz:
Michael Weigend: „Python gepackt“

[6] Einführung:
J.M. Hughes: „Real World Instrumentation with Python“

[7] Verfügbare Module im Python-Package:
http://pypi.python.org/pypi

[8] Python für Elektroniker:
Andrew Pratt: „Python Programming and GUIs for Electronic Engineers“
www.elektor.de/python-programming



elektor ACADEMY
the school of electronics

NEU

Rapid SMD-Prototyping mit PCB-POOL und TARGET 3001! (1-tägiges Seminar)

Elektor bietet in Zusammenarbeit mit Beta Layout und IBF Friedrich die Möglichkeit eines eintägigen praxisbezogenen Seminars zum Thema SMD-Prototyping. Während Sie sich am Vormittag mit der Erstellung eines Leiterplattenprojekts auseinandersetzten, so haben Sie schon am Nachmittag die Gelegenheit durch die Benutzung von Stencils und einen Reflow Kit Ofen einzelne SMD Bestandteile zu löten. Letztendlich erstellen Sie mit Unterstützung von Experten Ihren eigenen Prototypen. Zudem erhalten Sie während des Seminars eine **TARGET 3001!-PCB-POOL Edition** (Vollversion), eine Software die auch schon beim Hand-on Seminar benutzt werden wird.

Referenten: Harald Friedrich und Gernot Seeger

Sonderangebot für Elektormitglieder: 149,00 € (inkl. MwSt.)
Teilnahmegebühr für Nicht-Mitglieder: 299,00 € (inkl. MwSt.)

Echtzeitbetriebssysteme in Theorie und Praxis (3-tägiges Seminar)

Das modular aufgebautes Seminar ist je nach Interesse und Wissenstand konzipiert, so dass Sie beliebig einsteigen können.

Tag 1 **Was ist Echtzeit? Was ist FreeRTOS? Installation von Werkzeugen, tasks, scheduling, queues,...**
Tag 2 **Interrupts, Nebenläufigkeit (concurrency), gegenseitiger Ausschluss (mutual exclusion), Speicherverwaltung,...**
Tag 3 **allgemeine Methoden zur Vermeidung von Fehlern, Wie kann man die Fehler, die man nicht vermeiden konnte, finden (mit Schwerpunkt Echtzeitbetriebssysteme)?**

Ziel des Kurses ist es, Ihnen die grundlegenden Konzepte eines Echtzeitbetriebssystems und dessen Handhabung zu vermitteln. Was sind zum Beispiel Vor- und Nachteile? Wie kann man diverse Programme auf einem PC erstellen/kompilieren und auf einem eingebetteten System ausführen? Eine Kombination aus Theorie und praktischen Übungen wird es Ihnen ermöglichen, das neu erworbene Wissen bei Eigenentwicklungen einzusetzen.

Referent: Robert Berger - Teilnahmegebühr: für alle 3 Seminar-Tage: 1.799,00 € (inkl. MwSt.)
für 2 Seminar-Tage: 1.250,00 € (inkl. MwSt.)
für 1 Seminar-Tag: 650,00 € (inkl. MwSt.)

Kommerzielle Verwendung von Open-Source-Software (1-tägiges Seminar)

Open-Source-Software (OSS) dringt weiter in die Embedded-Welt vor. Die hohe Leistungsfähigkeit von Prozessoren und Bausteinen macht mittlerweile den Einsatz anspruchsvoller Software darauf möglich.
Der Kurs soll einen Überblick über die Möglichkeiten und Grenzen der kommerziellen Verwendung von Open-Source-Software liefern. Den Seminarteilnehmern werden dazu die wichtigsten Rechtsgebiete vermittelt, wie das Immaterialgüterrecht oder das Vertragsrecht. Praktische Beispiele erläutern die Umsetzbarkeit und den Erfolg dieses Geschäftsmodells. Gleichzeitig sollen auch die rechtlichen Schwierigkeiten behandelt werden, welche die OSS- Lizenzierung mit sich bringt.

Referent: Bernd Suchomski Teilnahmegebühr: € 399,00 (inkl. MwSt)

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.

Echtzeitbetriebssysteme in Theorie und Praxis
Hanau: 03.06. bis 05.06.2013
Webinar: 13. Bis 15.05.2013

Linux Debugging
Hanau: 06.06. + 07.06.2013
Dortmund: 19.09. + 20.09.2013
München: 12.12. + 13.12.2013

Rapid SMD-Prototyping mit PCB-POOL und TARGET 3001!
Hanau: 05.06.2013
München: 11.09.2013

Arduino-Programmierung und Projektentwicklung
Dortmund: 05.09.2013
Zürich (CH): 07.09.2013

Kommerzielle Verwendung von Open-Source-Software
Hanau: 20.06.2013
Augsburg: 15.07.2013
München: 12.09.2013
Dortmund: 19.09.2013

INKLUSIVE elektor ZERTIFIKAT

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

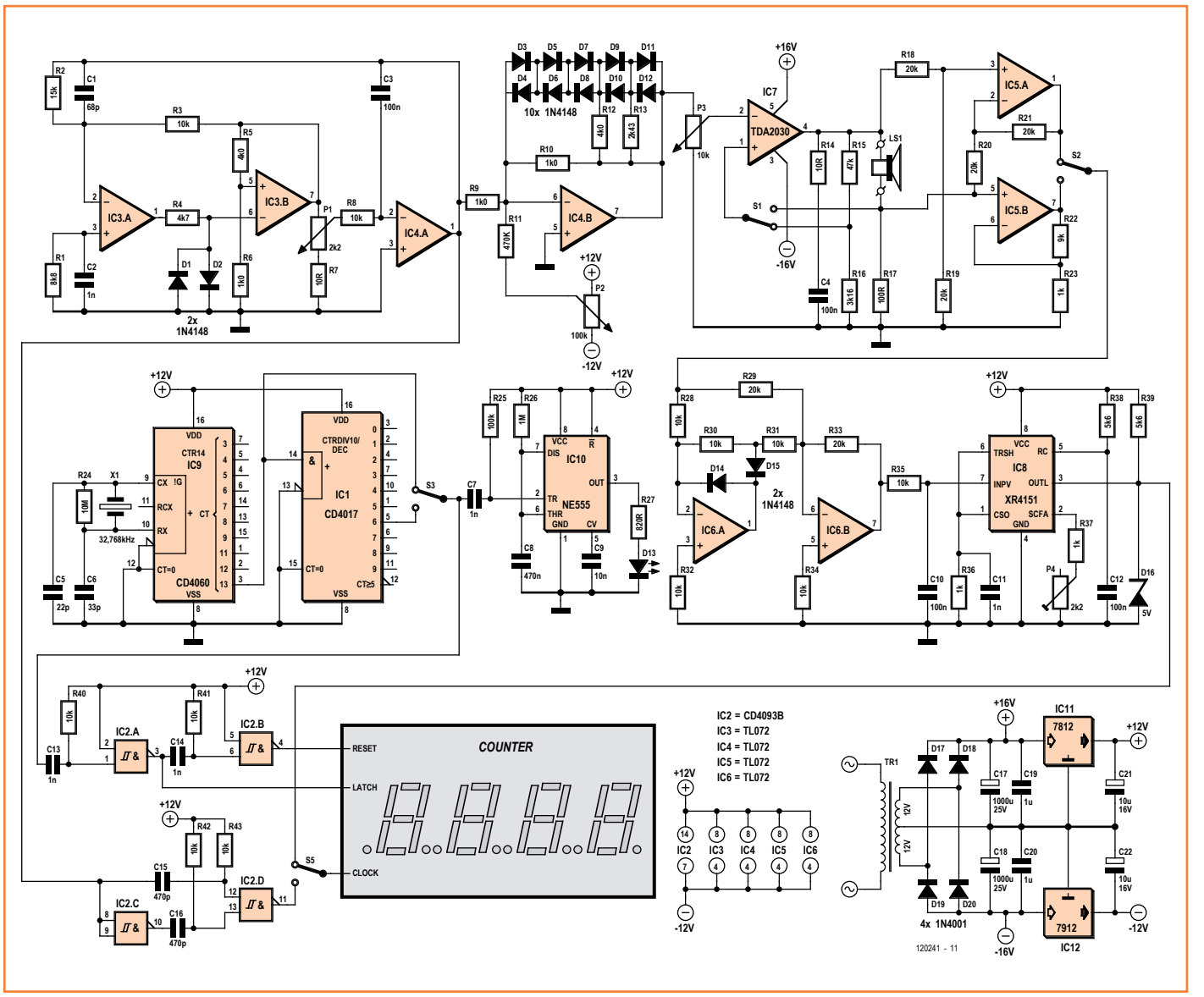
Resonanzmeter

Speziell für Lautsprecher

Von **Jac Hettema** (NL)

Was benötigen wir, um auf einfache Art und Weise die Resonanzfrequenz eines Lautsprechers bestimmen zu können? Nur einen guten Oszillator, einen Verstärker und ein Messgerät, das die Frequenz und die Spannung anzeigt.

Für den Oszillator ziehen wir ein altes Elektor-Projekt heran, den Funktionsgenerator aus 03/1995 (für unser Resonanzmeter verwenden wir nur die Sinusfunktion). Dieser Funktionsgenerator weist den großen Vorteil auf, dass mit nur einem Potentiometer P1 ein sehr weiter Frequenzbereich überschritten wird. Mit einem 10-Gang-Poti kann man so die Frequenz sehr genau einstellen. Zudem ist die Amplitude des Sinussignals sehr konstant und die Verzerrung gering.
Das von IC3 und IC4 erzeugte Sinussignal wird via Poti P3 dem Endverstärker TDA2030 zugeführt. Mit dem Schalter S1 kann man zwischen konstanter Spannung und konstantem Strom wählen. Die zweite Einstellung ist sehr nützlich, um die Amplitudenspitze bei der Resonanzfrequenz zu bestimmen; bei konstantem Strom wird sie sehr schön sichtbar.
Die Spannung über dem Lautsprecher wird von IC5.A gemessen, der als Differenzverstärker mit einem Verstärkungsfaktor von 1 beschaltet ist. IC5.B übersetzt den Strom durch den Lautsprecher in eine korrespondierende Spannung. Mit S2 kann man einen dieser Ausgänge auswählen und den Messwert zum aktiven Gleichrichter um IC6 leiten. Dessen Ausgangsspannung gelangt zu IC8, einem U/F-Wandler XR4151. Dessen Ausgangssignal wiederum gelangt via S5 zu einem Frequenzzähler (Counter). Mit S5 lässt sich einstellen, ob der Frequenzzähler die Frequenz des Sinussignals oder die Spannung über dem Lautsprecher (bzw. den Strom) anzeigen soll.
In seinem Messgerät hat der Autor für den Frequenzzähler ein IC mit der Bezeichnung 74C926 eingebaut. Dieses IC ist leider nicht mehr erhältlich, der Vollständigkeit halber ist aber im Schaltplan doch die dazu gehörende Elektronik (IC1, IC2, IC9, IC10) eingezeichnet. Es wird eine von einem (Uhren-)Quarz gesteuerte Zeitbasis von 10°s und ein Zähler verwendet, womit sich die Resonanz-



frequenz auf 0,1°Hz genau ermitteln lässt. Wenn Sie ein fertiges Frequenzzähler-Modul einsetzen, können die letztgenannten ICs entfallen.

Die Schaltung wird von einem Netzteil mit einem 2x12-V-Trafo und anschließender Gleichrichtung/Stabilisierung versorgt. Lediglich der Endverstärker wird direkt mit der unstabilisierten Spannung betrieben. Bei der Messung muss darauf geachtet werden, dass die Messspannung nicht zu hoch wird. Auch wenn der Lautsprecher eine nominelle Impedanz von nur 4°Ω aufweist, so kann dieser Wert bei der Resonanz zehn Mal so hoch liegen. Wird mit einem konstanten Strom gemessen, dann wird auch die Spannung über dem Lautsprecher zehn Mal größer. Die Spannung muss aber (deutlich!) unter der Clip-Spannung des Verstärkers liegen. Ein Clip-Indikator ist hier bestimmt nicht fehl am Platze.

Zu Beginn der Messung sollte der Pegelsteller P3 auf Null stehen. Dann wird langsam aufgedreht, bis aus dem Lautsprecher gerade hörbar ein Ton erklingt. Nun schalten Sie mit S2 auf Spannungsmessung und wählen die Frequenz mit P1 so, dass die gemessene Spannung maximal ist. Der Messstrom darf dabei nicht vom Startwert abweichen. Ist dies doch der Fall, wiederholen Sie die Messung mit einer niedrigeren Einstellung von P3. Die Frequenz, bei der die maximale Spannung über dem Lautsprecher abfällt, ist seine Resonanzfrequenz.

(120241)

Soziale Roboter

Die Zukunft im Gesundheitswesen?

Johan Hoorn
(Foto: Waag Society CC BY 2.0)

Von **Tessel Renzenbrink** (Redaktion Elektor TTF)

Als ich Alice begegnete, konnte sie schon stehen. Als sie lächelte, lächelte ich zurück. Im gleichen Moment wurde mir bewusst, dass ich einer „Kreatur“ nonverbale Signale gesendet hatte, die meine Signale nicht empfangen kann. Das sagt etwas aus, über Alice und über mich: Ihre Gesichtsmimik ist dem Menschen so vollendet nachgebildet, dass ich sie als soziales Wesen wahrnahm.

Ort der Begegnung mit Alice war die Forschungseinrichtung *Services of Electro-mechanical Care Agencies* (SELEMCA) an der Freien Universität Amsterdam [1]. Dort wird untersucht, wie intelligente technische Systeme, zum Beispiel Roboter, mit Menschen menschlich interagieren können. Das von staatlicher Seite geförderte Projekt soll einen Beitrag zur Beantwortung der Frage leisten, wie der wachsende Bedarf an sozialen Diensten bewältigt werden kann. Die steigende Lebenserwartung führt dazu, dass die Anzahl hilfebedürftiger Personen stetig wächst, während die Anzahl der Helfer stetig sinkt. Damit die Bevölkerung auch zukünftig auf die sozialen Dienste zählen kann, denken Experten über alternative Lösungen nach. Einen Teil der Leistungen, so ist die Idee, könnten intelligente technische Systeme übernehmen. Mit seinem Projekt *I-Care* ist SELEMCA dabei, einem solchen System ein menschenähnliches Gesicht zu geben. Johan F. Hoorn ist wissenschaftlicher Leiter des von SELEMCA durchgeführten Projekts. Enthusiastisch berichtet er über gesteckte Ziele, über die Hürden und Umwege, die genommen wurden und die noch zu nehmen sind: „Die Ansätze unserer Forschungen sind die Intelligenz des Menschen, seine emotionalen Fähigkeiten und seine Kreativität. Wir versuchen, diese für den Menschen urtypischen Wesenszüge auf Maschinen zu übertragen, soweit sie für das Interagieren mit einem Individuum oder einer Gruppe relevant sind. Das ist unser Projekt *I-Care*. Unser zweites, mindestens ebenso wichtiges Arbeitsfeld sind die so genannten Mensch-Maschine-Schnittstellen. I-Care ist fähig, für die Interaktionen mit dem Menschen in viele Gewänder zu schlüpfen.

Menschliche Maschinen

Ein Beleg für die Tiefe der Forschungen ist die Untersuchung der emotionalen Komponente moralischer Motivationen. Roboter, die sich starr und unfehlbar an ethische Gesetze halten, werden von vielen Menschen als gefühlskalte und bedrohliche Rationalisten erlebt. In einem wissenschaftlichen Beitrag über Moral Coppélia, an dem Johan F. Hoorn als Koautor mitwirkte, wird dies am „Wagen- und Fußgängerbrücken-Dilemma” [2] verdeutlicht. Ein losgerissener Eisenbahnwagen rast an einem Abhang ungebremst auf fünf Bauarbeiter zu. Durch spontanes Verstellen einer Weiche könnte der Wagen auf ein Gleis gelenkt werden, auf dem lediglich ein Bauarbeiter getötet würde. Die von einer moralischen Instanz gestellte Frage lautet hier: Handelt ein neutrales Individuum moralisch, wenn es ein Menschenleben opfert, um fünf Menschenleben zu retten? Oder ist es moralischer, unbeteiligt zu bleiben und dem Schicksal seinen

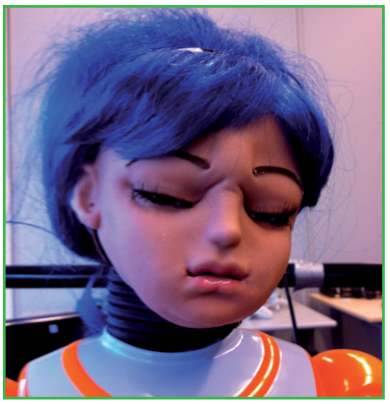
Lauf zu lassen? In einem anderen Szenario steht das Individuum neben einer zweiten Person auf einer Fußgängerbrücke. Wieder rast ein Wagen lebensbedrohlich auf die fünf Bauarbeiter zu. Diesmal lautet die Frage: Ist es moralisch gerechtfertigt, den Menschen von der Brücke auf die Gleise zu stoßen, um den Wagen zu stoppen?
Obwohl in den Szenarien das Verhältnis der Geretteten zu den Opfern übereinstimmend 5 : 1 beträgt, würden zwar die meisten Menschen die Weiche stellen, jedoch würde niemand einen anderen Menschen mutwillig von einer Brücke in den Tod stoßen. Menschen handeln nicht, nachdem sie Verhältniszahlen formelhaft durchgerechnet haben, in ihrer Moral folgen sie ihren Emotionen. Ein schematisch agierender Roboter würde jedoch emotionslos jederzeit ein Menschenleben opfern, wenn er damit fünf Menschenleben retten könnte.
Roboter, die Menschen von Brücken stoßen, werden als Killermaschinen wahrgenommen. Deshalb entwickelten Johan F. Hoorn und seine Mitarbeiter Systeme, die emotionale Intelligenz und moralische Motivation miteinander verknüpfen. Die Systeme simulieren menschliche Charakterzüge bezogen auf die Affekte, auf die moralische Instanz und die Kreativität. In I-Care implementiert stehen sie im Dienst des hilfebedürftigen, von seinem sozialen Umfeld abhängigen Menschen. Wenn ein Patient, der sich das Bein gebrochenen hat, eine Mahlzeit ablehnt, wird ein sozialer Roboter den Willen des Patienten respektieren. Ist ein anderer Patient an Alzheimer erkrankt, wird der Roboter die verminderte Entscheidungsfähigkeit erkennen und die Mahlzeit wiederholt anbieten. Jetzt kann der Roboter seine Kreativität unter Beweis stellen: Er wird dem Patienten nicht stereotyp immer wieder das Tablett vorsetzen. Wahrscheinlich würde dies den Patienten verärgern. Alternativ könnte der Roboter den Löffel füllen und dem Patienten den Löffel reichen.

Alice und DARwIn

Außerdem gibt es da noch das Interface, das Gewand, in dem I-Care nach außen in Erscheinung tritt. „Das Interface“, sagt Johan F. Hoorn, „kann fast jede Gestalt annehmen, die konkret denkbar ist. Das kann ein Roboter ebenso wie ein Spielzeug sein, eine interaktive Puppe oder ein virtueller Akteur auf einem Display. Hinter den Kulissen sind Systeme am Werk, die in ihren Strukturen übereinstimmen. Der äußere Habitus muss nicht zwingend menschenähnlich sein, es geht ausschließlich um ein menschennahes Verhalten. Auch alltägliche Haushaltgeräte wie beispielsweise eine Kaffeemaschine können als Avatar von I-Care fungieren. Beim Benutzer kann durchaus der Eindruck entstehen, dass er mehrere Systeme vor sich hat. In Wirklichkeit interagiert er mit I-Care, denn unser I-Care-System ist ein vielgesichtiges Chamäleon.

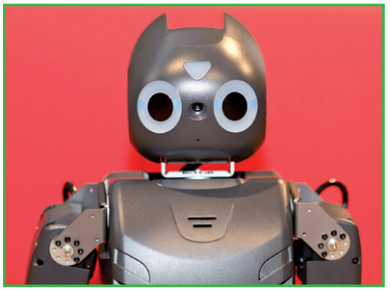
Alice und DARwIn-OP
(Foto: Waag Society CC BY 2.0)

Roboter Alice ist nur einer von vielen denkbaren Avataren, in die I-Care implementiert werden kann. Wegen der menschenähnlichen Züge von Alice fühlt sich der Betrachter inspiriert, über Alice mit I-Care zu kommunizieren. Die Entwicklung ihrer Motorik ist noch längst nicht abgeschlossen, hier muss noch viel Arbeit investiert werden. Inzwischen kann Alice zwar stehen, die Fähigkeit, zielgerichtet Handlungen zu vollführen, ist jedoch noch rudimentär. Ihrem Artgenossen DARwIn-OP (Dynamic Anthropomorphic Robot with Intelligence – Open Platform) muss zugestanden werden, dass er in seiner motorischen Entwicklung weiter fortgeschritten ist.
Auch Johan F. Hoorn weist darauf hin, dass das Interface nicht auf die Gestalt eines Roboters festgelegt ist. Im Entwicklungslabor der SELEMCA arbeiten die Forscher an einem interaktiven Fahrrad. Wenn es um Therapien geht, haben viele an Alzheimer erkrankte Patienten wenig Durchhaltevermögen. Eine Bewegungstherapie auf dem Hometrainer ist ihnen zu monoton, sie wird oft vorzeitig abgebrochen. Johan F. Hoorn und sein Team entwickeln eine virtuelle Landschaft, die dem Patienten suggeriert, dass er durch die Straßen einer Klein-



Der Roboter „Alice".

DARwIn-OP
(Foto: Waag Society CC BY 2.0)



stadt fährt. Über Online-Verbindungen sind sogar virtuelle gemeinsame Fahrradtouren möglich, beispielsweise mit einem Angehörigen des Patienten, der real mit dem Fahrrad unterwegs zur Arbeit ist. Dem Alzheimer-Patienten wird körperliche Bewegung verbunden mit sozialen Kontakten zuteil, ohne den realen Gefahren des Straßenverkehrs ausgesetzt zu sein. Die Kontaktperson erscheint als Avatar auf einem Display, das auf dem Fahrradlenker montiert ist. Indem die Kontaktperson als Interface von I-Care fungiert, wird menschliches Verhalten auf das System übertragen. I-Care in seinen vielfältigen Erscheinungsformen kann den Patienten rund um die Uhr unterstützen, ohne dass dies dem Patienten bewusst wird.

Gemeinsam in die Zukunft

Es ist wichtig, dass I-Care als offene, modulare Plattform gestaltet wird. Johan F. Hoorn: „Unsere Forschungen und Entwicklungen stehen der Welt offen. Wir arbeiten an einem Gerüst, einem Rahmen. Den Rahmen müssen andere mit Inhalten ausfüllen." Wenn die Industrie überlegt, eigene Module anzubieten und diese Module abschotten möchte, um Gewinne zu erwirtschaften, so ist auch dies möglich. „Ich vergleiche unser Projekt gern mit einer Kathedrale, an die Flügel und Seitenschiffe angebaut werden. Jeder Besucher soll einen Ort vorfinden, der seinem Bedürfnis nach religiöser Einkehr gerecht wird. Übertragen auf I-Care wünschen wir uns, dass sich Robotik-Experten, Interface-Designer und Elektromechanik-Spezialisten um unser Entwicklungslabor scharen. Die Popularität von I-Care muss so gesteigert werden, dass klein- und mittelständische Unternehmen ebenso wie die Industrie unsere Forschungsergebnisse zum Wohl des Menschen in Produkte umsetzen."

„Mir ist unverständlich, dass das Interesse jener Kreise auf sich warten lässt. Umso mehr, als ein Markt für die potentiellen Produkte über viele Jahre vorhanden sein wird. Voruntersuchungen der Marktakzeptanz sind eigentlich unnötig, denn wir haben die mutmaßlichen Käufer von Beginn an in unsere Arbeit einbezogen. Von staatlicher Seite erwarten wir mehr Bereitwilligkeit, denn wir leisten einen Beitrag zur Lösung eines wachsenden gesellschaftlichen Problems. Unsere Forschungsergebnisse haben in Hongkong und Südkorea einen höheren Bekanntheitsgrad als hier in Europa. Auf unserem Kontinent hören wir oft: ‚Überaus interessant, einfach herausragend, schön, dass Sie daran arbeiten', doch mehr geschieht nicht. Was hier fehlt, ist ein wirklich innovationsfreundliches Klima. Stattdessen existieren zahllose Kommissionen, die mit ihrem unproduktiven Tauziehen ihre Existenzberechtigung zementieren wollen. Sie stehen zukunftweisenden Innovationen eher im Weg."

„In der Robotik ist bereits manches möglich, doch es mangelt an solidarischer Zusammenarbeit. Alice besticht durch die hochentwickelte Ausdrucksfähigkeit ihrer Gesichtsmimik, doch ihre Körpermotorik ist noch stark eingeschränkt. Bei DARwIn sind die Schwerpunkte umgekehrt verteilt. Die DARPA-Maschinen (Defense Advanced Research Projects Agency) können herbe mechanische Stöße einstecken, ohne dass sie umfallen. Nach einem Stoß setzen sie ihren Weg blindlings fort, als ob nichts geschehen wäre. Von kreativem Verhalten sind sie jedoch noch Lichtjahre entfernt. Viele inselhafte Detailentwicklungen können zwar als ausgereift gelten, doch eine integrierende Plattform fehlt. Wenn sich alle Beteiligten unvoreingenommen an einen Tisch setzen und ihre Arbeiten in ein gemeinsames Vorhaben einbringen, wäre das Ergebnis sicher mehr als erstaunlich".

(130039)gd

SELEMCA ist Teil des Creative Industry Scientific Program (CRISP), gefördert vom Niederländischen Ministerium für Bildung, Kultur und Wissenschaft [3].

Mit Dank an die Waag Society für das Organisieren von *PhDO – Trust me, I'm a Robot* und das Bereitstellen der veröffentlichen Fotos [4].

Weblinks

[1] http://crispplatform.nl/selemca/selemca

[2] http://dare.ubvu.vu.nl/bitstream/handle/1871/38598/Moral%20Coppelia%20IBERAMIA%20Proof%2076370442.pdf?sequence=1

[3] www.crispplatform.nl

[4] http://waag.org/en

Alle Elektor-Artikel der „70er-Jahre“ auf DVD!

NEU

Ein Muss für jeden Elektor-Leser!

ISBN 978-3-89576-263-5
€ 69,00 • CHF 85,60

elektor

Jetzt unter www.elektor.de/70-79 bestellen!

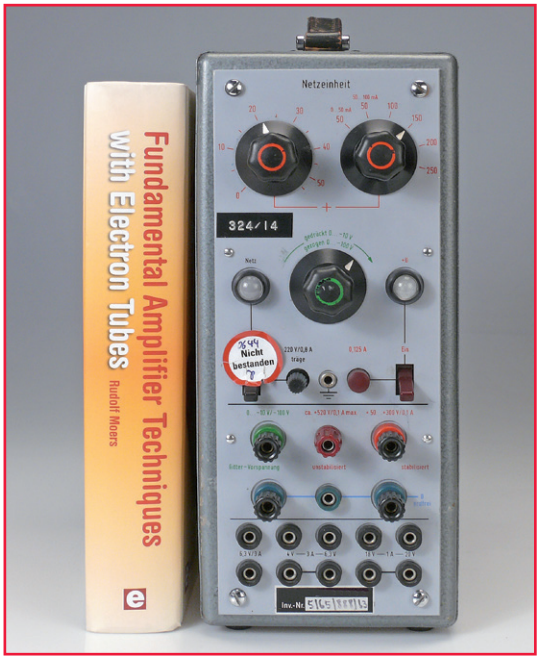
Wandel & Goltermann

NE-171 Röhren-Netzteil (ca. 1963)

Der Traum eines jeden Radio- und Fernsehtechnikers

Von **Jan Buiting**
(Elektor-Redaktion)

Man nehme ein beliebiges Radio aus der Zeit von 1920 bis 1965: Höchstwahrscheinlich hat es ein Problem bei Netzteil, Schalter oder Lautstärkeeinstellung. Beim Netzteil hat man es mit defekten Gleichrichtern (Röhre oder Selen), überhitzten Trafos, durchgebrannten Sicherungen, ausgetrockneten Elkos, Spinnennestern, Mäusekötteln und verheizten Ableit- oder Vorwiderständen zu tun. Leider kann man Fehler nicht beseitigen, bevor man weiß, wo sie stecken. Dazu muss man die Fehlerursache suchen.

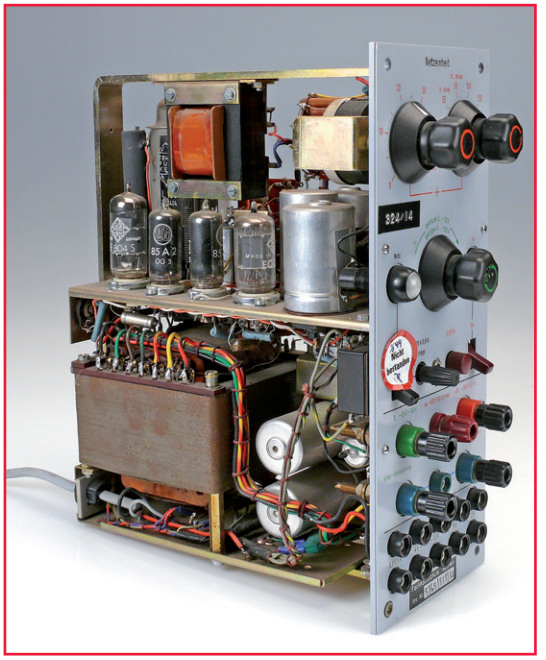


Probleme löst man nicht nur bei Radios, sondern bei jedem Stück Elektronik, indem man verdächtige Bauelemente durch **gute** ersetzt und dann **ausprobiert**, ob es wieder **tut**. Das ist einfach, wenn der Trafo primärseitig einen Kurzschluss hat oder aber ein Lastwiderstand glüht und stinkt. Richtige Probleme hat man aber, wenn ein Netzteil nur tut, wie und wann es will. Etwa so: „Das Radio funktioniert prima, aber nach einer Viertelstunde geht es aus, brummt und es raucht. Aber nur Mittwochs oder am Sonntag Abend, wenn Tante Maria nicht in der Nähe ist.“ Solche Probleme und weniger ernste erfordern die Abtrennung des Netzteils in Radios oder Verstärkern und das Versorgen des Rests mit Strom aus einer **guten** Spannungsquelle. Das Gute daran ist, dass so ein Netzteil (oder was davon übrig ist) durch das Auslöten weniger Leitungen (typischerweise die Heizung und die Anodenspannung) abgetrennt ist. Wer diese Leitungen nicht findet, der sollte besser gar nicht erst versuchen, antike Elektronik oder Röhrengeräte zu reparieren, sondern sich zeitgemäß mit RPi oder iPhone beschäftigen. Nun das Schwierige: Man braucht ein gutes Netzteil

für alle benötigten Spannungen und Ströme. Also 50...300 V Anodenspannung sowie 6,3 V oder 4 V Wechselspannung für die Heizung und −10...−100 V als negative Gitterspannung. Mit anderen Worten: Man braucht ein Multitalent! Glücklicherweise gibt es solche Apparate. Die Spezifikationen im **Kasten** sind kein Traum, sondern Realität. Sie beschreiben die „Netzeinheit – *elektronisch stabilisiert*“ vom Typ NE-171 der schwäbischen Firma *Wandel u. Goltermann*. Anlässlich einer Laborauflösung erhielt ich ein NE-171 – speziell für diesen Retronik-Beitrag.

Made in Germany

Was BMW und Mercedes für die Qualität deutscher Autos bedeuten, entspricht *Wandel & Goltermann* im Bereich elektronischer Test- und Messgeräte in der Zeit nach dem zweiten Weltkrieg bis zum



IC-Zeitalter (d.h. bis zur Übernahme durch Wavetek 1999). Geräte von W & G waren immer in der oberen Preiskategorie angesiedelt. Und sie sind es noch, wie man auf eBay sieht. Letzteres reflektiert ihren hohen Sammlerwert.

Zwei Handbücher

Auch wenn man einige Infos zum NE-171 via Internet - speziell bei Jogi’s Röhrenbude [1] - finden kann, so ist das kein Ersatz für das „richtige“ Handbuch. Ich hatte das Glück, dass sogar zwei Handbücher beim NE-171 dabei waren. Das ältere ist mit „27. Juni 1958“ gestempelt. Es beschreibt die Produktions-Serie C - daher meine Vermutung, dass das Gerät schon weit früher auf den Markt kam. Das Handbuch ist eigenartig: eine Matrizenkopie eines maschinengeschriebenen Originals. Die neuere Version vom September 1962 ist richtig gesetzt und gedruckt. Das Buch mit dem leuchtendroten Einband betrifft die Serien F, G und H. Meine Nr. 16432H spricht für ein Produktionsdatum von 1962, möglicherweise auch 1963.

„Nicht bestanden“ (Juni 2006)

Trotz seines Alters sieht mein NE-171 immer noch sehr gut aus. Nur innen ist es etwas staubig. Auch wenn der Knebel des Netzschalters mit einem Aufkleber „Nicht bestanden“ versehen war, hatte ich doch keinerlei Befürchtung, dass es fehlerhaft sein könnte. Dafür gibt es drei Gründe:

- Es stammt von Wandel & Goltermann.
- Es stammt aus einer Umgebung, wo frühere und aktuelle Elektor-Leser sorgfältig mit so etwas umgingen.
- Ich habe einen Regel-Trenntrafo, der für den angemessenen Soft-Start nach Jahren des Schlafs sorgt.

Ich brach also das Siegel und schaltete das Gerät ein. Und tatsächlich lief es sofort. Nachfolgend zwei mögliche Erklärungen für so einen Warn-Sticker auf einem funktionierenden Gerät:

- In der 44. Woche 2006 schloss jemand aus der Aufkleber-Abteilung pflichtgemäß ein Fluke-Digitalmultimeter neuesten Typs an den Ausgang 1 (50 bis 300 V) an und schaltete das NE-171 ein. Als sich dann eine Spannungsspitze zeigt, schaltete der erschrockene Prüfer das Gerät sofort aus und zückte einen rot/weißen Warn-Sticker. Nachdem er den jährlichen Prüfbericht an die Buchhaltung geschickt hatte, wurde die Netzeinheit NE-171 als defekt deklariert und bis zur Verschrottung oder zum Transport auf die Mülldeponie eingelagert. Glücklicherweise stach das Teil sieben Jahre später einem Elektor-lesenden Labor-Mitarbeiter und engagiertem Retronik-Fan ins Auge. Ich bekam Post.
- Wie 1., aber: „war nicht gleich eine Spannung zu registrieren, …”

Und weiter geht’s...

Kompakt und leistungsfähig

Früher gab es viele einstellbare Hochvolt-Netzteile von verschiedenen Herstellern. In der Zeit von 1950 bis 1970 wurden sie für Röhrentechnik gebaut, sei es für Laborarbeiten oder für den Service. Sie waren durchweg voluminös, wie auch das Van der Heem 8619 Röhren-Netzteil, das in der April-Ausgabe 2007 von Elektor [2] beschrieben wurde. Damit verglichen ist das W & G NE-171 ein kompaktes Leichtgewicht, auch wenn es etwas weniger Spannung und Strom liefert. Mir gefällt seine Bauweise: fast wie ein dickes Buch. Es wird mir noch gute Dienste bei der Reparatur von Röhrenradios leisten.

EL156 ausgereizt

Aus elektronischer Sicht ist die Schaltung des einstellbaren Hochspannungsteils eher konventionell. Eine Röhre (Rö1) dient als Längsregler und ein Regelverstärker (Rö2) stellt zusammen mit einigen Widerständen und einem Poti die Ausgangsspannung ein. Die Referenzspannung von –85 V stammt von der Stabilisatorröhre 85A2.

Das Besondere steckt nicht in der Schaltung, sondern in der Wahl der Bauelemente. Wo viele andere einstellbare Netzteile mit einem Paar Röhren des Typs EL34 bzw. 6CA7 als Längsregler arbeiten, wurde hier eine EL156 in Kombination mit der „steilen“ Pentode EF804S eingesetzt. Die EL156 hat ordentliche Daten: Kathodenstrom = max. 180 mA, Anodenverlustleistung = max. 40 W und Anodenspannung bis zu 800 V. Diese Röhre ist wohl eine der besten Bauteil-Entwicklungen von Telefunken. In ihr kulminiert die ganze Erfahrung dieser Firma, wie das Datenblatt [3] belegt. Eine EL156 eignet sich auch prima als Endröhre eines Audioverstärkers.

NE-171 Spezifikationen	
Gleichspannung 1: Elektronisch stabilisiert und erdfrei, Grobeinstellung in 6 Stufen, Feineinstellung stetig	50 V...300 V
Maximal entnehmbarer Strom I_1 (bei I_2 = 0)	100 mA
Gleichstrom-Innenwiderstand gemessen bei 175 V und zwischen 20 mA und 100 mA	≤2 Ω
Unsicherheit der eingestellten Spannung	≤1 % ±2 V
Brummspannung bei 300 V und 100 mA	ca. 0,1 mV_{eff}
Änderung der Gleichspannung 1 (bei ± 10 % Netzspannungsänderung und 300 V und 100 mA)	ca. ±300 mV
Gleichspannung 2: etwa 520 V (bei 50 mA) Unstabilisiert und erdfrei. Minuspol, gemeinsam mit Gleichspannung 1, Maximal entnehmbarer Strom I_2 (bei I_1 = 0)	100 mA
Gleichspannungs-Innenwiderstand	ca. 1 kΩ
Brummspannung bei 50 mA	ca. 0,5 V_{eff}
Gleichspannung 3 Elektronisch stabilisiert und erdfrei, Pluspol liegt am Minuspol von Spannung 1, Maximal entnehmbarer Strom	0...≥ -10 V / 0...≥ -100 V (umschaltbar)
0...-10 V	bis Kurzschluss (ca. 1,5 mA)
0...-100 V	bis Kurzschluss (ca. 3,5 mA)
Brummspannung (bei 1 mA)	ca. 20 μV_{eff}
Innenwiderstand 0...-10 V	≤ 8 kΩ || 32 µF
0...-100 V	≤ 30 kΩ || 32 µF
Änderung der Gleichspannung 3 bei ± 10 %	
Netzspannungsänderung	ca. ± 0,1 %
Wechselspannungen, 3 getrennte, erdfreie Heizausgänge	4 / 6,3 V 3 A 6,3 V 3 A 18 / 20 V 1 A
Röhrenbestückung	EL156, EF804S, ECC82, 85A2
Netzspannung	220 V, 45...60 Hz
Leistungsaufnahme (voll belastet)	ca. 140 VA
Abmessungen	140 x 315 x 249 mm
Gewicht	ca.10 kg

Die Ausgangsspannung wird beim NE-171 durch einen Wahlschalter mit sechs Bereichen in 50-V-Schritten eingestellt. Mit einem Poti werden zusätzlich 0...50 V dazu gepackt. Bei meinem Gerät klemmt der Bereichsschalter bei 150 V, wodurch ich nur Spannungen zwischen 150 und 200 V einstellen kann. Irgendwann werde ich das in Ordnung bringen. Optisch wirkt die Mechanik des Schalters ganz in Ordnung.
Für den von 50 bis 300 V einstellbaren Ausgang gibt es einen separaten roten Schalter und eine rücksetzbare Sicherung für 125 mA. Das neuere Handbuch warnt deutlich davor, diesen zweiten Schalter umzulegen, bevor das Gerät eine Minute warm gelaufen ist. Andernfalls hat man es mit Überspannungen zu tun oder es zeigt sich keine Spannung. Das dürfte also der Grund sein, warum das Gerät ein „Nicht bestanden" erhielt, da der Prüfer wohl nicht so ganz firm in Sachen Röhrentechnik war. Er hätte den Aufkleber „Nicht verstanden" redlich verdient...

Es gab übrigens keinerlei Probleme mit den vorhandenen Mehrsektor-Elkos. Ich stellte fest, dass statt des laut Schaltbild originalen Brückengleichrichters B250C150 hier Dioden vom Typ BY179 verbaut waren. Und die BY179 stammt definitiv aus den 1970ern!
Nicht zuletzt sei ein großes Plus der Netzeinheit NE-171 erwähnt: Alle Ausgänge sind gegenüber dem Gehäuse potentialfrei.

NE-171 im Doppel

Die einstellbaren Ausgänge von zwei NE-171-Netzteilen können entweder parallel oder in Serie geschaltet werden, um mehr Strom (0...200 mA) oder eine höhere Spannung (0...600 V) zu erzielen. Für die Parallelschaltung muss auf der Rückseite ein Stecker abgezogen und ein spezielles Verbindungskabel eingesteckt werden. Das Kabel kann man leicht selbst machen, wenn man an die speziellen fünfpoligen Stecker herankommt.

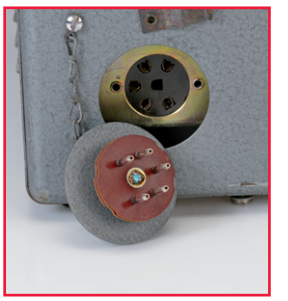
Praxis

Immer wenn ich ein altes Röhrenradio untersuche, trenne ich zuerst konsequent das Netzteil samt Siebelkos ab. Dann wird die eigentliche Elektronik vom NE-171 versorgt — nach dem Studium der Schaltung versteht sich. Zuerst küsse ich das antike Teil wach, indem für 15 Minuten lediglich die Heizung aktiviert wird. Hier zeigen sich dann schon mal erste Fehler der Heizwendeln oder ihrer Verdrahtung.
Als nächster Schritt kommt die Anodenspannung. Ich beginne mit nur 50 V und erhöhe dann gemächlich auf die Nennspannung (üblicherweise 250 V). In 80 % der Fälle wird das Radio dann halbwegs laufen. Nun werden die Elkos neu formiert, indem ich sie langsam mit 50...300 V beaufschlage und dabei einen Feuerlöscher griffbereit habe ☺. Dann arbeite ich mich rückwärts bis zum Eingang des Gleichrichters durch. Bei Zweifeln kommt ein Milliamperemeter in die Anodenleitung.
Die ganze Prozedur kann man ruhig mit einem guten Netzteil durchziehen, das Vertrauen einflößt. Wenn die Reparaturen der Audio-, ZF- und HF-Stufen eines Radios abgeschlossen sind, finde ich es fast schon etwas schade, das NE-171 auszuschalten und Schritt für Schritt die alten Verbindungen wieder herzustellen – und dabei natürlich immer alle Spannungen mit einer Hand in der Tasche zu überprüfen.

(130030)

Weblinks

[1] www.jogis-roehrenbude.de, suche nach NE-171
[2] Einstellbares Netzteil für hohe Spannungen, Retronik, Elektor April 2007, www.elektor.de/075036
[3] www.hifitubes.nl/weblog/wp-content/telefunken-el156.pdf



ESTD 2004

Retronik ist eine monatliche Rubrik, die antiker Elektronik und legendären Elektor-Schaltungen ihre Referenz erweist. Beiträge, Vorschläge und Anfragen telegrafieren Sie bitte an Jan Buiting (editor@elektor.com).

Hexadoku Sudoku für Elektroniker

Um dieses Hexadoku zu lösen, brauchen Sie keinen Computer und nicht einmal irgendwelche Elektronik. Ihr bestes Tool sind Ihre kleinen grauen Zellen! Beweisen Sie sich selbst, dass Sie die Herausforderung meistern. Dann sollten Sie uns die Lösung zusenden. Hier warten nämlich schon vier schöne Gutscheine auf die Gewinner…

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 **Eurocircuits/Elektor-PCB-Service-Gutschein im Wert von 100 €** und drei **Elektor-Bücher-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 30. Juni 2013!

Die Gewinner des Hexadokus aus der April-Ausgabe stehen fest!

Die richtige Lösung ist: **934CB**.
Der Eurocircuits/Elektor-PCB-Service-Gutschein im Wert von 100 € geht an: David Smart (USA).
Einen Elektor-Gutschein über je 50 € haben gewonnen: Paul Baak, Karsten Krummeich und Joseph Reding.
Herzlichen Glückwunsch!

5		D		9	0		A	6	7		B		1		
C	E		3		8		5						A		
		2								1		F		0	9
A				7		C	F			D			4		
F	4	C		3		5	8		E						1
0			6		7		B	C	D	9		2	5		
E	7		2				1	4		8	A				D
	1		B						F	5					8
		3	7	5			D		1	2	8	0		A	B
			5	B		7					D	C			
2	6	1			C					B				D	3
B		E				8	3				9	1			7
			F			B	6	9	8	E				1	
	8			1	5		E				4		B		0
		9			2			D	B		6			3	
				8	D				5	F	7	9		4	A

9	C	2	4	B	D	E	F	7	6	A	0	5	8	3	1
5	E	B	D	3	1	A	9	4	F	8	C	6	7	0	2
A	F	3	0	8	4	6	7	D	1	2	5	E	9	B	C
6	1	7	8	C	0	5	2	E	3	9	B	4	A	D	F
7	B	6	E	4	A	F	1	9	8	0	3	C	D	2	5
8	D	9	3	5	2	B	0	6	A	C	1	7	E	F	4
C	0	1	F	6	8	7	E	5	4	D	2	A	B	9	3
2	A	4	5	9	C	D	3	F	7	B	E	8	0	1	6
3	2	8	1	A	5	0	D	B	C	E	F	9	6	4	7
B	4	5	7	E	F	8	6	0	2	3	9	1	C	A	D
D	6	F	A	2	7	9	C	1	5	4	8	B	3	E	0
E	9	0	C	1	3	4	B	A	D	6	7	F	2	5	8
F	3	E	B	7	6	C	A	2	0	5	4	D	1	8	9
0	5	C	6	D	E	1	8	3	9	F	A	2	4	7	B
1	8	A	2	F	9	3	4	C	B	7	D	0	5	6	E
4	7	D	9	0	B	2	5	8	E	1	6	3	F	C	A

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

elektor GOLD membership

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 als PDF zum sofortigen Download unter www.elektor-magazine.de (für PC/Notebook) oder via App (für 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:** Der Online-Zugang zum neuen Community-Bereich www.elektor-labs.com bietet Ihnen zusätzliche Bauprojekte und Schaltungsideen.
- **Extra:** Die neue Elektor-Jahrgangs-DVD (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.

elektor GREEN membership

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

LCD-Graphik II, verkettete Strukturen II und die Fädelsprache LAX

1 AVR-Programmierung 4

In diesem neuen vierten Band der erfolgreichen Buchreihe zur Programmierung von AVR-Mikrocontrollern wird die LCD-Graphik aus Band 3 weiterentwickelt. Hinzu kommen das Füllen von Polygonen, die Zuordnung von Pixelkoordinaten zu Graphikobjekten und die Verwendung des Displays als Textfenster. Aufbauend auf der Darstellung der inneren Mechanik von Fädelsprachen im vorigen Band wird außerdem die Fädelsprache LAX vorgestellt und implementiert.

320 Seiten (kart.) • ISBN 978-3-89576-232-1
€ 46,00 • CHF 57,10

Schaltungsprojekte für Profis

2 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

Theorie und Praxis mit WinFACT und Multisim

3 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

Apps programmieren – Schritt für Schritt

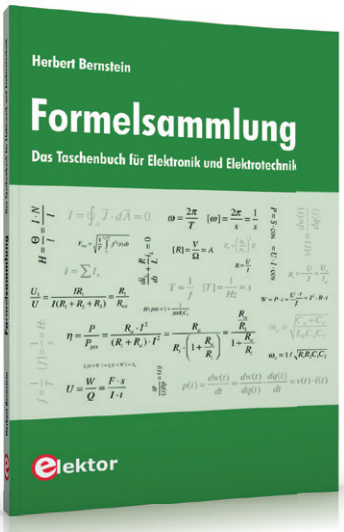
4 Android

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.

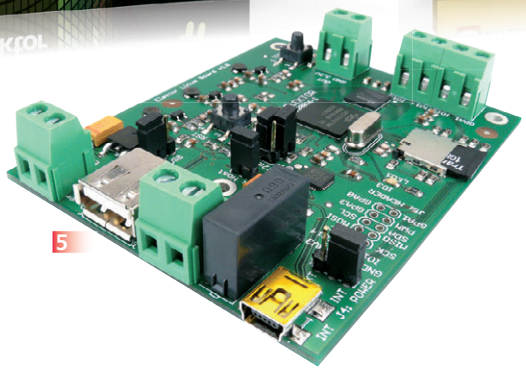
256 Seiten (kart.) • ISBN 978-3-89576-252-9
€ 34,80 • CHF 43,20

4

6



7



5

8

Bestücke und getestete Platine

5 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

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

Taschenbuch für Elektronik und Elektrotechnik

7 Formelsammlung

Diese „Formelsammlung“ beinhaltet alle wichtigen Details für Ingenieure, Techniker, Meister und Facharbeiter in der Elektrotechnik und Elektronik, die in Forschung, Entwicklung und Service tätig sind. Die logische Gliederung in zehn Kapitel vereinfacht das Nachschlagen und Aufsuchen der gewünschten Themen. In den einzelnen Kapiteln finden Sie immer die notwendigen mathematischen und physikalischen Formeln sowie die wichtigsten Tabellen.

271 Seiten (kart.) • ISBN 978-3-89576-251-2
€ 29,80 • CHF 37,00

Kompletter Elektor-Jahrgang 2012 auf DVD

8 Elektor-DVD 2012

Die neue Elektor-Jahrgangs-DVD enthält alle Artikel des Jahrgangs 2012. Sie verfügt über eine sehr übersichtlich gestaltete Benutzeroberfläche. Mit der Elektor-DVD 2012 können Sie Platinenlayouts in perfekter Qualität drucken; diese Layouts mit einem Zeichenprogramm verändern; die Schnellsuchfunktion benutzen, mit der Sie in den einzelnen Artikeln oder im ganzen Jahrgang nach Wörtern, Bauteilen oder Titeln suchen können; Schaltbilder, Platinenlayouts, Illustrationen, Fotos und Texte exportieren.

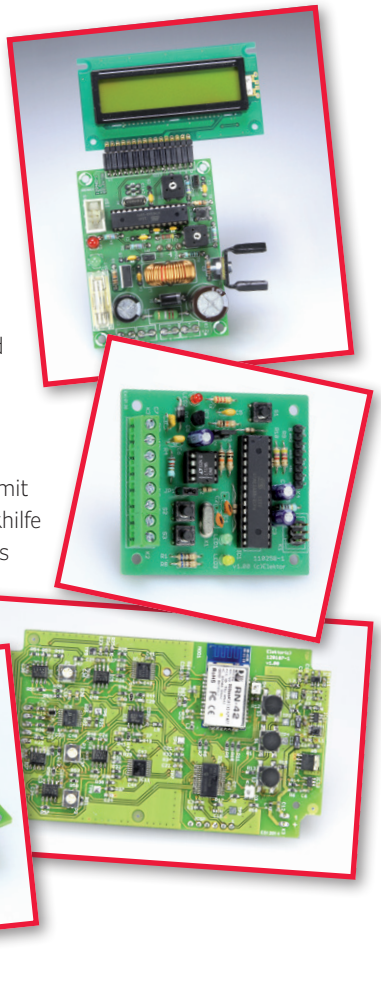
ISBN 978-90-5381-273-0
€ 27,50 • CHF 34,10

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



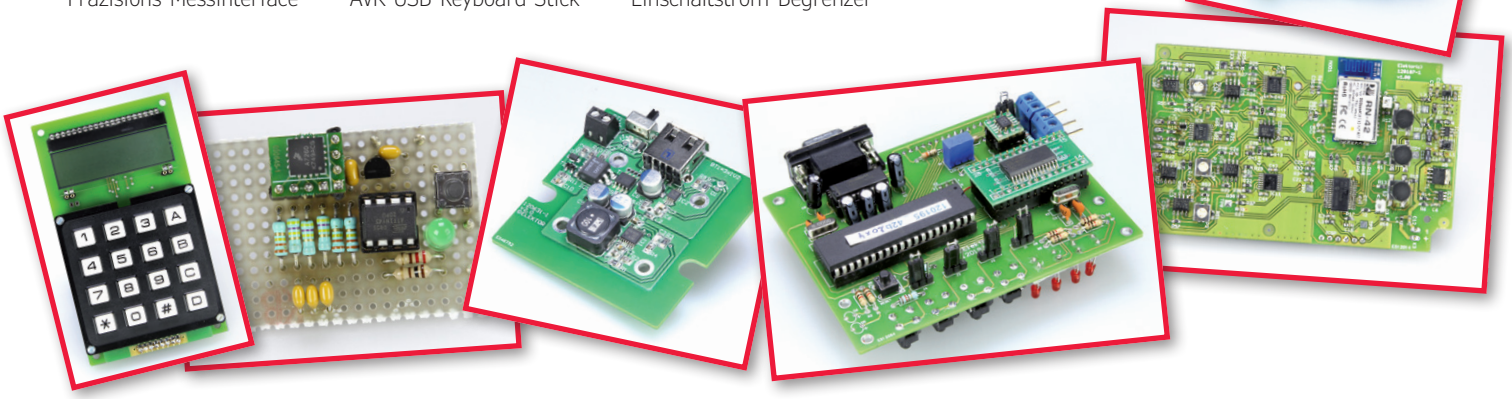
Elektor-Sommer-Doppelausgabe 2013

Doppelter Umfang, doppelter Elektronik-Spaß!

Im nächsten Monat erscheint wieder das Sommer-Doppelheft von Elektor. Auch in diesem Jahr präsentieren wir in der traditionsreichen Ausgabe frische Ideen und neue Projekte, originelle, praxisnahe Schaltungen und erprobte Software. Die Sommer-Doppelausgabe ist eine gelungene Mixtur aus großen und kleinen Projekten, sie lädt Sie zu einer sommerlichen Rundreise durch das vielgestaltige Land der Elektronik ein. Unsere Entwickler und Redakteure sind gerade dabei, ihr Bestes zu geben!

Aus dem Inhalt:

CAN-Tester – ElektorBus-Schrittmotor-Board – CDI-Zündung für Motorräder – Klasse-D-Audioverstärker mit 555 – Servotester – EKG-Monitor für Android – USB-Power-Pack – Numitron Arduino-Uhr – Einparkhilfe – Mehrkanal-Temperaturlogger – IR-Fernbedienung für Android – Superpräzise Digitaluhr – Universelles Präzisions-Messinterface – AVR-USB-Keyboard-Stick – Einschaltstrom-Begrenzer



Elektor Juli/August 2013 erscheint am 26. Juni 2013. Änderungen vorbehalten!

Spring into summer savings!

$25 off CC Gold

You'll have plenty of summer reading with *Circuit Cellar*'s CC Gold issues archive.

A lifetime of electronics engineering projects, tips, and analysis, packed onto a portable, USB flash drive. Keep your archive current with a digital subscription and download new issue PDFs directly to the drive! Plus, with 32 GB of storage, there's plenty of room for your own notes and projects.

Offer ends 6/30/13

*Complete archive includes all issues in print through time of purchase.

Visit www.cc-webshop.com to purchase.

Unsere Shopsprachen:

reichelt.de elektronik

✓ über 40 Jahre Erfahrung
✓ schneller 24-Std.-Versand
✓ über 45.000 Produkte am Lager
✓ kein Mindermengenaufschlag

Kundenbewertungen:

97.75% zufriedene Kunden

4.90 / 5.00

Rund 98 % unserer Kunden sind vom **reichelt**-Service überzeugt*

* Quelle: Shopauskunft.de (30.04.2013)

Hochwertige Marken-Steckverbinder

Professionelle Qualität zu attraktiven Preisen!

Der powerCON TRUE1

ist ein verriegelbarer Gerätesteckverbinder und ersetzt Kaltgerätestecker dort, wo eine sehr robuste und sichere Stromverbindung benötigt wird.

NEUTRIK

5,95 NEUTRIK NAC3MX

- ✓ Netzsteckvorrichtung mit Nulleiter
- ✓ Strombelastbarkeit bis 16 A
- ✓ schnelles und einfaches Verriegelungssystem
- ✓ extrem robust und zuverlässig
- ✓ einzigartige Neutrik Kabelzugentlastung

Kabelbuchse mit Verriegelung
NEUTRIK NAC3FX 5,95
Kabelstecker mit Verriegelung
NEUTRIK NAC3MX 5,95

ENEC Zertifizierung

UNSER TECHNIK-TIPP

EtherCon

RJ45-Stecksystem

Metall-Steckergehäuse zur Aufnahme eines fertig konfektionierten Cat.5-Kabels.

- optimale Zugentlastung
- Verriegelungsschutz

NEUTRIK NE-8MC RJ45 Stecker 3,10

RJ45-Flanschbuchsen

NEUTRIK NE-8FDP RJ45 auf RJ45 9,85
NEUTRIK NE-8FDV RJ45 auf LSA 9,10

USB-Einbaubuchse

für den Rack-/Paneleinbau

Ideal zur Integration von digitalem Equipment in Audio-Systeme.

NEUTRIK NAUSB 5,10

Geräteeinbau-Buchse

- 6,35 mm Flachsteckzungen
- Power OUT
- 2-polig +PE
- 43,0 x 35,4 mm

Einbaubuchse
NEUTRIK NAC3FPX 5,15
Dichtklappe für NAC3FPX
NEUTRIK SCNAC-FP 1,10

Geräteeinbau-Stecker

- 6,35 mm Flachsteckzungen
- Power IN
- 2-polig +PE
- 26,0 x 31,2 mm

Einbaustecker
NEUTRIK NAC3MPX 2,85
Dichtklappe für NAC3MPX
NEUTRIK SCNAC-MP 1,10

Gerätestecker-Einbaubuchse-Kombination

- Power-IN und Power-OUT in einem Gehäuse
- 6,35 mm Flachsteckzungen
- 2-polig +PE
- 61,0 x 38,3 mm

Duplex-Einbaustecker
NEUTRIK NAC3PX 6,95

Alle NEUTRIK-Produkte:

Einfach QR-Code per Smartphone scannen oder Kurzlink eingeben:

http://rch.lt/9z

Dichtklappe für NAC3PX
NEUTRIK SCNAC-P 1,15

Jetzt bestellen! **www.reichelt.de**

Bestell-Hotline: **+49 (0)4422 955-333**

Neue Katalogausgabe!
Kostenlos – Jetzt anfordern!

reichelt elektronik
Ihr kompetenter Partner für
Bauelemente • Stromversorgung • Messtechnik • Werkstattbedarf
Haus- & Sicherheitstechnik • Netzwerk- & PC-Technik • Sat-/TV-Technik